UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

**METHODOLOGY FOR IMPLEMENTATION
OF SYNCHRONIZATION STRATEGIES FOR
WIRELESS SENSOR NETWORKS**

DOCTORAL THESIS

Francisco Tirado Andrés

TELECOMMUNICATION ENGINEER

2020

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE
TELECOMUNICACIÓN

METHODOLOGY FOR IMPLEMENTATION OF SYNCHRONIZATION STRATEGIES FOR WIRELESS SENSOR NETWORKS

A dissertation

by

FRANCISCO TIRADO ANDRÉS

under the supervision of

DR. ALVARO ARAUJO PINTO

submitted in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

by Universidad Politécnica de Madrid

Madrid, 2020

How to cite this work:

F. Tirado-Andrés. “Methodology for implementation of synchronization strategies for wireless sensor networks” Ph. D dissertation. Escuela Técnica Superior de Ingenieros de Telecomunicación (ETSIT), Universidad Politécnica de Madrid, Spain, 2020.

Correspondence: frta@b105.upm.es

This work was performed in:

B105 Electronic Systems Lab

Escuela Técnica Superior de Ingenieros de Telecomunicación

Universidad Politécnica de Madrid

b105.upm.es

www.etsit.upm.es

www.upm.es

This work is licensed under a [Creative Commons Attribution 4.0 \(CC BY\) International License](https://creativecommons.org/licenses/by/4.0/). Any further distribution of this work must maintain the attribution to the author and the title of the work

Summary

Entre las dificultades se esconde la oportunidad

Albert Einstein

Over the last few years, we have seen and are experiencing how wireless communication devices are changing from being something exceptional to being commonplace in our daily lives. Advances in wireless communications, the miniaturization of electronic devices and the increasing capabilities of these devices have made wireless networks a very common resource in the industry and consumer electronics. Wireless Sensor Networks (WSNs) are spatially distributed sensors that monitor physical or environmental conditions and share their data over the network. WSNs are also characterized by the limitation of their resources as the need for low power consumption as they usually run on batteries, the need for low cost, or its low data throughput, among others. That is why in WSNs the resolution of some problems already solved in wired networks becomes a challenge, such as the implementation of time synchronization strategies.

This time synchronization is necessary for the successful operation of many applications based on WSNs, but each node is independent and has its own clock source. Therefore, it is not enough to synchronize the clocks of the different nodes at the beginning of the application because many factors can cause them to be out of sync, for example, the frequency of oscillation of the clock sources may be slightly different, temperature changes, battery status, etc.

Besides, the level of demand on time synchronization in WSNs depends on the type of application. There is a wide variety of applications in WSNs: applications that attempt to merge data from all nodes to obtain an overall picture of the scenario, applications that seek to develop communication strategies by making use of the medium access control layer, or applications that need to coordinate actions among all nodes to carry out cooperative tasks. Each application may have different requirements in terms of time synchronization that may range from very lax requirements, where several seconds offsets are allowed, too much more demanding requirements. It should also be noted that each developer may use a different hardware platform with different resources.

When developers of applications for WSNs want to implement a synchronization strategy for their system they face all the above issues when making their decision. Today this decision was made by developing their own synchronization protocol or by adapting an already created protocol to their needs. After the state-of-the-art review carried out, it has been proven that no methodology helps the developer to decide to choose a synchronization protocol for WSNs that is adapted to the specific needs of each case.

That is why this Doctoral Thesis has been developed in order to implement a methodology for the implementation of synchronization strategies for wireless sensor networks.

Taxonomy has been developed to organize and generate a structure that accommodates not only existing synchronization strategies but also those that may appear in the future. The development of this specific taxonomy has been done to facilitate and assist in the development of the methodology. The taxonomy is based on nine groups where each synchronization strategy must be cataloged.

The methodology is based on a flow chart that guides the user through the ten most important themes and parameters regarding time synchronization in WSNs. These blocks to analyze are objective, power consumption, monetary cost, security, network topology, message distribution, accuracy, stability, and software, and hardware abstraction. Each of the blocks has a level scale that allows its comparison and its graphic representation. With this methodology, the aim is that any user, regardless of their level of knowledge in the subject of time synchronization in WSNs, has the possibility of selecting a synchronization strategy that adapts to their specific needs and requirements, reducing the time of their selection.

In addition to the methodology, we also present a Time Synchronization Search Tool (TSST). This tool is composed of several blocks: the administration block, the user block and the database that interrelates both. In the administration block, the experts in synchronization strategies parameterize each strategy before incorporating it into the database. In the user block, users looking for a synchronization strategy that suits their needs to make use of a parameterized filter. This selection filter shows all those synchronization strategies that meet the requirements and needs that the user has imposed. The tool displays the results in table form and graphic form making the display and comparison of the strategies faster.

Once the methodology and the tool have been developed, it is necessary to validate them to verify their correct operation. The validation has been done through two use cases with different levels of exigency concerning time synchronization. The applications chosen as use cases are structure health monitoring and environmental monitoring. For the monitoring of structures, a theoretical study and simulations have been carried out with real data on how time synchronization affects the identification of the frequencies and modal forms of the structures. Also, tests have been carried out on a bridge in northern Spain. In the application of environmental monitoring, a study and real tests have been conducted on the behavior of different clock sources in the presence of temperature changes. This has allowed us to see how decision making in the early stages of design can help us to minimize the problem of time synchronization in WSNs.

After the good results obtained in the different studies, simulations, and experiments performed in the validation, it is concluded that the developed methodology works and helps in the selection of time-synchronization strategies for WSNs.

Resumen

Nada tarda tanto como aquello que no se empieza

Alain

Durante los últimos años hemos visto y estamos experimentando como los dispositivos con comunicaciones inalámbricas pasan de ser algo excepcional a ser algo habitual en nuestra vida cotidiana. Los avances en comunicaciones inalámbricas, la miniaturización de los dispositivos electrónicos y el aumento de las capacidades de estos dispositivos han hecho que las redes inalámbricas se conviertan en un recurso muy común en la industria y la electrónica de consumo. El subgrupo de las redes de sensores inalámbricas, en inglés Wireless Sensor Networks (WSNs) se caracterizan por ser sensores espacialmente distribuidos que monitorizan condiciones físicas o ambientales y comparten sus datos a través de la red. Las WSNs también se caracterizan por la limitación de sus recursos como necesidad de bajo consumo ya que suelen funcionar con baterías, la necesidad de que sean de bajo coste, o su baja tasa de transmisión de datos entre otros. Es por ello por lo que en WSN la resolución de algunos problemas ya resueltos en redes cableadas se convierte en un reto, como la implementación de estrategias de sincronización temporal.

Esta sincronización temporal es necesaria para el correcto funcionamiento de muchas aplicaciones basadas en WSNs, pero cada nodo es independiente y tiene su propia fuente de reloj. Por tanto, no basta con sincronizar los relojes de los diferentes nodos al inicio de la aplicación porque son muchos los factores que pueden hacer que se desincronicen, por ejemplo, la frecuencia de oscilación de las fuentes de reloj puede ser ligeramente diferente, los cambios de temperatura, el estado de la batería, etc. A esto hay que añadir problemas heterogéneos de diferente naturaleza que van desde utilizar diferentes plataformas hardware con diferentes recursos hasta el desarrollo de diferentes aplicaciones con diferentes requisitos.

Cuando los desarrolladores de aplicaciones para WSNs desean implementar una estrategia de sincronización para su sistema se enfrentan a todos los problemas anteriores a la hora de tomar su decisión. Hoy en día esta decisión pasaba por desarrollar su propio protocolo de sincronización o por adaptar un protocolo ya creado a sus necesidades.

Es por ello por lo que se ha llevado a cabo el desarrollo de esta Tesis Doctoral con el fin de desarrollar una metodología para la implementación de estrategias de sincronización para redes de sensores inalámbricas.

Se ha desarrollado una taxonomía con el objetivo de organizar y generar una estructura que acomode no solo a las estrategias de sincronización existentes, sino también a aquellas que puedan aparecer en el futuro. El desarrollo de esta taxonomía específica se ha hecho para facilitar y ayudar al desarrollo de la metodología.

La metodología se basa en un diagrama de flujo que guía al usuario a través de los diez aspectos y parámetros más importantes en referencia a la sincronización temporal en WSNs. Estos bloques para analizar son: objetivo que persigue la sincronización, consumo de energía, coste monetario, seguridad, topología de la red,

distribución de mensajes, exactitud en el protocolo de sincronización, estabilidad de la fuente de reloj, abstracción de software y abstracción hardware. Cada uno de los bloques tiene una escala de nivel numérica que permite su comparación y su representación gráfica. Con esta metodología se persigue que cualquier usuario, independientemente de su grado de conocimiento en el tema de sincronización temporal en WSNs, tenga la posibilidad de seleccionar una estrategia de sincronización que se adapte a sus necesidades y requisitos específicos reduciendo el tiempo de su selección.

Además de la metodología también presentamos una herramienta de búsqueda de sincronización temporal a la que hemos llamado TSST. Esta herramienta está compuesta por varios bloques: el bloque de administración, el bloque de usuario y la base de datos que interrelaciona a ambos. En el bloque de administración los expertos en estrategias de sincronización parametrizan cada estrategia antes de incorporarla a la base de datos. En el bloque de usuario los usuarios en busca de una estrategia de sincronización que se adapte a sus necesidades hacen uso de un filtro parametrizado. Este filtro de selección muestra todas aquellas estrategias de sincronización que cumplen con los requisitos y necesidades que el usuario ha impuesto. La herramienta muestra los resultados en forma de tabla y de forma gráfica haciendo que la visualización y comparación de las estrategias sea más rápida.

La validación se ha hecho a través de dos casos de uso con diferente nivel de exigencia respecto a la sincronización temporal: monitorización de salud de estructuras y monitorización de parámetros ambientales. Para la monitorización de estructuras se ha llevado a cabo un estudio teórico y simulaciones con datos reales de como la sincronización temporal afecta a la identificación de las frecuencias y formas modales de las estructuras. Además, se han realizado pruebas sobre un puente en el norte de España. En la aplicación de monitorización de parámetros ambientales se ha hecho un estudio y pruebas reales del comportamiento de diferentes fuentes de reloj frente a cambios de temperatura. Esto nos ha permitido comprobar como la toma de decisiones en las primeras fases de diseño nos puede ayudar a minimizar el problema de la sincronización temporal en WSNs.

Agradecimientos

*Y tengo tantas cosas que decir
Y tú como si no fuera contigo
La historia se repite y aún así
Prometo estarte agradecido*

Rosendo

Esta es una de las primeras páginas que aparecen en el documento, pero al menos en mi caso, ha sido la última en escribirse. Supongo que no debería haberla dejado para este último momento ya que corro el riesgo de olvidarme agradecer a alguien su importancia en este largo camino.

El trabajo desarrollado en esta tesis no hubiera sido posible sin la ayuda, colaboración y apoyo de muchas personas. Cuando este trabajo dio comienzo lo que importaba no era el destino, sino el viaje. Y ahora, echando la vista atrás, toda esta aventura ha merecido la pena. El recorrido no ha sido directo de puerta a puerta, sino que ha habido muchos desvíos y sinuosos caminos, pero todo el conjunto ha hecho que este haya sido un entretenido y emocionante viaje, repleto de aventuras, experiencias y conocimiento.

Un mes antes de acabar mi estancia erasmus en Suecia me puse en contacto con Carlos López Barrio para preguntarle por oportunidades laborales. Este me puso en contacto con Octavio Nieto, y este a su vez con Jaime García. Y aquí empezó el viaje. Agradezco a Carlos ese primer voto de confianza. Pero a Octavio y a Jaime les tengo mucho más que agradecer. Además de abrirme las puertas a sus laboratorios, me han acompañado durante todo el viaje y no solo he compartido diversos proyectos con ellos, sino también largos desplazamientos en coche y avión, madrugones y anécdotas variadas. En el laboratorio de Hidráulica de Caminos conocí a Roberto Ortega, a Marcus, Christian y otros tantos más. Y el inicio de este viaje no hubiera sido igual sin ellos. Tengo mucho que agradecer a Roberto y a Amandine, su pareja. Ambos se convirtieron en muy buenos amigos con los que compartir largas charlas en torno a una mesa repleta de empanadas chilenas o tartiflette francés. Y en la segunda parte del viaje tengo que agradecer a José Manuel su disponibilidad plena siempre que he necesitado su ayuda o consejo.

Pero sin duda alguna a quien más tengo que agradecer este trabajo es a Alvaro Araujo. No solo ha sido mi director de Tesis, sino que ha sido el copiloto durante todo el trayecto. Siempre positivo y con la respuesta adecuada ante cualquier pregunta. Alvaro no solo me dio la oportunidad de realizar esta Tesis Doctoral con él, sino que me abrió las puertas a la familia del B105 y se ha convertido en director, jefe, compañero y amigo. Muchas horas de reunión han venido siempre acompañadas de comidas, BBQs y sobre todo cerveza Mahou. Muchas gracias Alvaro no solo por guiarme en este trabajo de Tesis, sino por ser ejemplo en cada una de las aventuras que hemos vivido y compartido.

Y este viaje ha sido tan agradable y divertido gracias al B105. No me imagino en ningún otro grupo realizando este mismo trabajo. Javi y Elena siempre han estado presentes de principio a fin porque ellos han sido los doctorcitos que nos iban mostrando el camino. Mariano con quien tan largas conversaciones he tenido y quien tanto me ha ayudado en la burocracia que ha sido necesaria, además de abrirnos las puertas a La Polar. Guillermo, compañero y amigo, del que a pesar de sus esfuerzos nunca he aprendido nada de estilo, aunque sí mucho de tecnología de leds y de Mass Effect. El B105 ha evolucionado durante estos años y yo me siento orgulloso de haberlo hecho con él. Y a cada uno de los compañeros tengo algo que agradecerles. Tanto a los que estuvieron al inicio del viaje como Dani, Juancar, Pedro, Elfo, Gis, y otros muchos más. Como a aquellos con quien más horas he compartido desarrollando proyectos, viajes, comidas, aventuras y cervezas: albarc, rutrilla, rrodriguezz, jmartin, anacp, sreal y los muchos rookies y no tan rookies, como Karen o Curro, que tan ameno han hecho este recorrido. Con todo este grupo las aventuras y anécdotas han sido muy diversas y siempre han estado acompañadas de comida y bebida (#gordos). Ha sido tal el buen ambiente que no solo compartimos tiempo en el laboratorio, sino que hemos hecho múltiples viajes de ocio tanto nacionales como internacionales. Sin duda alguna a esta entidad B105 le tengo mucho que agradecer, no solo por la oportunidad laboral y por acogerme para llevar a cabo esta Tesis, sino por hacerme una persona más completa. Porque *de eso se trata, de coincidir con gente que te haga ver cosas que tú no ves. Que te enseñen a mirar con otros ojos.*

Por último, me gustaría agradecer a mi familia su constante apoyo no solo en el desarrollo de esta tesis, sino en todos los pasos previos hasta llegar aquí. El esfuerzo que han hecho mis padres para que hoy yo pueda escribir estas líneas tiene un valor incalculable. Debo estarles agradecido por la educación que me han dado en casa y por darme la oportunidad de elegir mi propio futuro además de apoyarme tanto económica como moralmente a intentar cumplir todos los desafíos que surgen a lo largo de la vida.

Muchas gracias a todos por formar parte de este camino.

Table of Contents

Covers.....	i
Cover details.....	iii
How to cite	iv
 Summary.....	 v
Resumen	vii
 Table of Contents	 xi
List of Figures.....	xv
List of Tables.....	xxiii
List of Equations.....	xxv
 1. Introduction.....	 1
1.1. History of time	2
1.2. Wireless Sensor Network	3
1.3. Time Synchronization.....	7
1.3.1. Difficulties implementing time synchronization in WSN	8
1.4. Motivation.....	10
1.5. Objectives.....	12
1.6. Contributions	12
1.7. Organization of the document	13
1.8. Publications.....	13
1.8.1. Journal Articles	13
1.8.2. Conference Proceedings.....	14
1.8.3. Presentations at conferences	14
1.8.4. Posters.....	15

2. Concepts and definitions.....	17
2.1. Terminology	18
2.2. Clock sources.....	23
2.2.1. Relaxation Oscillators	24
2.2.2. Piezoelectric resonator	24
2.3. Clock model	28
2.4. Sources of delay.....	29
2.5. Conclusion.....	30
3. Related work.....	31
3.1. Traditional synchronization methods.....	31
3.2. Time synchronization strategies for WSNs	34
3.2.1. Hardware synchronization strategies.....	34
3.2.2. Software time-synchronization strategies	36
3.3. Conclusion.....	49
4. Taxonomy of time-synchronization strategies	51
4.1. Target	54
4.2. Software implementation	57
5. Developed Methodology	63
5.1. Difference between ‘method’ and ‘methodology’	63
5.1.1. Benefits of a methodology	64
5.2. Methodology for implementation of synchronization strategies for WSN	65
5.2.1. Administration view	70
5.2.2. User view.....	71
5.2.3. Parameter evaluation	73
5.2.4. Report results.....	85
5.3. Time-Synchronization Search Tool (TSST)	87
5.3.1. Database.....	88
5.3.2. Webserver.....	90
5.3.3. Administration TSST.....	90
5.3.4. User TSST.....	94
5.3.5. Display of the results.....	97
5.3.6. Associated tools.....	98

6. Validation	101
6.1. The methodology of the validation	101
6.2. Method validation	103
6.2.1. Structural Health Monitoring application.....	104
6.2.2. Environmental Monitoring application	123
6.3. Methodology validation	144
6.3.1. Structural Health Monitoring application.....	145
6.3.2. Environmental Monitoring application	148
6.3.3. User experience assessment	150
6.4. Conclusion.....	153
7. Conclusion	155
7.1. Dissertation conclusion	155
7.2. Future work	159
References	163
List of Acronyms	173

List of Figures

Figure 1.1.	The evolution and progress of wireless standards for 2.4GHz, 5GHz and 60GHz.....	4
Figure 1.2.	The number of devices connected to the internet reached 22 billion worldwide at the end of 2018, according to the latest research from Strategy Analytics. Enterprise IoT remains the leading segment, accounting for more than half of the market [10].	6
Figure 1.3.	Through the succession of nine images, we see how a group of starlings in movement creates shapes and silhouettes. This synchronization of movements in flight creates fascinating figures like this giant bird. Photographs by Daniel Biber, January 2018.	7
Figure 1.4.	Measurements made in an SHM application with nine spatially distributed channels that have taken their samples in a synchronized manner.	11
Figure 2.1.	Graphical explanation of the term's accuracy and precision. Accuracy is the proximity of measurement results to the true value. Precision refers to the closeness of two or more measurements to each other.	19
Figure 2.2.	Explanation of the term's accuracy and precision through the dispersion of samples.	19
Figure 2.3.	Phase offset (θ) between two signals in the time domain.	20
Figure 2.4.	Clock terminology: graphical behavior of the clock components over time.....	21
Figure 2.5.	Graphical representation of the parameters of a jittered signal.	21
Figure 2.6.	The simplest CMOS oscillator is created hooking an RC feedback around a CMOS logic inverter with Schmitt trigger input [28].	24
Figure 2.7.	The six delay components in the transmission path.....	29

Figure 3.1.	Diagram of how the GPS determines geographic location. A GPS unit compares the time a signal was sent by a GPS satellite (monitored by ground control stations) with the time it was received by the unit. The time difference tells how far away the satellite is. Additional distance measurements from other satellites allow the unit to determine its exact location.....	32
Figure 3.2.	A record count of publications about time synchronization in Wireless Sensor Network by year. Extracted from the analysis of the results of the Web of Science [63].	37
Figure 3.3.	Critical path analysis for traditional time synchronization protocols (left) and RBS (right) [64].	38
Figure 3.4.	Two-way message exchange between a pair of nodes in TPSN protocol [65].....	38
Figure 3.5.	FTSP decomposition of the message delivery delay over a wireless link [66].....	39
Figure 3.6.	Glossy schema showing that the protocol decouples flooding from other application tasks executing on the nodes [68].....	40
Figure 3.7.	Different approaches to classify features of clock synchronization protocols from Swain survey (2015) [24].....	46
Figure 3.8.	Taxonomy and classification of different protocols based on the implementations aspect extracted from the review of Djenouri (2016) [23].....	47
Figure 3.9.	Classification of clock synchronization protocols based on the propagation of clock parameters extracted from Geetha survey (2017) [85].....	47
Figure 3.10.	Various factors affecting packet-based synchronization extracted from Mahmood survey (2017) [86].....	49
Figure 4.1.	Time-synchronization strategies taxonomy based on target and software implementation.....	53
Figure 4.2.	Detail of the taxonomy showing only the target block.....	54

Figure 4.3.	First flexural mode of vibration of a bridge. For this mode, nodes A and B must measure the same acceleration in the same direction. The colors in the figure show the deformation of the structure, that is, the red and orange colors are where more acceleration has been detected and the blue colors where less acceleration has been detected.....	55
Figure 4.4.	Beamforming scenario between wireless nodes and the far-away base station showing the system model for carrier synchronization. Extracted from work [89].....	56
Figure 4.5.	Detail of the taxonomy showing only the security block.....	57
Figure 4.6.	Detail of the taxonomy showing only the energy efficiency block.	58
Figure 4.7.	Detail of the taxonomy showing only the complexity block.....	58
Figure 4.8.	Detail of the taxonomy showing only the hardware requirements block.....	59
Figure 4.9.	Detail of the taxonomy showing only the clock correction block.	59
Figure 4.10.	Detail of the taxonomy showing only the software abstraction block.	60
Figure 4.11.	Diagram of the software abstraction layers in WSN and the associated communication paths delays.....	61
Figure 5.1.	Flowchart of the proposed methodology, where the input blocks are green, and the output blocks are blue. White blocks are where the methodology makes decisions.....	67
Figure 5.2.	Graphical representation of the insertion of a new time-synchronization strategy into the tool database with all the possible solutions. This example shows the insertion of solution 4.....	69
Figure 5.3.	Graphic representation of a very demanding time-synchronization strategy (left) and therefore with a small area and a very relaxed time-synchronization strategy (right) and therefore with a much larger area.	71

Figure 5.4.	Flowchart of the user tool where all the input blocks are evaluated and queries to the time-synchronization strategies database are generated. The cycle ends when there is no solution available to fulfill the requirements or when all the input blocks are evaluated, and the radar chart is generated.	72
Figure 5.5.	Graphic representation of the application requirements creating a user filter.	73
Figure 5.6.	Graphical representation of how a valid solution is selected. Here the application requirements shape (blue area) is superimposed over all the available solutions in the database. Only the solutions 3 and 4 are valid (grey area) because they are fully contained in the user filter area. The solutions 1 and 2 are with a pointed line and are discarded.	86
Figure 5.7.	Index page of the Time-Synchronization Search Tool (TSST).....	88
Figure 5.8.	CockroachDB database log after initializing	89
Figure 5.9.	Display of the first section of the aTSST where they ask about publication details of the synchronization strategy.	91
Figure 5.10.	Display of the second section of the aTSST where the target is selected with checkboxes and the value of the parameters through sliders. The user interaction buttons are also shown.	92
Figure 5.11.	Display of the third section of the aTSST where when interacting with the buttons a table with the saved strategies is shown. Clicking on one of the strategies in the table displays its associated radar chart showing the value of its parameters.	93
Figure 5.12.	The user interface of the uTSST	95
Figure 5.13.	The output result of user TSST where strategies are shown as colored lines that comply with the filter represented in a semi-transparent blue fill.	96
Figure 5.14.	The output result of user TSST where one strategy is highlighted and four are disabled.	97
Figure 5.15.	Initialization pseudo-code for time synchronization of the secure time synchronization protocol against fake timestamps for Large-Scale IoT [79]	97

Figure 5.16.	A calculator that converts the quality of a clock expressed in ppm into time deviations and can also add a deviation term according to a temperature coefficient.....	99
Figure 5.17.	A calculator that converts between multiple units: decimal values, percentage, per miles, ppm, ppb, and ppt.	99
Figure 5.18.	Development that dynamically calculates the compromise between the maximum allowed synchronization error in an application, the quality of a clock source and the intervals between synchronization rounds.....	100
Figure 6.1.	Flow diagram showing the analysis structure of each use case.	104
Figure 6.2.	Block diagram of the developed hardware platform (left) and a simplified diagram of a real measurement scenario on a road bridge (right).	106
Figure 6.3.	From left to right and from top to bottom. a) Image of the tests made on a road in a bridge in Spain. b) Image of one of the clients used to record the vibrations of the road over the bridge. c) One of the accelerometers used to record the accelerations. d) Server with a wireless module used to send commands and synchronization pulses to the clients.	107
Figure 6.4.	Radar chart generated based on the input form for a specific Structural Health Monitoring application. The most demanding parameter is precision with a value of 4 and the most relaxed is the security with a value of 10.....	111
Figure 6.5.	User selection tool showing a table with all the viable time-synchronization that fulfills the user filter and a graphical representation with a radar chart.....	111
Figure 6.6.	Example of first four mode shapes of a Bernoulli beam with five measurement nodes.....	113
Figure 6.7.	Influence of phase deviation between different measuring nodes	114
Figure 6.8.	Comparison of mode shape 9 (red line) with 7 (blue line) and 27 (green line) measurement points.....	116
Figure 6.9.	Diagram of the monitored road bridge with the position of the 34 measurement points.	117

Figure 6.10. Identified mode 1 (left) and mode 5 (right) of the road bridge after 10 min measurement.	118
Figure 6.11. Identified mode 1 (left) and mode 5 (right) of the road bridge after 15 min measurement.	118
Figure 6.12. The eight identified mode shapes of a Bernoulli beam with seven measurement points with the theoretical modal shape (dashed red line), the modal identified theoretical shape (blue line) and the modal identified shape when synchronization error of 1ms is introduced at nodes 3 and 5 (green line).	120
Figure 6.13. Superposition of the results obtained with the wired system (black line) and the wireless system developed, taking measurements of 10 (red line) and 15 (blue line) minutes.	122
Figure 6.14. Modular stackable hardware platform. The initial basic system composed of two layers: power management, and control and communications (left). Detail of different clock sources (right): a) external 32.768 kHz, b) external 8 MHz for PLL, c) external DS3231 TCXO.	124
Figure 6.15. Schematic diagram of the board containing the four parallel voltage regulators.	125
Figure 6.16. Schematic diagram of the control, storage, and communications board.	125
Figure 6.17. The radar chart generated for the specific requirements of an environmental monitoring application. Big values represent few constraints or big freedom in the selection, and low values represent high demands or very restrictive conditions.	128
Figure 6.18. User selection tool showing a table with all the viable time-synchronization that fulfills the user filter and a graphical representation with a radar chart.	129
Figure 6.19: Example figure of a DS3231 TCXO where the behavior of a typical uncompensated crystal oscillator and a compensated oscillator can be seen. The figure shows how an oscillator slows down at cold and hot temperatures [119].	132

Figure 6.20. The absolute maximum of the theoretical value of the deviation of an oscillator with different accuracy (yellow bars) and the influence of constant temperature ($\pm 10^{\circ}\text{C}$ red bars, $\pm 20^{\circ}\text{C}$ blue bars) with a 1,000-ppm oscillator (left) and with a 2-ppm oscillator (right).....	133
Figure 6.21: Graphical representation of the WSN scenario	135
Figure 6.22: Long-term experiment with non-calibrated internal CMOS oscillators. Two nodes are at room temperature (yellow and green lines) and the other two nodes alternate between the room and cabin temperature (cyan and magenta lines).	137
Figure 6.23. The accumulated drift of non-calibrated internal CMOS oscillators versus temperature: room temperature in (left) and drastic temperature changes in (right). The accumulated drift follows an ascending line when the temperature falls and a downline when the temperature grows.	138
Figure 6.24: Long-term experiment with calibrated internal CMOS oscillators. Two nodes are at room temperature (yellow and green lines) and the other two nodes alternate between the room and cabin temperature (cyan and magenta lines). A pointed line represents the linear fitting behavior of each node's clock.	139
Figure 6.25. The accumulated drift of calibrated internal CMOS oscillators versus temperature: room temperature in (a) and drastic temperature changes in (b). The skew grows linearly but the drift does not, and it directly depends on temperature changes.	140
Figure 6.26. Long-term experiments with external 32.768 kHz oscillators. In Figure (left) it can be seen how the accuracy of the external oscillators is better than that of the LSI by several orders of magnitude. In Figure (right) the same figure is plotted without the LSI line.	140
Figure 6.27: Stability of an XO and a TCXO under drastic temperature changes.	141

Figure 6.28. HSE drift behavior under two different temperature environments: room temperature (left) and drastic temperature changes (right). The influence of temperature changes can be considered negligible for the HSE drift.....	142
Figure 6.29. The behavior of an external 16 MHz oscillator in an ATmega328P device under two different temperature environments: room temperature (a) and drastic temperature changes (b). The cabin environment with drastic changes in temperature has more influence in the oscillators drift.	142
Figure 6.29. Level of expertise in WSNs (top), and degree of knowledge in the field of time synchronization in WSNs (bottom), being 1 the lowest degree of knowledge and being 5 the expert level.	144
Figure 6.30. Values assigned by test users to all parameters in the case of the SHM application. All parameters except the target are represented in histogram graphs.....	146
Figure 6.31. Results obtained by two users who have applied the methodology and made use of the uTSST to choose a suitable synchronization protocol for an SHM application. On the left are the results of a non-expert user and on the right of an expert user.	147
Figure 6.32. Values assigned by test users to all parameters in the case of the Environmental Monitoring application. All parameters except the target are represented in histogram mode.	149
Figure 6.33. Results of the filters created for an environmental monitoring application. The figure shows the values assigned to each parameter by a non-expert user (left) and an expert user (right).	150
Figure 6.34. Histogram of users' answers when asked their opinion about the methodology developed.....	151
Figure 6.35. Histogram of users' answers when asked their opinion about the developed Time Synchronization Search Tool (TSST).....	152

List of Tables

Table 2.1.	Comparison of oscillators accuracy.....	28
Table 3.1.	A comparison chart of various time sync schemes and metrics extracted from Ranganathan survey (2010) [7].	45
Table 3.2.	Global objective features of representative clock synchronization protocols from the Swain survey (2015) [24].	45
Table 3.3.	Comparison between different clock synchronization parameters extracted from Geetha survey (2017) [85].....	48
Table 5.1.	Correspondence between the blocks to be followed by the methodology and the parameters that are evaluated in the radar chart.....	74
Table 5.2.	Values associated with the level scale for the power consumption parameter ranging from 1 to 9.	74
Table 5.3.	Values associated with the level scale for the monetary cost parameter ranging from 1 to 9.	76
Table 5.4.	Values associated with the level scale for the security parameter ranging from 1 to 9.....	77
Table 5.5.	Types of network topologies associated with the level scale for the network topology parameter which has five options.	78
Table 5.6.	Message distribution types associated with the level scale for the message distribution parameter which has five options.	79
Table 5.7.	Values associated with the level scale for the accuracy and precision parameter ranging from 1 to 9.	80
Table 5.8.	Values associated with the level scale for the clock stability parameter ranging from 1 to 9.	81
Table 5.9.	Software abstraction levels associated with the level scale for the software abstraction parameter that has eight options.	82
Table 5.10.	Hardware abstraction levels associated with the level scale for the hardware abstraction parameter which has seven options.	84
Table 5.11.	Summary of the values and levels associated with each of the parameters that form the radar chart. Through this table, the relative evaluation of each administrator and user is translated into an absolute evaluation, the same for all.....	85

Table 6.1.	Summary of the values assigned to the nine parameters of the methodology for a specific application of Structural Health Monitoring	110
Table 6.2.	Theoretical modes and errors associated with 1000 Hz of sampling frequency. For each of the modal frequencies (f_n) associated to the modal order (n), the values of phase deviation (φ), amplitude error (Amp_{error}), and MAC values are obtained with 7 (MAC_{7n}) and 27 (MAC_{27n}) measurement points are shown.	116
Table 6.3.	Frequency comparison of series where the generated theoretical frequencies (f_{theo}), the identified frequencies from the signal without error (f_{wout}) and the identified frequencies from the signal with the induced offset (f_{with}). The lack of synchronization does not affect the identification of the natural frequencies of the structure.	119
Table 6.4.	Normalized amplitude of each of the seven measurement points (N_{dx}) between the modal identified theoretical shape (ϕ_{wout}) and the modal identified shape when synchronization error of 1ms is introduced (ϕ_{with}).....	121
Table 6.5.	Summary of the values assigned to the nine parameters of the methodology for a specific environmental monitoring application.	127
Table 6.6.	The relation between oscillator quality and maximum synchronization error allowed.....	135
Table 6.7.	Measured accuracy of LSI clocks: the LSI clock factory deviation from an ideal 32 kHz without calibration.	136

List of Equations

Equation 2.1	28
Equation 2.2	28
Equation 2.3	29
Equation 6.1	132
Equation 6.2	134

Chapter 1

Introduction

Content

1.1. History of time	2
1.2. Wireless Sensor Network	3
1.3. Time Synchronization.....	7
1.4. Motivation.....	10
1.5. Objectives.....	12
1.6. Contributions	12
1.7. Organization of the document	13
1.8. Publications	13

Lo importante es no dejar de hacerse preguntas

Albert Einstein

Throughout this chapter, the reader is shown the importance of time in our daily lives. The concept of Wireless Sensor Networks (WSNs) is introduced and how these are being expanded to perform many tasks previously done with wired networks. The importance of Time Synchronization (TS) in WSNs is highlighted, and the difficulties involved in synchronizing a network of distributed wireless nodes. This chapter presents the motivation that has led us to develop this doctoral thesis, the objectives that we intend to achieve and the publications that have been carried out throughout the development of the work.

1.1. History of time

Humanity has experienced the need to measure time from the beginning of civilization. The notion of time is fundamental to our existence and very familiar in our lives. The measurement of time has evolved according to the era thanks to the technology used, making these measurements increasingly accurate and with better performance. In prehistory, time was measured using natural phenomena, such as the movement of stars, and calendars were soon created based on the sowing and harvesting periods from the Sumerian civilization (2400 BC).

The concept of the clock appeared for measuring time. A clock is a device that counts events indicating the passage of a time interval (e.g., 1 second) and uniquely labels each of those intervals (e.g., 3rd Jan 1982 23:54:37). Their quality depends on the stability of their frequency, and therefore on the ability to generate events at a constant frequency [1]. The absolute value of the frequency compared to the desired value, or its frequency accuracy, is important, but calibration can compensate for an inaccurate but stable standard. The first clocks on record were the sundial and the water clock from the Egyptian civilization (1500 BC). The hourglass was invented in the 13th century AD and mechanical clocks appeared some centuries later and. Then, thanks to C. Huygens and his study of the pendulum, the pendulum clock (1657) emerged, where the minute hand was added, gaining accuracy and reliability. The invention of the stopwatch followed in 1735. The electronic clock (1841) was introduced by the Scotsman A. Bain using electrical energy that moved a pendulum with electromagnets. In 1920 the quartz clock, based on piezoelectricity, came on the scene thanks to W. A. Morrison. Finally, the atomic clock appeared in 1948. Today, the most accurate timepiece is an atomic clock, the NIST-F1, developed by the National Office for Standardization (NIST) of the United States, launched in 1999. It is so accurate that it has a margin of error of only one second every 30 million years [2]. The most common unit for time is seconds and, since 1967, the second has been defined as exactly “the duration of 9,192,631,770 periods of the radiation corresponding to the transition between the two hyperfine levels of the ground state of the 133 Cesium atom” (at a temperature of 0 K) [3].

In electrical engineering, the use of the word “clock” differs from the common use of that term, which refers to a device that keeps track of the hours of the day. Reasonable quality mechanical watches can be expected to keep time to within fifteen seconds per day. Standard quartz watches can be expected to keep time to within fifteen seconds per month. Thermo-compensated quartz watches take performance to the next level. Current models are specified by their manufacturers to keep time to within fifteen seconds per year or better and usually perform much better. To put this performance in perspective, fifteen seconds per year is five hundred parts per billion.

A clock signal is a binary signal in digital electronics that serves to coordinate the actions of several circuits, especially for the synchronization of the flip-flops in complex digital systems. The use of the clock in electronics opened the door to synchronous systems. In asynchronous sequential circuits, the changes of states occur at the natural rhythm associated with the logic gates used in their implementation.

The synchronous sequential circuits only allow a change of state in the instants marked or authorized by a signal of synchronism of oscillatory type (i.e., the clock), which solves the problems that the asynchronous circuits originated by not having uniform changes of state within the system or circuit. In cases where there is a clock signal, it is usually given by a clock generator. The signal oscillates between a high and low status. This signal is characterized by a period of oscillation, the clock frequency, and the duty cycle. The use of the clock in an electronic system causes a machine-coded instruction to be executed at each clock cycle. Most complex integrated circuits require a clock signal to synchronize different parts of the chip and to balance the gate delays. As chips become more complex, it becomes increasingly difficult to provide an accurate and consistent clock signal everywhere. Today, nearly all microcontrollers and microprocessors depend on some kind of clock to execute all their instructions. The higher the working frequency of the clock, the faster it executes instructions. In addition, the use of the clock in electronic systems has allowed the use of timestamps, mechanism that allows the system to associate a timestamp to a certain event and many time-synchronization strategies have been created using this mechanism.

Electronic systems are synchronized to perform tasks or events simultaneously, or near-simultaneously, from a certain perspective. However, in his first relativity paper in 1905, Albert Einstein proved that there are no such things as absolutely simultaneous events. According to Einstein's special theory of relativity, it is impossible to say in an absolute sense that two distinct events occur at the same time if those events are separated in space [4]. Leaving aside the precise definition of Einstein's absolute simultaneity, in this document, when we talk about synchronized events, or events that are performed simultaneously, we use the traditional definition: events that happen at the same time.

This brief review of the history and evolution of the clock reflects the importance of the measurement of time throughout history. This importance is still evident today in all facets of our lives.

1.2. Wireless Sensor Network

A Wireless Network (WN) is any type of network of devices that use wireless technology to establish communication connections to exchange information between the different nodes that make up their network. The desire for connectivity and the need to avoid, for example, the costly and tedious process of deploying cables, has resulted in wireless networks becoming a method of interconnection and intercommunication between devices in constant growth, increasingly integrated into professional and daily life.

Typical examples of wireless networks that inundate our daily environment include such networks as the Wireless Local Area Network (WLAN) (Wi-Fi at homes), telephone networks, terrestrial microwave networks, such as those used for television and radio distribution, and many more.

The exchange of data between wireless devices has grown considerably over the last few years. A clear example of such growth is the WLANs, as can be seen in Figure 1.1. In 1997, the standard WLAN, IEEE 802.11 [5], had a data rate of 2 Mbps. The use of the new standard 802.11ac, also called Wi-Fi 5 or Wi-Fi Gigabit, can now achieve a throughput of about 433 Mbps with a single link, theoretically achieving rates of 1.3 Gbit/s using 3 antennas. Even the 802.11ax, also named Wi-Fi 6 or Wi-Fi 6th generation, improves the global spectral efficiency in the bands of 2.4 GHz and 5 GHz. This means that the data rate has multiplied by 25,000 in around 20 years.

Figure 1.1. The evolution and progress of wireless standards for 2.4GHz, 5GHz and 60GHz.

Another example of WNs is Wireless Personal Area Networks (WPAN), defined as networks with coverage of fewer than 10 meters radius and centered on personal devices without a fixed infrastructure. An example of WPAN is Bluetooth, which was later standardized as IEEE 802.15.1 [6]. Nowadays, all devices such as mobile phones, tablets, audio systems for vehicles or homes incorporate a Bluetooth system for interconnection between devices. There are many more wireless applications and many have low data rate requirements and few bits a day are needed.

Wireless Sensor Networks (WSNs) are a subset of WNs. These are characterized by autonomous sensors that are spatially distributed to monitor physical or environmental conditions and that share their data through the network. A WSN node consists of four main components: the processing unit, the communication unit, the sensing unit, and the power unit.

Generally, their objective is to collect information to be pieced together in order to assemble a broader picture of the environment that each sensor would have individually. One of the characteristics of a WSN is that the aggregated global information of its different sensors becomes greater than the sum of its parts. In most cases, the primary function of a WSN is to sense the environment and transmit that information to a base station for deeper processing with secure time information [7].

The devices in a WSN form a network without the need for any infrastructure. In some cases, network nodes cooperate for communication by forwarding each other's packets for delivery from a source to a destination. This yields a multi-hop communication environment. Today's WSNs have proved to be a success in certain applications where traditional measurement systems were not, and they have even made new solutions, that were not viable before, possible. The deployment of WSNs offers developers the ability to adapt the measurement system to the needs of the application. A WSN can observe the environment in a very different way from

previous systems. Information can be extracted in a way that is closer to the phenomenon's occurrence, making use of wide and dense areas in both space and time [8]. WSNs must usually operate unattended for long periods (months or years) and remain reliable, which is one of the reasons, among others, why low consumption of the nodes is sought. As WSNs have limited power resources, computing capabilities, storage capacity, and bandwidth, solving the problems they face becomes more complex due to the trade-off between the parameters involved; for example, the development of energy-saving techniques that are used to reduce consumption across the network.

WSNs are characterized by their ability to deal with a combination of technical difficulties that are not found in other types of networks. These difficulties are detailed below [9]:

- *Low Power Consumption.* The average power consumption in a WSN is usually very low and with limited peak power sourcing capabilities.
- *Low-Cost.* WSNs are usually networks with low resource nodes. One of the objectives of the engineer is to design low-cost nodes to make networks with many nodes possible.
- *Worldwide Availability.* WSNs should meet the varying worldwide requirements and employ a single band worldwide, usually a free radio band, such as Industrial, Scientific, or Medical (ISM) radio bands.
- *Network type.* Different topologies can be used in a WSN, such as star, mesh, cluster, tree, or others.
- *Security.* How secure the network is and how secure the network is perceived. WSNs are probably going to replace a wired version of the system, so the use of encryption ensures the integrity of the transmitted messages.
- *Data throughput.* WSNs usually have limited data throughput requirements. One of the reasons is not to waste other resources, such as power energy, in transmissions.
- *Message Latency.* Depending on the application, the message latency requirement in WSNs can be very relaxed.
- *Mobility.* In general, WSNs do not require mobility, but the nodes are usually free to move in a coverage area.
- *Communication failures.* WSNs can be aware of communication failures and have the ability to withstand harsh environmental conditions.
- *Scalability.* WSNs should be able to scale the network deployment without degrading the performance of the whole system.
- *Autonomous.* The network should be a WSN capable of self-configuration and self-maintenance.

Figure 1.2. The number of devices connected to the internet reached 22 billion worldwide at the end of 2018, according to the latest research from Strategy Analytics. Enterprise IoT remains the leading segment, accounting for more than half of the market [10].

In recent years, another group of wireless devices called the Internet of Things (IoT) has appeared. It could be said that there are few differences between WSN and IoT, and perhaps the most relevant and striking is the Internet connection. WSNs do not need to have an internet connection because they are final sensor nodes and coordinating nodes talking to each other. A WSN can be transformed into an IoT network if we provide it with an internet connection and if we consider that both are based on monitoring and recording the physical conditions of the environment. The IoT, as well as the WSN, is the integration of connectable devices and sensors to enable remote monitoring, creating a network of networks of autonomous objects. The wireless IoT family is expanding to different areas, such as vehicles, buildings, construction equipment, smart wearables, smart home automation, smart health monitoring, smart cities, smart security, etc. Every day, more applications and wireless devices are available to interconnect and solve multiple and diverse problems. According to the Gartner Inc. 2017 prediction, the IoT will include 20.4 billion units installed by 2020 [11] with almost \$3 trillion of investment. The numbers vary from source to source, but all of them agree that the amount of IoT connected devices installed worldwide will continue to increase [12] [13] and that, in the near future, IoT devices are expected to be integrated into a larger number of areas (home, industrial, medical, etc.) and applications (smart security solutions, smart home automation, smart health care, smart wearables, transportation, building automation, renewable energy, etc.). The IoT is the cornerstone of many organizations' digital transformation, enabling them to optimize existing operations and excel at creating and pursuing exciting new business models [14]. The traditional fields of embedded systems and WSNs, among others, all contribute to enabling the IoT. Figure 1.2

shows graphically the research conducted by Strategy Analytics in May 2019 [10], where the millions of devices connected to the network can be seen separated into groups. The enterprise IoT remains the leading segment, accounting for more than half of the market.

1.3. Time Synchronization

The flexibility offered by WSNs keeps growing rapidly, and this has meant that they have been considered a fundamental structure for future ubiquitous communications due to a variety of promising potential applications. Some of these applications need time synchronization.

Figure 1.3. Through the succession of nine images, we see how a group of starlings in movement creates shapes and silhouettes. This synchronization of movements in flight creates fascinating figures like this giant bird. Photographs by Daniel Biber, January 2018.

For a WSN to work properly, it is necessary for the different elements that make it up to be coordinated. In particular, those elements necessary for wireless communications and all those actions that must be executed simultaneously in the different devices for the correct operation of the application. If these actions are synchronized properly, communication collisions are minimized, thus minimizing retransmissions and helping to save energy. Similarly, if the acquisition of data by the sensors is synchronized, we would have a true picture of the monitored area. An example of synchronization without collisions can be seen in nature when a group of starlings flies and their movement create shapes and silhouettes. This synchronization of movements in flight creates fascinating figures like the giant bird that can be seen in Figure 1.3.

Due to the usefulness of WSNs in multiple fields, the applications based on wireless sensor end devices have diversified and expanded. Nowadays, it is possible to find wireless network applications dedicated to monitoring parameters in a distributed way [15]. Some wireless networks coordinate sleep wake-up node scheduling mechanisms [16]. There are also some wireless network applications with communication protocols based on time slots to transmit and receive messages so as to make efficient use of the radio spectrum [17]. All of them need time-synchronization strategies to operate correctly. Each of these applications has different needs concerning the time synchronization of the network. Thus, many strategies have appeared over the years to synchronize WSNs and continue to do so, adapting the solutions to a specific problem.

Because of this wide variety of WSN applications, time-synchronization requirements in WSNs are very different, going from very lax to very demanding ones. Every time developers face the problem of choosing and implementing a time-synchronization strategy for a specific WSN application, they realize that an enormous amount of research is necessary before undertaking the task. Solving the time-synchronization challenge must be done considering many other trade-offs, because time synchronization is just another service in the WSN application routine. There is no single time-synchronization strategy that maximizes all parameters, nor is there a time-synchronization strategy that works for every application. For example, in most cases, the more precise the time-synchronization strategy is, the more message exchanges between nodes are needed, or the better and more expensive hardware it requires [18]. Strategies can pursue an accurate time-synchronization in top layers, such as the application layer [19], or low layers, such as the Medium Access Control (MAC) layer [20]. Time synchronization does not mean that all clocks are always perfectly synchronized on the network. That would be the strictest and most complicated way to implement. Precise synchronization is not always essential. Protocols that go from indulgent to strict are available to meet the user's needs. Depending on the application, various degrees of accuracy are required.

In centralized systems, there is no need for synchronized time because there is no time ambiguity. A process, task or event gets the time by simply issuing a system call. When another event then tries to get the time, it will either get an equal or higher time value. Thus, there is a clear ordering of events and times at which these events occur. In distributed systems, there is no global clock or an implicit notion of common time. Each node in the network has its own internal clock and its own notion of time. Also, because different clocks tick at different rates, they may not always remain synchronized, although they might be synchronized at any time, for example, when they start.

1.3.1. Difficulties implementing time synchronization in WSN

After this justification of why time synchronization is not only important but vital in sensor networks, we would like to highlight the complexity and difficulty of achieving it in the WSN field.

Synchronizing a WSN is not a trivial task. Clocks drift out of synchronization, especially those using low-cost, commodity computing parts that are often used in low power WSNs. Since each of the devices in the network has its own clock, a synchronization strategy is needed to make them all work in unison. The most common strategy consists in synchronizing all the devices' clocks with a reference clock. In most cases, it is not enough to synchronize local clocks once at the beginning of the application, because local clocks in the wireless devices are not usually stable enough to keep the time synchronization in the network during the lifetime of the application. In most cases, more advanced synchronization strategies are needed to keep the application properly synchronized during all its operations. We must also consider that the synchronization procedure is not the primary task of a wireless sensor node; it is usually a service routine that allows us to improve (or warrant) the reliability of the main task [21].

We have independent nodes, that is, each of them has its own modules: battery, oscillators, main microcontroller, radio transceiver, sensors, etc. Also, they are distributed by an area, and therefore, the atmospheric conditions they face may be different one from the other. Hardware clocks that use both the microcontroller and the transceivers are not perfect, and therefore there are variations in the oscillator. Some nodes can be at ground level, others at 3 meters in height. Some may be in free space, others may see their communications attenuated by trees, walls, etc. Some can be in the sun and therefore with high temperatures, while others can be in the shade. Some can be maintained under stable atmospheric conditions, while others can face intense atmospheric changes. The instructions of the MicroController Unit (MCU) necessary to execute certain actions are not always the same and will therefore not always have the same duration. Imagine the case of interruption by the reception of a package. Time synchronization in many cases goes unnoticed because, either the problem is not critical or it has been solved by a third party without us realizing it, for example, when the synchronization protocol is inherent in the dissemination of network messages. The non-determinism in the dynamics of the network complicates the task of synchronizing the systems. When a node timestamps a packet to send it to another node, the packet that transports the timestamp faces variable delays until it reaches the destination and is decoded. These delays do not allow the recipient to compare the clocks accurately and therefore do not allow precise synchronization.

Since WSNs are limited in energy resources, computing capabilities, storage capacity, and bandwidth, the implementation of traditional algorithms for time synchronization, such as Network Time Protocol (NTP) and Global Positioning System (GPS), is not practical in WSN. Researchers have developed successful synchronization protocols for wired networks in recent decades. But these are unsuitable for WSNs because of the challenges posed by the special characteristics of this type of networks.

Many factors, such as temperature, oscillator aging, battery charge, pressure, etc., all affect time synchronization, causing dynamic deviations over time [22]. For example, temperature changes affect the accuracy of local clocks by dynamically modifying their frequency and causing them to deviate from the reference. Other factors that affect an accurate and stable time synchronization are the number of

nodes in the network, the topology, the type of message distribution, software layers with non-deterministic delays in communication processes, attacks from malicious time sources, etc. [23].

Clock error (deviation) is inevitable. Clock values of different wireless sensor end devices diverge from each other and, consequently, many clock synchronization protocols for wireless networks have been proposed in the past. These protocols differ from each other considerably due to their structural, technical or global objective features [24]. To synchronize a wireless network, it is necessary to reach a common notion of time, taking into account the fact that the clocks of the devices can deviate due to multiple reasons, that the time requirements for every application vary, and that the target parameters to maximize in the application differ in every case. For example, there are one-hop vs. multi-hop wireless networks, internal vs. external clock synchronization, stationary vs. mobile nodes, fault tolerance, scalability, sleep scheduling, energy efficiency, etc. The approach of every clock synchronization protocol is different due to the limited resources of every network, the nature of the communication, or the physical environment. To develop an appropriate time-synchronization strategy, several things need to be known: the source from where the clock value is propagated, how the clock value is propagated, when the clock value is propagated, how the logical clock time is determined from the physical clock value, whether the time synchronization is a priori or post-facto, or whether it is probabilistic or deterministic, among other concerns.

In short, time synchronization in WSNs is not an easy task. This is because there are many different scenarios and applications out there, and each of those scenarios and applications may require different hardware and may have different requirements and needs for time synchronization. Given the need to have a synchronization system for any WSN, and given the difficulty of finding a suitable system for each need, this work faces the problem and proposes a solution.

1.4. Motivation

Several years ago, we were developing a project for the monitoring of civil structures. The objective was to develop a system that would collect the vibrations of a structure. Initially, the platform focused on the development of a high-speed acquisition system with 24-bit resolution. Additionally, to improve the system, we were using a model of accelerometers with very high sensitivity and each of the implemented nodes had wireless communications to be able to locate them in the optimal places for measurement and data collection.

After the first field tests, we realized that, although we were using good quality accelerometers, high speed and high-resolution Analog to Digital Converter (ADC) modules, and that the system acquired the data correctly; when we merged the data from different nodes, the results were not as expected. This led us to discover the importance of time synchronization in these applications when using WSNs. The problem had gone unnoticed when the measurements were made with wired systems since, in this type of system, there are no synchronization problems between the

different measurement points. Without time synchronization, all our efforts in the acquisition were useless since, when merging the data, a small deviation between the samples taken of several nodes meant that the results were not the correct ones, thus giving us a distorted vision of the reality.

When we decided to integrate a synchronization strategy, we realized that there were different options available. Each of these options had their pros and cons. We had to analyze each of them in-depth to understand them and see which was best suited to our needs at the time. This need to find a time-synchronization strategy that would adapt to the needs of the application is what motivated us to carry out this Doctoral Thesis.

During the development of this thesis, we abstracted from the sensors part and focused on those units that have a direct relationship to the system clock. To do this, we analyze both the hardware and the software parts involved.

At the end of the structure monitoring project, we already had a valid synchronization protocol integrated into the measurement system. Figure 1.4 shows measurements made in a Structural Health Monitoring (SHM) application with nine spatially distributed channels that have taken their samples in a synchronized manner.

Figure 1.4. Measurements made in an SHM application with nine spatially distributed channels that have taken their samples in a synchronized manner.

In short, time synchronization is a service, not only necessary, but in many cases vital in WSN applications. However, today, the choice of a time-synchronization strategy that adapts to the specific needs of each case requires detailed knowledge of the subject. For these reasons, it has been considered necessary to develop a methodology for the implementation of time-synchronization strategies in WSNs. We consider that this methodology facilitates and speeds up the choice of time-synchronization strategies for all developers of WSN applications, regardless of their degree of knowledge in the field of time synchronization.

1.5. Objectives

This thesis has a general objective: the development of a methodology for the implementation of time-synchronization strategies for wireless sensor networks. The methodology presented in this dissertation offers WSN developers a way to choose a time-synchronization strategy methodically. The proposed methodology provides developers, looking for a time-synchronization strategy for their WSN, with a set of strategies that fit the needs of each case. Thus, developers do not need to be experts in the specific issue of time synchronization to choose an adequate strategy. Each time-synchronization strategy has previously been evaluated by an expert in the field and has been cataloged based on multiple parameters. The methodology compensates for the lack of experience that developers may have by adapting the level of abstraction to the experience of the developer. So the methodology reduces development time by offering solutions graphically pre-evaluated by experts. These features ensure a successful decision for everyone.

To achieve this general objective, we have set other specific and shorter objectives.

- Analysis of the state of the art in the field of time synchronization in WSNs.
- Analysis of the clock sources applied in the nodes of the WSN and their influence in time synchronization.
- Development of a hardware platform that allows us to evaluate the implementation of time-synchronization strategies.
- Implementation of a synchronization strategy in a real application, e.g., structural health monitoring.
- Analysis of the influence of time synchronization in a real application, e.g., structural health monitoring.
- Establish a framework for comparing new and existing time-synchronization strategies.
- Development of a taxonomy for the cataloging of time-synchronization strategies.
- Development of a tool that facilitates the use and application of the previously developed methodology.
- Validation of the methodology and tool previously developed.

1.6. Contributions

Due to the complexity and challenges involved in choosing the time-synchronization strategy in each case, a methodology for choosing time-synchronization strategies for WSNs is presented in this dissertation. Developers can focus their efforts on the application and main services because the methodology abstracts them from the complexity of the time-synchronization strategies, facilitating their choice and assuring a strategy adapted to the WSN needs. The methodology helps experts and developers who lack the experience in time synchronization to find the appropriate solution for specific cases. In most cases, the

methodology speeds up decision-making when choosing an appropriate time-synchronization strategy for an application, even when the person who must choose it is an expert in the field.

An online tool has been developed to put the application of the methodology into practice. This tool allows any WSN application developer to obtain a set of time-synchronization strategies as output by using some entry requirements. The database that has been developed as part of the tool also becomes a reference point for experts and non-experts in time synchronization. The database is where any time-synchronization strategy is analyzed and parameterized before being inserted and, therefore, at the time of its consultation, allows the evaluation of the protocols from a standardized point of view.

In addition, to implement different synchronization strategies and to be able to apply these strategies in real environments, several hardware platforms have also been developed.

1.7. Organization of the document

The document is organized as follows. This first introductory chapter is followed by Chapter 2 of general concepts and definitions, necessary to understand the rest of the document. Chapter 3 reviews the state of the art in synchronization strategies. Chapter 4 presents the development of a taxonomy for cataloging synchronization strategies. The methodology is developed and explained in Chapter 5. This same chapter also presents the online tool that allows us to execute the methodology in a guided manner. Chapter 6 explains the validation procedures for both the methodology and the tool. The same chapter shows the results of the analysis of the clock sources and the influence of time synchronization on the application of structural health monitoring. Finally, Chapter 7 highlights the most relevant conclusions of this doctoral thesis. At the end of the document, the reader can find the sections of References and the List of Acronyms.

1.8. Publications

1.8.1. Journal Articles

- F. Tirado-Andrés, A. Rozas, and A. Araujo, “A Methodology for Choosing Time Synchronization Strategies for Wireless IoT Networks,” *Sensors*, vol. 19, iss. 16, 2019.
- F. Tirado-Andrés and A. Araujo, “Performance of clock sources and their influence on time synchronization in wireless sensor networks,” *International Journal of Distributed Sensor Networks*, vol. 15, iss. 9, 2019.
- Araujo, J. García-Palacios, J. Blesa, F. Tirado, E. Romero, A. Samartín, and O. Nieto-Taladriz, “Wireless Measurement System for Structural Health Monitoring with High Time-Synchronization Accuracy,” *IEEE Transactions on Instrumentation and Measurement*, vol. 61, iss. 3, pp. 801-810, 2012.

1.8.2. Conference Proceedings

- J. H. García-Palacios, I. M. Díaz, J. C. Mosquera, J. M. Soria, and F. Tirado-Andres, “Learning dynamic analysis of structures using handy and affordable equipment. On the way of smart structures,” in *IV International Conference on Structural Engineering. Education Without Borders*, 2018, pp. 134-142.
- R. Rodriguez-Zurrunero, F. Tirado-Andres, and A. Araujo, “YetiOS: an Adaptive Operating System for Wireless Sensor Networks,” in *2018 IEEE 43rd Conference on Local Computer Networks Workshops (LCN Workshops)*, 2018, pp. 16-22.
- J. García-Palacios, J. M. Soria, I. M. Díaz, and F. Tirado-Andres, “Modal tracking with only a few of sensors: application to a residential building,” in *8th European Workshop on Structural Health Monitoring (EWSHM 2016)*, 2016.
- J. García-Palacios, J. M. Soria, I. M. Díaz, and F. Tirado-Andres, “Ambient modal testing of a double-arch dam: the experimental campaign and model updating,” in *13th International Conference on Motion and Vibration Control (MoViC 2016)*, 2016.
- J. García-Palacios, F. Tirado-Andres, J. M. Soria, I. M. Díaz, and A. Araujo, “Effects of time synchronization on operational modal analysis,” in *6th International Operational Modal Analysis Conference (IOMAC 2015)*, 2015.
- A. Araujo, F. Tirado-Andres, J. García-Palacios, and J. Blesa, “High precision structural health monitoring system using wireless sensor networks,” in *3rd International Symposium on Life-Cycle Civil Engineering (IALCCE 2012)*, 2012, p. 1093–1101.
- J. García-Palacios, A. Samartín, R. Ortega, F. Tirado-Andres, A. Araujo, O. Nieto-Taladriz, J. Blesa, E. Romero, E. Reynders, G. D. Roeck, L. He, and F. Percivale, “Some advances in extensive bridge monitoring using low cost dynamic characterization,” in *International Conference on Experimental Vibration Analysis for Civil Engineering Structures (EVACES 2011)*, 2011, pp. 417-424.

1.8.3. Presentations at conferences

- “Remotely Deployable Cognitive WSN for RNBQ” in *International Defense and Security Exhibition. Defense and Security Innovation Brokerage DSiB 2019*, Madrid, Spain. Celebration date: 30/05/2019.
- “C4W – Command, control, communications and computer for Wireless”, in *International Defense and Security Exhibition. Defense and Security Innovation Brokerage DSiB 2019*, Madrid, Spain. Celebration date: 30/05/2019.

- “High precision structural health monitoring system using wireless sensor networks,” in *3rd International Symposium on Life-Cycle Civil Engineering (IALCCE 2012)*, 2012, p. 1093–1101. Vienna, Austria. Celebration date: 3/10/2012.

1.8.4. Posters

- II edition symposium: “Cuéntanos tu Tesis”, Madrid, 2018.

Chapter 2

Concepts and definitions

Contents

2.1. Terminology	18
2.2. Clock sources.....	23
2.3. Clock model	28
2.4. Sources of delay	29
2.5. Conclusion	30

*Lo que no se sabe expresar
es que no se sabe*

Friedrich Engels

Throughout this second chapter, we review certain concepts that are necessary to understand the rest of the document. At the beginning of the chapter, we discuss the clock sources, their types, and characteristics and we give a brushstroke to the reasons why these clock sources are not perfect, causing all WSNs clocks to end up deviating over time. After that, we explain the terminology that is used repeatedly throughout the document. Concepts are clarified through theoretical definitions and figures, giving way to the presentation of a mathematical model of the behavior of a clock in WSNs. To better understand future synchronization protocols, we make a general review of the most used network topologies in WSNs, showing graphically the distribution of the nodes that make up the network. In the same way, we also clarify the idea of the software layer interconnection model used in WSNs and the modulations most used in radio communications.

2.1. Terminology

This subsection defines various terms commonly used in the document. We explain their differences and similarities through images and their relationship with time synchronization in WSNs.

The terms that we present below are intended to clarify the terminology that is used throughout the rest of the document. In this way, the reader can skip this subsection and come back to it when the reader finds a term that needs clarification.

Physical and Logical Clock

A logical or electronic clock, from now on simply a clock, is a device that counts events indicating the passage of some time interval in an electronic device [8]. The quality of a clock normally boils down to the stability of its frequency, that is to say, the ability of the clock to generate events at a constant frequency over time.

In a WSN, each sensor node has its own hardware-based clock that can be used to assign time-stamps to events that occur in the system. The clock is based on the count of an oscillator at a given frequency and the count is stored in a register of the microcontroller.

Following the same example of an embedded system with a crystal oscillator of 8 MHz with a PLL that transforms that frequency to 96 MHz is easy to understand the difference between the physical and the logical clock. In this case, the physical clock is the one generated physically by the crystal oscillator (8 MHz), but the logical clock is the one generated through the PLL, the system frequency of 96 MHz. In some cases, physical and logical clocks are equivalents.

Accuracy and precision

All clock sources in a WSN are based on some kind of oscillator. The terms accuracy and precision can easily be confused since they can be used as synonyms in the colloquial language, but they are very different in the context of the scientific method.

The accuracy of an oscillator represents how much the frequency of the oscillator can deviate from its nominal value and it is expressed in parts per million (ppm). Accuracy refers to the closeness of a measured value to a standard or known value, in other words, the degree of conformity (closeness) of a measure to a standard or a true value. The accuracy is the fact to be exact or correct, the absence of error. In measurement theory, the accuracy is known as the systematic error due to measurement bias.

Precision refers to the closeness of two or more measurements to each other. The precision of a system, related to reproduction and repetition, is the degree to which these measures can be repeated under invariable conditions showing the same result. Precision is the measure of statistical variability. In measurement theory, the precision is known as the random error due to measurement variation.

For example, if an experiment contains a systematic error when increasing the number of samples, generally, we improve the precision, but we do not improve the

accuracy. The result is consistent but inaccurate. The accuracy improves if the systematic error is eliminated or reduced, but the precision is not modified.

Figure 2.1. Graphical explanation of the term's accuracy and precision. Accuracy is the proximity of measurement results to the true value. Precision refers to the closeness of two or more measurements to each other.

Figure 2.1 shows a graphical explanation of the accuracy and precision' terms. During the present document, that differentiation between both terms will be of crucial importance for the evaluation of the different clock sources and time synchronization strategies.

A good time synchronization strategy is that with high accuracy and high precision. But, a WSN with low accuracy and high precision, would be easy to correct with software techniques such as correcting the offset. A WSN with high accuracy and low precision has more disperse values of the clock, but the average can be accurate. The time synchronization in this system would be more difficult to solve. Figure 2.2 shows the graphical explanation of the terms related to the dispersion of the clocks values. In this sense, high precision is preferable over high accuracy in time synchronization, although it would be desirable to have a system with high precision and accuracy.

Figure 2.2. Explanation of the term's accuracy and precision through the dispersion of samples.

Offset, skew, drift and jitter

Continuing with the terminology used by [25] and [26], we differentiate the following terms:

- **Offset** is a delay of a given clock source or the difference between two clocks. It is also named as phase difference, phase offset (Figure 2.3) or phase error. It can be measured in time units or phase degrees. The offset of the clock C_i is given by $C_i(t) - t$, instantaneous difference between a clock and the reference time. The offset of the clock C_i relative to C_j at time $t \geq 0$ is given by $C_i(t) - C_j(t)$. The offset error is the simplest and most commonly used time synchronization metric. If two clocks are perfectly synchronized, then their offset is zero, meaning they have the same value at that instant.

Figure 2.3. Phase offset (θ) between two signals in the time domain.

- **Skew** is the difference between the frequencies of a given clock and the reference clock. Skew is the frequency difference between oscillators, also named as frequency error or frequency offset. We use the term skew to refer to the first derivative of the clock value concerning time, $C'_i(t)$. The skew of a clock C_i relative to clock C_j at time t is $(C'_i(t) - C'_j(t))$. Clock skew refers to multiple related phenomena that cause a clock not to advance exactly at the same rate as a reference one. After some time, one clock gradually desynchronizes from the other clock. This metric is important because it predicts the growth of the offset over time. If two clocks are perfectly synchronized, then their relative drift is 1, meaning that the clocks have the same rate.
- **Drift** is the second derivative of the clock value for time, namely $C''_i(t)$, that is why drift is also known as frequency drift. Clock drift refers to variations over the skew usually due to environmental conditions. Clock drift is the best indicator to know the stability of an oscillator against temperature variations. In oscillators with temperature control, when the temperature varies, the drift remains stable and does not vary. In contrast, in oscillators without temperature control, it is normal that after a change in temperature, the drift also changes, causing a variation in the oscillation frequency. A graphic representation of the previous terms can be seen in Figure 2.4.

Figure 2.4: Clock terminology: graphical behavior of the clock components over time.

- **Jitter** is the deviation from the true periodicity of a presumable periodic signal (Figure 2.5). The jitter is related to the temporal variations in consecutive edges of the clock signal. The jitter can be decomposed in a random jitter (gaussian) and deterministic jitter. In general, the jitter is considered random, and in this case, can be considered the random component of the skew. Jitter refers to variations in the output signal or its precision. The fluctuation or jitter is the uncertainty in the measurement of the clock.

Figure 2.5. Graphical representation of the parameters of a jittered signal.

Real-Time Clock

Whether based on CMOS oscillators or Crystal Oscillators (XOs), it is possible to create the so-called Real-Time Clock (RTC). The term RTC was created to differentiate this type of clocks from the logical clocks, which simply measure the time counting pulses of a signal, without being directly related to temporal units. RTCs are more similar to the clocks and calendars we use regularly, and they work with seconds, minutes, hours, days, weeks, months and years.

RTCs have the advantage of freeing the system from having to perform time accounting. RTCs can be an external chip or can be an internal software module in the microcontroller. Besides, in some cases, the RTC incorporates some type of battery. This battery is a backup to maintain the time value in case of loss of power or in cases in which the microcontroller enters in modes of low power consumption.

In general, all RTCs are configured to generate an interrupt every second. That is why most of them are based on 32.768 kHz (2^{15}) oscillators, so the oscillators' pre-scalers can be divisible by 2.

Spatial and temporal synchronization

In a WSN, the nodes are spatially distributed. The term spatial synchronization refers to the physical space and indicates that the goal is not to improve the individual time synchronization accuracy of the network nodes but the accuracy of the entire network. When the term spatial synchronization is used it means that the synchronization of the whole is more important than the individual synchronizations.

Temporal synchronization refers to the accuracy of measurement concerning time. In this specific case, the term temporal synchronization refers to the individual time synchronization of the clock of a node to a reference clock. During the rest of the document we use the term time synchronization, or simply, synchronization indistinctly.

Resolution

The term resolution refers to the smallest change in the underlying physical quantity that produces a response in the measurement. The resolution refers to the precision of measurement to time. In our case, the temporal resolution is the ability of a system to distinguish as two different events separated in time. For example, in a system with a clock with a frequency of 1 MHz, where events are generated at a speed that is the inverse of the frequency: $\frac{1}{10^6} = 1 \mu\text{s}$, the resolution is that the smallest time variation (1 μs) it can detect. If two or more events occur in an interval of fewer than 1 μs , they cannot be detected.

Delay and Latency

Latency is a time interval between the stimulation and response, or, from a more general point of view, as a time delay between the cause and the effect of some physical change in the system being observed. Latency is physically a consequence of the limited speed with which any physical interaction can propagate. This velocity is always lower than or equal to the speed of light. Therefore, every physical system that has spatial dimensions different from zero will experience some sort of latency, regardless of the nature of stimulation that it has been exposed to.

Latency and delay are intrinsically linked and sometimes interchangeably used. However, they are not always the same.

Delay is the time it takes for data to travel from one endpoint to another or in other words, the time needed by the packet or bit of data to travel across the network endpoint. Delay is the one-way trip of a packet crossing a network.

Latency, though, can be one of two things. Latency is sometimes considered the time a packet takes to travel from one endpoint to another, the same as the one-way delay. More often, latency signifies the round-trip time. Round-trip time encompasses the time

it takes for a packet to be sent plus the time it takes for it to return. This does not include the time it takes to process the packet at the destination.

Network monitoring tools can determine the precise round-trip time (latency) on a given network. Round-trip time can be calculated from the source since it tracks the time the packet was sent and computes the difference upon acknowledgment of return. However, a delay between two endpoints can be difficult to determine, as the sending endpoint does not have information on the time of arrival at the receiving endpoint.

Synchronization lifetime

We use the term synchronization lifetime to refer to the interval that the clocks remain synchronized in the system. It may be that the synchronization lifetime coincides with the network lifetime, but it may also be that we only want the network to remain synchronized in certain smaller and more specific periods.

Scope

We use the term scope as the size of the region in which a timescale is defined. The scope may be purely geographic (e.g., distance in meters). In other contexts, it is more useful to think about the logical distance, such as the number of hops through a network. As the scope increases, there is usually a cost in cumulative phase error. GPS is a counterexample to this rule, it provides synchronization that is Earth-wide in scope, with the same error bound regardless of location.

Convergence time

Convergence time is the time needed in the system to acquire the synchronization bound for all nodes in the network. The importance of convergence time as a metric is directly tied to the need for energy efficiency.

2.2. Clock sources

Most electronic systems require measuring the passage of time and performing actions at specific times. A device that measures the passage of time, a clock, has a particularly simple dynamics: its state progresses linearly in time [27]. A clock provides the essential “heartbeats” and the timing of electronics [28].

A clock directly depends on the ticks generated by the oscillator. A clock is an electronic device that counts oscillations in an, for example, accurately-machined quartz crystal, at a specific frequency. Clocks are essentially timers. The timer counts the oscillations of the crystal, which is associated with, at least, a counter register and a buffer register. For each oscillation in the crystal, the counter is decremented or incremented by one. When the counter becomes zero or rises to its maximum, an interrupt is generated, and the counter is reloaded from the holding register. Therefore, it is possible to program a timer to generate an interrupt, every second,

or every millisecond, or even in every oscillator tick. In every of those clock ticks, the interruption process is executed so the local time is updated.

The oscillator can vary its frequency due to changes in aging, temperature, battery charge, pressure, etc. [22]. Therefore, the more accurate and more stable the oscillator is, the better results its clock gives, generating measures increasingly accurate and with better performance. The most common oscillators to create clock sources for WSNs are summarized here: from relaxation oscillators to the stable quartz-crystal oscillators.

2.2.1. Relaxation Oscillators

The most simple and inexpensive kind of oscillators consists of a feedback loop that repetitively charges a capacitive load through a resistance until it reaches a threshold level, and then discharges it, beginning the cycle anew (Figure 2.6). They are called RC oscillators if the components needed to build them are resistors and capacitors, or CMOS oscillators if they are based on CMOS digital logic inverters. During the rest of the document, we only discuss the CMOS oscillators, since they are the most common type of relaxation oscillators that are integrated into the microcontrollers. These kinds of oscillators are very cheap but they are not very accurate nor stable.

Figure 2.6. The simplest CMOS oscillator is created hooking an RC feedback around a CMOS logic inverter with Schmitt trigger input [28].

2.2.2. Piezoelectric resonator

A Basic Crystal Oscillator (XO) is an electronic oscillator circuit that uses the mechanical resonance of a vibrating crystal of piezoelectric material to create an electrical signal with a precise frequency. This frequency is commonly used to keep track of time, as in quartz wristwatches, to provide a stable clock signal for digital integrated circuits, and to stabilize frequencies for radio transmitters and receivers. The most common type of piezoelectric resonator used is the quartz crystal, so oscillator circuits incorporating it became known as crystal oscillators.

The quartz crystal oscillators are attractive because they are cheap, small, use a little amount of energy and work surprisingly well despite their low resource requirements; they only need a couple of capacitors to work. Crystal is the common term used in electronics for the frequency-determining component, a wafer of quartz crystal or ceramic with electrodes connected to it. The piezoelectric resonator is a more accurate term for it.

The frequency accuracy of an XO is in the order of 10^{-4} and 10^{-5} , i.e. two similar but unbalanced oscillators will drift between 10 and 100 μ s every second, or

between 1 and 10 seconds per day. However, their frequency stability is significantly better, with frequency changes of 1-part between 10^{-9} to 10^{-11} when averaged over several seconds or more.

The frequency generated by the crystal may deviate from its nominal value and its value is expressed in ppm. The ppm units are like percent which is parts per hundred (10^2) but based on million (10^6). Therefore, 1% is equivalent to 10,000-ppm. For very stable and accurate oscillators with slight variations, it can be used the units of parts per billion (ppb).

For example, a 20-ppm crystal with a nominal frequency of 32.768 kHz means that the frequency can be between 32.7673 kHz (-20-ppm, the minus sign means behind, or 0.999980 nominal value) and 32.7687 kHz (+20-ppm, the plus sign means ahead, or 1.000020 nominal value). These numbers may give the reader a comfortable feeling but remember that with a month the oscillator accumulates 79.1 seconds, so using a 20-ppm crystal to build an alarm clock, it may have an error of 1 minute per month. And for a crystal with an error of 100-ppm translates as $\frac{100}{10^6}$ or 10^{-4} . So, the total error on a day is $86,400 \times 10^{-4} = 8.64$ seconds per day. In a month, the oscillator would deviate $30 \times 8.64 = 259$ seconds or more than four minutes.

An oscillator depends on conditions like the ambient temperature, the supply voltage, the age of the oscillators, etc. all changing over time [22]. Crystals have a temperature coefficient graph meaning that one of the main sources of error is temperature. Timing components are typically characterized at 25°C. As the temperature changes, the system can experience substantially greater drift, depending on how long the temperature condition persists. That is the reason that there are many variations of XO. In the next paragraphs are summarize the main characteristics of the most known XO variations.

Basic Crystal Oscillator

A Basic Crystal Oscillator (XO) is a device that uses the mechanical resonance of a vibrating crystal of piezoelectric material to create an electrical signal with a precise frequency. The accuracy of an oscillator is typically specified in tens of ppm. The initial accuracy at room temperature (+25°C) is dictated by the calibration of the crystal for the most part. A typical calibration tolerance for a basic crystal would be 20-ppm to as much as 1000-ppm. The typical aging rate ranges from 1-ppm/year to 5-ppm/year. This kind of oscillator achieves its temperature stability from the quartz crystal's inherent temperature stability. They are cheap (cost less than \$1) and they require just two additional capacitors to work.

Voltage-Controlled Crystal Oscillator

A Voltage-Controlled Crystal Oscillator (VCXO) is a crystal oscillator with a frequency that can be adjusted by an externally applied voltage. VCXOs have a wide variety of applications in Frequency Modulation (FM) and Phase-Locked Loop (PLL) systems. Typical deviation ranges can be 10-ppm to as much as 100-ppm. The typical yearly aging rate is between 1-ppm to 5-ppm. The temperature stability is between

20-ppm and 1000-ppm. Their cost is in the range between \$1 and more than \$100. This price range varies depending on different factors such as linearity, amount of control voltage applied, the type of crystal being used, circuit design, accuracy, operating temperature range, manufacturing size, packaging, etc.

Temperature-Compensated Crystal Oscillator

A Temperature-Compensated Crystal Oscillator (TCXO) includes an internal temperature sensor. The basic building block for a TCXO is a VCXO with an approximately 50-ppm deviation range and a temperature-sensitive network. This temperature-sensitive network (temperature compensation circuit) applies a voltage to the varactor diode that corrects the frequency of the VCXO at any temperature within the operating temperature range. Typical temperature stabilities achieved from TCXOs would be from 0.10-ppm to 10.0-ppm with a typical aging rate that is between 0.50-ppm/year to 2-ppm/year. From this, we can see that a TCXO offers about a tenfold improvement in temperature stability over an XO. The only disadvantage over a regular XO is the higher cost that can be from \$1 to more than \$10,000.

Oven Controlled Crystal Oscillator

An Oven Controlled or compensated Crystal Oscillator (OCXO) provides the highest resilience to temperature change. OCXOs offer the lowest drift under normal operating conditions. The temperature stability for a typical OCXO ranges from 0.5-ppm to 0.001-ppm. OCXOs typical aging rate is 0.2-ppm /year to 0.02-ppm/year. This type of oscillator has a temperature controlling circuit to maintain the crystal and key components at a constant temperature. OCXOs are typically used when temperature stabilities on the order of 1×10^{-8} or better are required. While this type of oscillator has a tenfold improvement over a TCXO for temperature stability, the OCXO tends to be higher in price (cost between \$25 to \$3,000) and consumes more power. Typical power at +25°C ambient is 1.5 Watts to 2.0 Watts in a steady-state condition.

Microcomputer Compensated Crystal Oscillator

Like a TCXO, a Microcomputer Compensated Crystal Oscillator (MCXO) integrates a temperature sensor. The MCXOs employ external compensation providing at least 10- to 100-times improvement in overall frequency accuracy when compared to conventional TCXO [29]. They have frequency stability between 0.05 and 0.14-ppm, typical TCXOs aging rate is 1-ppm/year and MCXOs cost more than TCXOs (cost over \$50).

Oven Controlled Microelectromechanical Oscillators

An Oven Controlled Microelectromechanical Oscillators (OCMO) or MEMs Oscillators are based on MEMs technology to generate the predetermined frequency with a temperature control [30]. A micro-oven resistor is automatically controlled by

the analog loop to provide active temperature stabilization for the resonator. This kind of oscillator achieves between 50- to 5-ppb frequency stability. MEMs oscillators have excellent long term aging performance with 1.5-ppm in one year, 3.5-ppm in ten years or some of them with 500-ppb in 20 years. The temperature stability of this kind of oscillators is between 3-ppb and 15-ppm. They cost from \$1 to more than \$100.

Atomic clocks

Atomic clocks are the most accurate time and frequency standard known and are used as primary standards for international time distribution services. An atomic clock is the most accurate man-made time device. In an atomic clock, the atom is the pendulum of the clock. When an electron inside an atom transits between two energy levels, it either emits or absorbs an energy quantum of a precise frequency, which is related to the energy difference between the two atomic levels (according to Planck's formula $E = h\nu$). Thus, it creates an atomic oscillator. The most used atomic clocks are the rubidium, hydrogen, and cesium atomic clocks. Cesium atomic clocks employ a beam of cesium atoms. The clock separates cesium atoms of different energy levels by the magnetic field. Hydrogen atomic clocks maintain hydrogen atoms at the required energy level in a container with walls of special material so that the atoms don't lose their higher energy state too quickly. Rubidium atomic clocks, the simplest and most compact of all, use a glass cell of rubidium gas that changes its absorption of light at the optical rubidium frequency when the surrounding microwave frequency is just right.

Since 1967, the International System of Units (SI) has defined the second as the duration of 9,192,631,770 cycles of the radiation corresponding to the transition between two energy levels of the ground state of the cesium-133 atom, which has drift rate of one part in 10^{13} . The Coordinated Time Universal Clock (UTC) is currently a real-time average of about 50 cesium-clocks. This time standard is distributed via shortwave radio (WWV in the US) and satellite GPS around the world.

A comparison of oscillators' accuracy can be seen in Table 2.1 [31] [32]. Through this comparative table, it is observed how the accuracy of the oscillators improves as their cost increases, so there is a tradeoff between the quality of the oscillator and its price.

Table 2.1. Comparison of oscillators accuracy

Type	Accuracy (ppm)	Aging 10 year (ppm)	Cost (€)
Relaxation Oscillator	>1000	--	<1
External XO	10-1000	10-50	~1
VCXO	10-100	10-50	~5
TCXO	0.1-10	5-20	~5
OCXO	0.001-0.5	0.2-2	~50
MCXO	0.05-0.14	~10	~55
MEMs OCXO	0.005-0.05	0.025-15	~100
Rubidium Atomic	10^{-6}	0.005	~1000
Cesium Beam	10^{-7}	--	~3000

Usually, the embedded systems make use of PLL circuits to minimize the impact of the temperature and to stabilize the oscillator. For example, thanks to the use of a PLL the system can transform an oscillator of 8 MHz in a system frequency of 96 MHz making use of a combination of pre-scalers.

Most WSNs make use of XOs as oscillators to build their clock source. If not specified otherwise, in this document, we must consider that the clock source of a WSN is based on an XO.

2.3. Clock model

Even though the frequency of a clock changes over time, for relatively extended periods, it can be approximated with good accuracy by an oscillator with a fixed frequency [33]. Equation 2.1 shows the value of the clock C_i at the instant t relative to a reference clock, which is considered perfect. The Equation 2.1 shows the deviation value of a clock over time

$$C_i(t) = a_i + b_i(t) \quad \text{Equation 2.1}$$

where a_i is the initial clock offset and it does not depend on the time and b_i is the dynamic clock term that is dependent on the time. In general a_i y b_i are different for each node and b_i is approximately constant for an extended period. The dynamic component of the clock can be expanded as follows [34]:

$$b_i(t) = t + \psi_i(t) \quad \text{Equation 2.2}$$

where t represents the *true* time and $\psi_i(t)$ represents the phase error of the clock that has units of time. For an ideal clock source, $\psi_i(t) = 0$ and the dynamic term is $b_i(t) = t$. The phase shift can be decomposed in a polynomial of *true*-time t as follows:

$$\psi_i(t) = \phi + \frac{d\phi}{dt}t + \frac{1}{2}\frac{d^2\phi}{dt^2}t^2 + q(t) = \phi + \dot{\phi}t + \ddot{\phi}\frac{t^2}{2} + q(t) \quad \text{Equation 2.3}$$

where ϕ indicates the time or phase offset. The term $\dot{\phi} = \frac{d\phi}{dt}$ represents the rate of change of phase, the frequency offset or the skew. The rate of change of frequency offset or frequency drift, of just drift, is denoted by $\ddot{\phi} = \frac{d^2\phi}{dt^2}$. Finally, the term $q(t)$ represents the random deviation which is non-deterministic where are included the rest of the terms, such as jitter.

If two clocks are perfectly synchronized, then their relative skew is $\frac{dc}{dt} = 1$, which means that both have the same rate, and their relative offset is 0, which means that they have the same value at that instant. The skew for a fast clock (ahead of the reference) is $\frac{dc}{dt} > 1$ and for a slow clock (behind the reference) is $\frac{dc}{dt} < 1$. Given Equation 2.1, it is possible to calculate the time of an ideal clock, considering no initial offset and no phase errors ($\psi(t) = 0$ and subsequently $[\phi, \dot{\phi}, \ddot{\phi}] = [0, 0, 0]$), so the perfect clock is $C_i(t) = t$. When developers observe short periods of time or the behavior of unstable clock sources, we see a phenomenon called drift, which is the skew variation to time. Keep in mind, that through the literature referring to TS in WSNs, in many cases, it is assumed that the skew is constant for long periods, so researchers use the term skew and drift indistinctly.

2.4. Sources of delay

In practice, the clock sources in each of the nodes in a distributed system run at slightly different frequencies, causing the clock values to gradually diverge from each other. This divergence has different names depends on the source which causes the deviation. Clock synchronization is performed to correct these clock differences in distributed systems.

In WSNs all information exchange is done through sending and receiving messages. Therefore, in any time synchronization process involving message exchange, the following sources of delay will appear in the communication path: send time, access time, transmission time, propagation time, reception time and receive time. In Figure 2.7 we can see the different blocks divided between the transmitter, the receiver and the propagation medium [35] [24].

Figure 2.7. The six delay components in the transmission path

The send time is the time spent by the message transmitter in the application layer. This time is the time spent building the package and sending it to the MAC layer. This time block is non-deterministic as it depends on processor load, speed,

stability, etc. The access time is the time spent in the MAC layer to access the transmission medium. This time is variable unless a time slot is specified for each node to access the channel. The transmit time is the time required to transmit the message to the wireless channel. This time is deterministic and mainly depends on the size of the message and the data rate of the channel. The propagation time is the time required to transfer the packet from the source to the destination through the propagation medium, usually air. This time depends mainly on the distance between source and destination and the propagation speed of the medium. As a general rule, this time is negligible compared to other sources of error since the propagation speed is usually very high and the distances between nodes relatively small. The reception time is the time required to receive the packet from the propagation medium to the MAC layer. This time is deterministic as it was with transmission time. Finally, the receive time is the time needed to transport the packet from the MAC layer to the application layer and also to process its content. Like send time, it is a non-deterministic time that depends mainly on the load of the node processor.

The biggest problem in time synchronization is not that these delays exist but to specify how much is delayed at each stage. Eliminating or correcting the delay at any stage will improve the precision and accuracy of the synchronization protocol. These six delay components always exist, regardless of whether the protocol is simple or complex. All of these sources of error create uncertainty in communication latency.

2.5. Conclusion

In Section 2.1, we show a general review of some terms that we are going to use during this document. The terms accuracy and precision can easily be confused since they can be used as synonyms in the colloquial language, but they are very different in the context of time synchronization.

The offset is the instantaneous difference between two clocks in a precise moment. Two clocks synchronize if their offset is zero. It is possible to read the values of the clocks continuously and also correct them continuously to have a synchronized system, but this process is very demanding and it consumes many resources. That is why the skew calculation acquires importance in time synchronization in WSNs. With the skew, a developer can predict the growth of the offset over time and then make predictive corrections to keep the network synchronized.

Chapter 3

Related work

Contents

3.1. Traditional synchronization methods	31
3.2. Time synchronization strategies for WSNs	34
3.3. Conclusion	49

*El ojo ve solo lo que la mente
está preparada para comprender*

Henri Bergson

This chapter reviews the state of the art regarding the work on time-synchronization strategies that have been published. We begin by discussing the traditional synchronization systems that have been used in wired networks and commenting on the reasons why they do not apply to WSNs. After that, we review those solutions, which under a hardware perspective, have been used to synchronize WSNs. Then we present the most representative software solutions that have been implemented to date. As there are many published proposals, we have decided to summarize them in different groups: reference protocols, protocols that consider security, protocols that consider temperature, etc. Finally, and after the many options discussed, we conclude that to facilitate the correct choice of a time-synchronization strategy, the use of a methodology is necessary.

3.1. Traditional synchronization methods

Since the appearance of the first computers and electronic systems, mechanisms have been sought to be able to synchronize events and to be able to share the information of a global clock. The most widespread methods for synchronizing clocks in distributed systems have been NTP (Network Time Protocol) [36] and GPS (Global Positioning System) [37]. Both solutions are global, have been extensively tested and are used worldwide.

NTP is a networking protocol for clock synchronization between computer systems over packet-switched, variable-latency data networks. In operation since before 1985, NTP is one of the oldest Internet protocols in use. NTP clients synchronize their clocks to the NTP time servers with accuracy in the order of milliseconds of Universal Time Coordinated (UTC) [38] by statistical analysis of the round-trip time. NTP stands out by its scalability, self-configuration in large multi-hop networks, robustness to failures and sabotage, and ubiquitous deployment. NTP is now the de-facto standard for time synchronization on the Internet.

GPS provides localization service by allowing users to measure the time of flight of radio signals from satellites (at known locations) to a receiver. Since GPS time is kept within several nanoseconds of the UTC timescale, GPS has emerged as the pre-eminent method for distributing UTC and frequency worldwide in outdoor spaces.

Figure 3.1. Diagram of how the GPS determines geographic location. A GPS unit compares the time a signal was sent by a GPS satellite (monitored by ground control stations) with the time it was received by the unit. The time difference tells how far away the satellite is. Additional distance measurements from other satellites allow the unit to determine its exact location.

But the limitation of resources in energy, monetary cost, processing and bandwidth of the WSNs, in combination with the potential high density of the nodes makes the methods, that have been traditionally used for time synchronization, inappropriate for such networks. Time synchronization techniques that have been used in wired networks are not suitable for WSNs. NTP, which has been widely used on the Internet, is too complicated to implement and is not energy efficient [39]. NTP is stable but its accuracy is limited in many cases, probably because NTP is an application layer protocol. GPS has a high-power consumption for use in battery-powered WSNs and the electronic GPS modules are not cheap. The GPS module is only functional with satellite coverage and the accuracy is variable depending on the number of connections. Some time-synchronization protocols for

WSNs, which we discuss later in this section, have based their synchronization mechanisms on the methods used by NTP. But all of them have tried to adapt these mechanisms to the limitations of the WSNs.

The use of GPS in WSNs to synchronize the network is more widespread than that of NTP. GPS modules not only distribute global time through their satellite communications, but their main function is to provide global geographic positioning as seen in Figure 3.1. One of the examples can be seen in [40] where the authors make use of GPS modules to synchronize the RTCs of each of the nodes of the network. For this solution, the authors adopt a statistical signal processing framework in conjunction with a long-term frequency calibration of a TCXO made every four hours. In [41] the authors investigate low-cost GPS receivers for the time-synchronization of WSNs with SHM (Structural Health Monitoring) applications in mind. They conclude the work saying that the use of GPS for time-synchronization is valid, but to obtain all the accuracy that the GPS-PPS (Pulse Per Second) signals could provide, large power consumption would be required. The solution implemented in [42] also makes use of GPS-PPS signals to synchronize the nodes of the network for SHM applications. The authors emphasize that software protocols for time synchronization have an increased power-consumption to maintain wireless beacons and that these protocols, in the best case, only reach accuracies of tens of microseconds. On the contrary, the authors do not analyze the additional power consumption that comes with the use of GPS modules, but they do show their interest in future work of looking for techniques to reduce the high-power consumption. The complete solution they provide is a combination of hardware and software techniques that generates a time-synchronization accuracy between nodes of 6 nanoseconds. One of the advantages that the use of GPS has in the time-synchronization of WSNs is that it can reduce, and even avoid, wireless communications due to time-sync packet exchanges. An example where time-synchronization messages are not exchanged between the nodes of the network is in [43], where each node has a low-cost GPS module. After the tests performed, also in the field of SHM, they show an accuracy of sub-microsecond between nodes.

There are no solutions for implementing NTP directly in a WSN due to the limitations of this type of network. The use of GPS has been extended more since the most limiting factor is power consumption, and this can be minimized with larger energy sources, or using the system for short periods. In general, the limitations of WSNs do not allow the use of traditional protocols because WSNs have limitations of size, consumption, and complexity. Neither GPS nor NTP was designed with these limitations in mind.

In addition to NTP and GPS, over the years, there have been other communications standards focused on synchronization, such as PTP (Precision Time Protocol) [44] or TSN (Time-Sensitive Networking) [45]. PTP was originally defined in the IEEE 1588 standard. This standard was designed for systems whose specifications could not be met by NTP or GPS protocols. This system uses a hierarchical master-slave architecture for the distribution of the clock reaching accuracy in the sub-microsecond range. Some research articles such as the [46] have implemented PTP over WSNs achieving an accuracy of 200 nanoseconds offset from

the reference clock of a master node. The [46] solution requires a lot of message exchange and entails a high overhead in the synchronization phase which leads to high power consumption. The PTP implementation was designed for wired ethernet and that is why in its adaptation and implementation for WSNs has difficulties such as complying with symmetric communication paths and accurate time stamping [47]. Even so, some authors like [48] have proposed that PTP become a standard also for WSNs, although without much success. In the proposal of [49], an extension of PTP for multi-hop wireless networks is carried out. This proposal reduces the convergence time and the number of packets required for synchronization without compromising on the synchronization accuracy.

Time-Sensitive Networking (TSN) is based on the IEEE 802.1 Standard Ethernet for guaranteed real-time communication. These standards enable deterministic real-time communication over Ethernet. TSN achieves determinism over Ethernet by using time synchronization and a schedule that is shared between network components. By defining queues based on time, Time-Sensitive Networking ensures a bounded maximum latency for scheduled traffic through switched networks. TSN denotes a set of IEEE 802 standards, which extends the functionality of Ethernet networks to support a deterministic and high-availability communication. Wired TSN for industrial use is undergoing acceptance and deployment, but for the industry, the next challenge will be developing seamless, wireless TSN capability [50]. In this line of work is the publication [51] where there is a state-of-the-art analysis for the implementation of TSN for WSNs and where the main challenges and gaps to overcome are shown.

Although for wired networks the protocols and standards for time synchronization are abundant and have demonstrated their proper functioning, their adaptation for WSNs is not simple and in many cases is not feasible. That is why, as shown in Section 3.2, since the appearance of the WSNs, many authors are investigating and developing solutions to synchronize WSNs.

3.2. Time synchronization strategies for WSNs

Throughout this subsection, we review the state of the art on the works related to time-synchronization strategies. The wide variety of existing strategies is evident because each of them focuses on solving the needs from a specific perspective. Some strategies decide to face the problem of synchronization in the early stages of design, that is, in the choice and design of hardware elements, such as clock sources. Instead, most works focus on software strategies, that is, on so-called synchronization protocols. Besides, as if that were not enough, with the rise of the IoT, time synchronization has also gained importance, developing new strategies focused on the field of IoT but most of them also valid for general WSNs.

3.2.1. Hardware synchronization strategies

We can find different approaches to face the time-synchronization challenge. The hardware approach is based on using additional hardware modules to improve

some characteristics of the network to enhance some parameters, for example, to make the strategy more accurate or more stable. This hardware approach tends to focus on the early design phases of the synchronization strategy, that is, on the best use or use of better clock sources.

Design decisions in the early stages are very important. Choosing a correct clock source directly affects the problem of time synchronization. Temperature changes can influence the behavior of the clock sources, mainly on the stability to offer a constant oscillation frequency.

Several studies have pointed out the importance of the early phases of new product development projects in different fields, such as medicine [52] or building construction [53], to name a few. The importance of early-stage design decisions has to do with several critical decisions that have a great impact on the performance of product development, in our case, the performance of a WSN. Any fault occurring in these early phases in understanding the importance of time synchronization in WSNs becomes a problem to solve in the future.

If we focus exclusively on the analysis, performance evaluation or performance comparison of clock sources, the number of works in the scientific community is not very extensive compared to other topics [54].

The use of synchronization protocols allows us to dynamically correct the deviation of the clock and, therefore, limit the error in time-synchronization of the network. However, the selection of the most appropriate clock source for a certain application reduces the number of operations to be performed by these synchronization software protocols, thus achieving a more efficient and robust operation.

We must keep in mind that the clock directly depends on the ticks generated by the oscillator. The oscillator can vary its frequency due to changes in aging, temperature, battery charge, pressure, etc [22]. There are specific studies of the sensitivity of quartz oscillators [55], ad-hoc developments of clock oscillators, such as OCMOs [30] or CMOS oscillators [56], and analyses about the influence of temperature on the clock skew of the nodes [57]. Therefore, the better and more stable the oscillator is, the better the results its clock gives. Some examples of works of the most common oscillators to create clock sources for WSNs are summarized here: relaxation oscillators, basic XOs, and TCXO.

Relaxation Oscillators

The definition of a Relaxation Oscillator can be seen in Section 2.2.1. The approach of DiStiNCT work [58] is to use the internal CMOS oscillator of the microcontroller as the clock source of the nodes. This kind of oscillator has an accuracy of up to 15,000-ppm instead of the typical crystal oscillators with 10- or 100-ppm. They use this solution arguing some advantages, such as reduced power requirements, improved ruggedness and the ability to integrate the microcontroller and the clock source on the same chip. When talking about time synchronization, this kind of imprecise oscillators is not usually considered as a clock source in WSNs.

However, DiStiNCT achieves a mean time-synchronization error of 6.6 ms with 9 sensor nodes.

Piezoelectric resonator

The definition of a Piezoelectric Resonator can be found in Section 2.2.2. There are many variations of XO like TCXOs (cost over \$5), OCXOs (cost over \$50), OCMOs (cost over \$160), etc. These variations are more stable, but several are the reasons that make these variations not recommended for WSNs. In some cases the main reason for not using them despite their good benefits is the price, in other cases, they need a longer stabilization time or they are larger.

Whether based on CMOS oscillators or XOs, it is possible to create the so-called RTC, as explained in Section 2.1. The practical solution for time-synchronization in WSNs proposed by [59] makes use of RTCs. Authors say that when a local RTC is involved in the system, there is no need for time corrections if the local clock skew is low enough and/or the re-synchronizations are made frequently. Both for short and long term situations, the local RTC is a good solution in terms of energy requirements and stability. So in that work, they use an RTC as high precision and low energy solution. For many applications, this can not be considered high precision but for some others, it is a good enough solution.

Based on the same idea of using RTCs, the work developed in [60] uses an RTC in one of the nodes in the network to improve the robustness of time-synchronization. The authors propose a time-synchronization solution using an RTC claiming the advantage of an improvement in the robustness of offline time synchronization protocols. In further work [61] they make a new proposal where the network is synchronized using several RTCs. The main idea is based on the stability of the RTCs. The researchers state that if all RTCs in the network start synchronized, the global synchronization will stay stable.

As commented in the previous section, the use of GPS in WSNs is not typical but it has still been implemented in some works. In [62] the researchers make use of GPS modules. The authors emphasize that GPS has the advantage of transporting UTC information worldwide. This allows worldwide sensor networks to be synchronized. They highlight that the greatest risk is to have degraded timing accuracy attributable to much noise on the signal transmission path. The solution they propose is a mix of hardware/software that minimizes the overall delay time before the software timestamp the event.

3.2.2. Software time-synchronization strategies

The software approach centers its efforts on the development of software algorithms and strategies to improve the time-synchronization techniques. The task of clock synchronization in WSNs is typically carried out by synchronization protocols rather than by specific hardware solutions. Due to the impact, that time synchronization has on the proper functioning of a WSN, during the last years there have been many analyses and protocol developments around this topic. An ever-

increasing number of works dealing with the design and the implementation of protocol-based techniques for synchronization can be found in the literature. Most of these developments and analyses focus on software protocols that are implemented in the nodes that form the network. Many of those cases are ad-hoc developments for specific hardware.

Reference protocols

Some synchronization protocols have become reference protocols. Most of them because they were the first ones developed or because of the novelty of the way they approach the problem. Figure 3.2 is shown the evolution of works published by year with the topic “time synchronization in Wireless Sensor Networks”. Since 2003 there is a growing interest in the topic. These protocols have been referenced in multiples occasions and then they have become the most prominent examples of time synchronization protocols. The works that we have considered as references are those that have been used as the basis for future developments: Reference Broadcast Synchronization (RBS), Timing-sync Protocol for Sensor Networks (TPSN), Flooding time synchronization (FTSP), and the efficient network flooding and time synchronization with Glossy.

Figure 3.2. A record count of publications about time synchronization in Wireless Sensor Network by year. Extracted from the analysis of the results of the Web of Science [63].

The **Reference Broadcast Synchronization** (RBS) work was one of the first developed [64]. In this work, the broadcast nature of the physical channel is exploited to synchronize a set of receivers. The receivers record their local time when they receive the reference broadcast and exchange the registered values with the neighboring nodes. The biggest advantage of RBS is that it eliminates the non-determinism of the transmitter-side in the communication path. Figure 3.3 can be seen the difference between the traditional time synchronization path and the RBS path. The disadvantages of this protocol are, on the one hand, that the transmitter does not synchronize with the receivers, and on the other hand that an additional message exchange is needed to exchange the time values between the receivers. Besides, the protocol was initially designed for small networks, so it was not designed

to be extended to multi-hop networks, although it can be implemented in those networks.

Figure 3.3. Critical path analysis for traditional time synchronization protocols (left) and RBS (right) [64].

The **Timing-sync Protocol for Sensor Networks** (TPSN) work was also one of the first developed [65] and it bases on the idea of NTP. This algorithm is based on tree topology. TPSN first creates a spanning tree of the network and then performs two-way pairwise synchronization along the edges. Each node synchronizes with the rest by exchanging two messages with its first top-level reference node in the hierarchy. This two-way message exchange averages the uncertainties and estimates the propagation delay. This means that this protocol only compensates for the offset error but not the skew or drift. Figure 3.4 is shown the two-way message exchange between a pair of nodes. This protocol achieves twice better performance than RBS because it timestamps the radio input messages in the MAC layer while RBS was tested with application-layer timestamping. Some disadvantages of this protocol are that it is not able to manage dynamic changes in the topology or that it does not estimate the clock drift of nodes. Also, the two-way message exchange introduces significant communication overhead and wastes energy and channel resources.

Figure 3.4. Two-way message exchange between a pair of nodes in TPSN protocol [65].

The **Flooding Time Synchronization** (FTSP) [66] work uses a single broadcast channel to synchronize neighbors with a dynamically elected reference node. The protocol compensates for the sources of error in the sender-receiver path except for the propagation delay as can be seen in Figure 3.5. This protocol also integrates clock skew estimation with linear regression. This was the first protocol that considered interrupt jitter compensation. This is by recording several timestamps per packet at the sender and receiver sides and then averaging and

normalizing the obtained timestamps to come out with a final timestamp of the outgoing message. One drawback of FTSP is that the clock drift estimation applies a fixed-size regression table that does not change dynamically with the linear time frame underestimation. This means that under different synchronization periods, the table can contain the past drift data points which deviate from the linearity resulting in large regression errors. Another disadvantage is that the periodic broadcast of synchronization messages limits its scalability due to the increase in collisions [67].

Figure 3.5. FTSP decomposition of the message delivery delay over a wireless link [66].

The efficient network flooding and time synchronization with **Glossy** [68] exploits constructive interference of IEEE 802.15.4 symbols for fast network flooding and implicit time synchronization. This emerging trend allows multiple senders to transmit an identical packet simultaneously. The Glossy design temporally decouples flooding from other network activities (Figure 3.6). This assumes that the packet forwarding mechanism has the highest priority and that priority inversion never occurs. These implicit synchronization protocols work over mesh network topologies and can reach sub-microsecond time synchronization accuracy. The flooding message distribution system is based on constructive interference. For this interference to work properly, the maximum time offset of concurrent packet transmissions must not exceed a given hardware constrained threshold. Glossy make use of two clocks: the high-frequency digitally controlled oscillator and the low-frequency external crystal. The Virtual High-resolution Time (VHT) technique has been employed to translate the high-resolution estimate of the reference time to a low-resolution value with high precision at the external crystal clock. Influenced by FTSP, Glossy compensates software jitter by measuring the gap between interrupt reception and interrupt service and then accordingly inserting a certain number of non-operations instructions at the beginning of the interrupt handler [23].

Figure 3.6. Glossy schema showing that the protocol decouples flooding from other application tasks executing on the nodes [68].

Reference protocols variants

In addition to the aforementioned reference protocols, many variants exist. In **Maximum Time Synchronization** (MTS) the idea is to maximize the local information to achieve a global synchronization [69]. The MTS algorithm can calculate skew and offset simultaneously and can be completed at the same time. MTS requires a tuple of four timestamps to determine the relative clock skew. The sender node sends two messages with its local clock that is timestamped in the receiver node.

Based on the same idea, in the same work, the **Weighted Maximum Time Synchronization** (WMTS) is proposed. This variant is used when the communication delay is significant. The weight is the number of hops the logical information takes to reach the node from the reference node.

The **Gradient Time Synchronization Protocol** (GTSP) was designed to provide accurately synchronized clocks between neighbors [70]. In GTSP every node periodically broadcasts its time information and synchronization messages received from direct neighbors are used to calibrate the logical clock. Logical skew and offset are calculated at each node using simple averaging. GTSP uses several timestamping per packet for jitter compensation, like FTSP. But this algorithm requires neither a tree topology nor a reference node.

The central idea of the **PulseSync** [71] is to distribute the information on clock values as fast as possible (similarly to Glossy) while minimizing the number of messages required focusing on multi-hop synchronization. The authors propose a root square order bound algorithm based on their probabilistically analyzed GTSP and FTSP and the lower bound of their respective skews. PulseSync can adapt to network dynamics and changing clock drifts. This protocol is asymptotically optimal in contrast with FTSP which exhibits an error that grows exponentially with the size of the network. Similarly, to FTSP, a root node periodically floods its clock value through the network. Linear regression is used for skew estimation.

MAC protocols

The **Time Synchronized Mesh Protocol** (TSMP) work [72] is the basis for some industry-specific standards such as WirelessHART and ISA100. TSMP is the protocol that combines Time Division Multiple Access (TDMA), channel hopping, and simple time synchronization, which all are efficient methods to increase reliability and determinism which are a very desirable feature for industrial networks. TSMP uses a simple timing synchronization mechanism to maintain TDMA slot synchronization between a time master and several slaves.

Another example of a time synchronization algorithm that is used to develop MAC protocols is the solution proposed by [20]. The authors present a TDMA timing synchronization protocol which is simple to implement and negligible computational or network overhead. The distribution of messages is like the FTSP in the way that this protocol is sending one synchronization packet to all nodes in the network. The time master measures global time several times and adds the average to the synchronization packet just before transmitting it. A receiver records its perception of global time several times at reception, averages them, and adjusts its clock using linear regression.

The authors of [73] propose an uncertainty-driven approach to duty-cycling, where a model of long-term clock drift is used to minimize the duty-cycling overhead. To do this, the authors develop a one-hop synchronization scheme that is integrated with third-party MAC protocol, generating the UBMAC, which aims at minimizing uncertainty for preamble management. This proposal is known for the feature of a **Rate-Adaptive, energy-efficient long-term Time Synchronization** (RATS) algorithm that adapts the sampling period to the user specification on the synchronization bound. In other words, the algorithm can adapt to changing clock drift and environmental conditions. Radio duty cycling has significant importance in the field of MAC protocols and topology management. This predictive duty cycling provided by RATS gives power consumption savings when using the UBMAC protocol.

Temperature variants

Some software proposals try to correct the drift of the clock by compensating for the temperature variations. The authors in [74] stand out the importance of environmental temperature changes as one of the major causes of the clock frequency difference between nodes. Their **Temperature Compensated Time Synchronization** (TCTS) protocol uses the on-board temperature sensor to autonomously calibrate the local oscillator and removes the effects of environmental temperature changes.

The authors of [57] propose an innovative correction factor that minimizes the impact of temperature in the clock skew. They also make use of the onboard temperature sensor as an input for their mathematical model.

The authors in [75] say that the clock skew is a non-stationary random process highly correlated to temperature. They propose a piece of additional information **Aided Multi-model Kalman Filter** (AMKF) algorithm, which uses temperature

measurements to assist clock skew estimation. Making use of the AMKF, the authors also propose an **Environment-Aware Clock Synchronization** (EACS) scheme to dynamically compensate clock skew.

The authors of [76] point to temperature variations as one of those responsible for the loss of synchronization, among other effects. To study the effects of temperature variations in WSN they have designed a testbed called TempLab. They use TempLab to analyze the detrimental effects of temperature variations in three cases: on processing performance, on a tree routing protocol, and CSMA-based MAC protocols.

Linear regression algorithms

In the analysis made by [21] is highlighted that the adoption of synchronization algorithms is preferable to the adoption of hardware solutions, such as GPS-based. This analysis focuses the attention on the adoption of linear regression algorithms for those synchronization algorithms that use one way messaging. The paper proposes a deep performance analysis aimed at highlighting the sensitivity of the regression-based algorithms to some factors that influence the accuracy in typical low-cost applications, such as the finite resolution of the local timing clock, the presence of clock drift, clock high-frequency noise, and low-frequency noise, the presence of latencies due to the radio devices, the presence of latencies due to the microcontroller device. Their main goal is to evaluate the effect of each one of these factors of influence on the overall synchronization performance.

In addition to the previously mentioned algorithms that make use of linear regression to estimate the skew, the **Recursive Time Synchronization Protocol** (RTSP) [77] propose an accurate synchronization of all the nodes in a network to a global clock using multi-hop architecture in an energy-efficient way. This solution uses MAC-layer time-stamping, a dynamic elected reference node, compensation of the propagation delay and adjustment of the timestamps at each hop, estimation of the relative skew and offset using least square linear regression on two data points, adaptive re-synchronization interval, and energy awareness.

Security

Another software approaches highlight the critical and challenging security issue in the WSN time-synchronization process. Wireless networks are more susceptible to security attacks compared to wired networks, where the user needs to physically be connected to the network. The work [78] establishes a fundamental theory of secure clock synchronization. This work is not exclusively focused on WSN, but it gives a global idea of the importance of security in clock synchronization processes. They make a distinction between those one-way communications and those two-way communications. The work shows that all one-way time transfer protocols are vulnerable to replay attacks that can potentially compromise timing information. They also propose a set of conditions for secure two-way clock synchronization mechanisms. In some environments, malicious nodes may surround our network. The

malicious sensor nodes could decrease the accuracy of the whole network by broadcasting fake timestamp messages.

The authors in [79] propose a secure time synchronization model for large-scale IoT networks based on spanning-tree topology. The work developed in [80] says that the attacks to the time synchronization service may incur data distortion or even malfunction of the whole system. They highlight the Sybil attack as one of the most common attack types for WSNs, where a node illegitimately claims multiple identities. They propose a Robust and secure Time-Synchronization Protocol (RTSP) to defend against Sybil attacks. The work [81] is concerned with secure time synchronization for WSNs under message manipulation attacks. They revise their previous protocol MTS, proposing a secured maximum consensus-based Time Synchronization (SMTS) protocol to detect and invalidate message manipulation attacks.

Difficulties in choosing the right time-synchronization strategy

The objectives of time-synchronization strategies are very diverse and, therefore, synchronization strategies can be applied in very different ways. Among all the synchronization protocols, some exclude the reference node from the synchronization of the network. Others need specific network topologies, such as a tree topology. Some algorithms integrate the time-synchronization task as an inherent service while distributing messages. The development of synchronized MAC schemes or the development of secure time-synchronization algorithms are other topics to focus the researcher's effort. Another line of research in the field of time synchronization is the study and analysis of how to compensate for variations in clock sources due to variations in temperature. Some time-synchronization protocols correct the offset, others the skew, others the drift, while others correct a combination of all. Some time-synchronization strategies make use of linear regression algorithms, some make use of statistical solutions.

Whenever developers are dealing with the decision to choose a synchronization strategy that suits the needs and requirements of their application, they realize that they have two options: develop their own strategy or adopt another strategy to their own needs.

Over time, there have been several works in which some ideas that are repeated are highlighted and condensed and other works in which an analysis of the synchronization protocols is shown with a different approach to the rest.

Throughout all the work developed certain ideas are repeated, and as time has passed, and the technique has advanced, they have been refined, detailed and improved. For instance, the decomposition of error sources in time synchronization methods in the network at the beginning was decomposed into four basic components: send time, access time, propagation time and receive time [82]. But over the years, this decomposition of time has become more detailed and today six terms are commonly used, as we saw in Section 2.4: send time, access time, transmission time, propagation time, reception time and receive time [35] [24]. Even these delay

components can be categorized into two classes: deterministic (fixed portion) and stochastic (variable portion).

Another idea that is repeated as a conclusion of many works is the objectives pursued with the implementation of synchronization strategies in WSNs. In most of the works, they talk about coordinating operations between the nodes and collaborating to archive more complex tasks, i.e., data fusion. There is also considerable talk of developing power-saving schemes to avoid collisions between transmissions and to increase network lifetime, for example with the use of power-saving modes in the nodes. The terms scheduling algorithms or MAC schemes such as TDMA are also used where time synchronization is an essential part of transmission scheduling. Finally, many authors highlight cooperative transmissions as another application where time synchronization between nodes of a WSN is indispensable.

The way to approach the calculation of the deviations in the nodes that compose the network is another of the recurrent discussions. It can be approached from a signal processing point of view [83] or a statistical point of view [35] [84]. The statistical approach concludes that while estimating the clock skew between two nodes is possible, it is impossible to determine the clock offset for pairwise synchronization unless the delays in two-way message communication are symmetric.

There has also been much discussion about problems such as trade-offs of complexity versus accuracy and fault tolerance or the close relationship between distributed synchronization and distributed estimation/detection applications. Table 3.1 shows a comparison of various time sync schemes with some of the most representative parameters.

Security in synchronization protocols is another growing concern where latest developments are looking for a secure-time synchronization protocol that is scalable, topology independent, fast convergent, energy-efficient, less latent and less application dependent in a heterogeneous hostile environment [7]. The authors discuss a different type of attack: masquerade attack, replay attack, message manipulation attack and delay attack.

Table 3.1. A comparison chart of various time sync schemes and metrics extracted from Ranganathan survey (2010) [7].

Time Sync methods	Accuracy	Energy Efficiency	Complexity	Scalability	Fault Tolerance	Byte Alignment
1. RBS	Low, 29.1 μ s is the average error per hop	High	High	Good	No	Not Handled
2. TPSN	16.9 μ s per hop	High	Low	Poor	No	Not Handled
3. FTSP	1.48 μ s per hop	High	High	Average	No	Handled
4. GTSP	Average network wide error, 14 μ s Avg. neighbor sync error is 4 μ s	High	Average	Good	Yes	Not Handled
5. PulseSync	4.4 μ s (compared to 23.96 μ s for FTSP between any two nodes)	High	High	Good	No	Unknown
6. All-node	Low	Low	Low	No	No	No
7. Diffusion techniques	Average	Average	High	Good	No	No
8. Mock et.al	High	Low	Low	NA	Yes	Unknown
9. Ganeriwal	High	Average	Low	Good	Yes	No
10. Ping	High	High	Low	Good	Yes	Unknown
11. SLTP	High	High	Average	High	No	NA

With the many developments that have arisen in recent years, there is a difficulty in choosing the right synchronization strategy for a specific application. For this reason, some authors have published some taxonomies cataloging a group of protocols according to certain characteristics. An example of taxonomy can be seen in Figure 3.7 [24] and the classification of protocols according to this taxonomy can be seen in Table 3.2.

Table 3.2. Global objective features of representative clock synchronization protocols from the Swain survey (2015) [24].

Synchronization protocol	Fault Tolerance	Scalability	Sleep scheduling	Energy efficiency	Global vs. Local
RBS	Yes	No	Yes	High	Local
TPSN	Yes	Yes	Yes	Low	Global
TDP	Yes	Yes	Yes	High	Global
FTSP	Yes	Yes	Yes	Average	Global
Global clock synchronization	Yes	Yes	No	Low	Global
GTSP	Yes	Yes	Yes	Average	Global
Consensus based time synchronization protocol	Yes	Yes	No	Low	Global
Probabilistic clock synchronization	No	Yes	Yes	High	Global
Weighted average based clock synchronization protocol	Yes	Yes	No	High	Global
External clock synchronization	Yes	Yes	No	High	Global

Figure 3.7. Different approaches to classify features of clock synchronization protocols from Swain survey (2015) [24].

Some authors [23] stand out that most performance evaluations of the proposed solutions have been limited to theoretical analysis and simulation, ignoring several practical aspects (packet handling jitters, clock drifting, packet loss, etc.). For this reason, their work takes into account the practical aspects of implementation and the authors create a new taxonomy by evaluating a group of protocols as shown in Figure 3.8. The authors conclude that clock drifting and delay variation are the most influencing features when the practical aspects of the implementation are considered.

Figure 3.8. Taxonomy and classification of different protocols based on the implementations aspect extracted from the review of Djenouri (2016) [23].

Another classification based on the way the clock parameters (progressive type vs. recursive type) are propagated can be seen in Figure 3.9 [85]. The same authors have also compared the protocols based on different performance parameters (synchronization error, scalability, computation overhead, etc.) as can be seen in Table 3.3.

Figure 3.9. Classification of clock synchronization protocols based on the propagation of clock parameters extracted from Geetha survey (2017) [85].

Table 3.3. Comparison between different clock synchronization parameters extracted from Geetha survey (2017) [85].

Protocol	Synchronization error	Scalability	Accumulative error	Computation overhead
LCHP	0 s	Yes	Low	Low
2LTSP	9 ms	Yes	Low	High
ATW & AWS	8.9 μ s & 12.42 μ s	No	High	Medium
EBP	0.0511 ticks/s	Yes	Low	High
BILSLR	147 ns	Yes	Low	High
AOTSP	1 ns	Yes	Medium	Medium
MLE Receiver to Receiver	10 ns	Yes	Medium	High
PlSync	0.5 ms	Yes	Medium	Low
MTS	100 μ s	Yes	Medium	High
MTS Switching	2 ns	Yes	Low	High
CDSDBT	0 s	Yes	Medium	High
RoATS	1.96 ms	Yes	Low	High
ATSP	1 μ s	Yes	Medium	Low
DADMM	100 μ s	Yes	Low	Low
FUSA	0 s	Yes	Low	High
TIWICS	6.2 μ s	Yes	Low	High
ETC	0.1 ms	Yes	Medium	Medium
SCReM	ISS 17 TCik DSS 59 TCik	Yes	Low	Low
DRBS	2.45 μ s-1 hop 19.93 μ s-6 hop	Yes	Low	Low
TKDS	10 μ s	Yes	Medium	Medium
SATS	2.3 ms	Yes	Low	Medium
WMTS	5 μ s	Yes	Low	High

And finally, another issue to highlight the synchronization strategies is the technique used to make the timestamp. As can be seen in Figure 3.10 [86], the authors make an analysis of different timestamping techniques (hardware and software), different propagation delay compensation methods, different clock adjustment procedures and they also analyze the impact of synchronization rate and packet loss. They conclude that the performance of a time-synchronization protocol is directly affected by the timestamping mechanism being used. The best option to timestamp the incoming messages is the hardware timestamping-base solution.

Figure 3.10. Various factors affecting packet-based synchronization extracted from Mahmood survey (2017) [86].

3.3. Conclusion

We can extract the main ideas that are repeated throughout the previously presented works. On the one hand, it seems that the decomposition of the error sources has been standardized in six components: send time, access time, transmission time, propagation time, reception time and receive time. We have also notice that the most representative functionalities of a WSN for the implementation of time synchronization strategies are three: data fusion, MAC schemes, and cooperative transmissions. The concern for having secure time-synchronization strategies exist, but it is underdeveloped. The problem of synchronizing the nodes in the WSN can be faced from statistical signal processing or other more deterministic. Besides, many authors have developed taxonomies for the ordering of the different protocols, but each approach is different when considering different parameters and objectives. Furthermore, although many works make theoretical analyses, very few focus their efforts on the real implementation of time-synchronization strategies. Finally, we can conclude that when using timestamping techniques, the hardware timestamping-based solution gives the best results.

We also would like to highlight two conclusions: that there is no unique solution to solve the global problem of time-synchronization in WSNs and that with the increase of applications and requirements, the time-synchronization is more challenging every day.

There is no single time synchronization strategy that maximizes all parameters, nor is there a synchronization strategy that works for every application. For example, in most cases, the more precise the time synchronization strategy is, the more message exchanges between nodes are needed, or the better and more expensive hardware it requires [18]. Strategies can pursue an accurate time synchronization in top layers, such as the application layer [19], or low layers, such as the MAC layer [20].

There is a wide variety of WSN applications. Due to that variety, time synchronization requirements in WSNs are very different, going from very lax to very demanding ones. Every time developers face the problem of choosing and

implementing a time synchronization strategy for a specific wireless application, they realize that huge research is necessary before undertaking the task. Solving the time synchronization challenge must be done considering many other trade-offs because time synchronization is just another service in the WSN application routine.

Due to the usefulness of WSNs in multiple fields, the applications based on wireless end devices have diversified and expanded. Most of the applications for WSNs need time synchronization strategies to operate correctly. Each of these applications has different needs concerning the time synchronization of the network. Thus, many strategies to synchronize WSNs have appeared over the years and continue to do so, adapting the solutions to a specific problem.

As mentioned earlier, the WSN family of end devices continues to grow and is becoming more varied. Therefore, the task of synchronizing a greater quantity and variety of end devices becomes more complex every day. Similarly, since the task of synchronizing a wireless network is becoming increasingly complex, the number of experts in the field is reduced. Therefore, it is necessary to create and use a methodology to address this task of selecting time synchronization strategies appropriate to each scenario and wireless network. Nowadays, there is a lack of methodologies to assess these time synchronization strategies, to compare them, and to determine the ones that are best suited for a given network, scenario and application. As commented in the previous sections, nowadays, no methodology is found among the published works that help developers find the most appropriate solution for each case.

For all those reasons we consider the need to implement time-synchronization strategies in WSNs. This need has led to many works that have been validated solving the problem of time synchronization but applying it to specific cases. Also, today, to carry out the choice of the appropriate time-synchronization strategy is necessary to have deep knowledge in the field of time synchronization for WSN. And finally, of all the work developed to date, there is no methodology to help and guide the developer of WSN applications in the choice of time-synchronization strategies for WSN.

There are many works validated, but that all have been developed to solve a specific problem. Therefore, if application developers for WSN need to implement a synchronization strategy for their network, should they do a study and understanding of the broad set of validated solutions? What would be the steps to follow to adopt the correct solution?

A methodology for choosing and implementing time-synchronization strategies for WSN would allow the final user, the application developer, to be abstracted from the inherent complexities associated with that task. We can see it analogously in the security systems for low power consumption networks. Today, it is not necessary to be a security expert to implement the necessary security strategies in a system. It is only necessary to follow a series of recommended steps, and any non-expert can reach the required level of security in their system.

Chapter 4

Taxonomy of time-synchronization strategies

Contents

4.1. Target.....	54
4.2. Software implementation.....	57

*Cuanto más se dividen los obstáculos,
más fáciles de vencer*

Concepción Arenal

This chapter focuses on the development of a taxonomy, to organize and generate a structure that accommodates not only existing synchronization strategies, but also those that may appear in the future. The development of this specific taxonomy has been done to facilitate and assist in the development of the methodology. Each of the classification parameters is explained in detail by including at least one example.

As seen in Chapter 3, several authors have performed different taxonomies with different approaches. There are taxonomies based on implementation aspects, others based on technical (e.g.: a priori vs post-facto synchronization or how the clock value is propagated) or structural characteristics (e.g.: internal vs external clock synchronization or one-hop vs multi-hop WSNs), others based on clock propagation parameters, etc.

The taxonomy we want to develop has several requirements. The taxonomy to be developed must have the capacity to catalog any time-synchronization strategy.

Therefore, either software or hardware strategies, or strategies already created or strategies that may appear in the future must be able to be classified by the taxonomy. This first requirement forces us to create a taxonomy with enough general categories where every strategy can be classified but at the same time allow us to characterize and differentiate some strategies from others. Besides, the taxonomy must be clear and precise in the classification branches so that any expert in the field of time synchronization in WSN was able to classify the new strategy developed according to this structure. Also, this classification must be sufficiently simple and clear for those developers who are not experts in time synchronization at WSN. These developers must be able to understand simply the fundamental features of the strategies already classified by the taxonomy. Given these requirements and evaluating the taxonomies presented in the previous chapter, we note that none of these taxonomies are adapted to our needs. For this reason, we see the need to develop a new taxonomy.

The developed taxonomy is shown in Figure 4.1 in which a classification tree is observed. For the development of this taxonomy, we have focused more on the way synchronization strategies are implemented and we have downplayed the importance of formal or mathematical development. Also, as shown in Figure 4.1, there are certain branches within a color square. The colored squares indicate mandatory classification points for each of the synchronization strategies. This is that any synchronization strategy must be able to be classified in each of these colorful squares. To be more precise, each synchronization strategy must be able to be classified into each of the nine colored branches in the figure. It is also noteworthy that the proposed taxonomy is divided into two large independent groups: the target with which the synchronization strategy has been developed and the software with which the synchronization strategy is implemented.

The development of this specific taxonomy has been done to facilitate and assist in the development of the methodology. The importance of this taxonomy lies in the fact that, once the methodology has been developed, the cataloging of the different protocols will be more agile and quicker. This makes the introduction and selection of synchronization strategies more agile and quicker during the execution of the methodology.

Through the following subsections, each of the branches of the taxonomy decision tree will be studied in depth.

Figure 4.1. Time-synchronization strategies taxonomy based on target and software implementation.

4.1. Target

Depending on the application, time synchronization is not only recommended but necessary to carry out certain tasks. At the top of the taxonomy shown in Figure 4.1 is the target branch as shown in Figure 4.2. This branch separates the synchronization strategies into three groups according to the type of target pursued with its implementation. Some strategies synchronize the network looking for data fusion, synchronization strategies that look for a cooperative transmission between several nodes and strategies that synchronize tasks in the MAC layer to save energy.

Figure 4.2. Detail of the taxonomy showing only the target block.

Data fusion

All those applications that want to synchronize the generation or capture of events need a time-synchronization scheme. The nodes of a WSN are usually spatially distributed forming a network. The time synchronization between the nodes allows us to coordinate operations and collaborate to reach complex sensing tasks. For example, the data monitored by each of the nodes can be merged with the rest giving a global meaning to the measure. This is called data fusion, where the total meaning is greater than the sum of the individual parts.

Some application examples of this data fusion are the applications that need to do signal processing to the merged data. There are applications very sensitive to the capture of events and it is very important to know if the event recorded by two nodes, which are spatially separated, is the same event at different times or if they are different events.

A clear example where small errors in the time-synchronization between samples taken by different nodes of the network lead to big changes in the results are structural health monitoring. In this type of application, the acceleration of a structure is sampled at different points of it. The data collected by each node are merged to obtain an image of the overall behavior of the structure as can be seen in Figure 4.3. The colors in the figure show the deformation of the structure, that is, the red and orange colors are where more acceleration has been detected and the blue colors where less acceleration has been detected. If, for example, node A and node B, which must vibrate in unison, are out of sync, it could be the case that when the data is merged, when one goes up, the other goes down, showing a movement of the structure that does not correspond to reality, in this case, confusing a flexural mode with a torsional mode. There are other examples, some equally demanding and others more permissive, where the fusion of data is important, such as the environmental monitoring of atmospheric variables in an area, or the estimation of vehicle trajectories.

Figure 4.3. First flexural mode of vibration of a bridge. For this mode, nodes A and B must measure the same acceleration in the same direction. The colors in the figure show the deformation of the structure, that is, the red and orange colors are where more acceleration has been detected and the blue colors where less acceleration has been detected.

Cooperative Transmissions

Under this target of designing synchronization, strategies are included all those synchronization schemes that use constructive interference between signals transmitted by the wireless sensor nodes. Cooperative transmissions allow extending the WSN transmission range as seen in Figure 4.4.

Distributed or cooperative beamforming is a cooperative transmission scheme whereby several nodes in a wireless network organize themselves into a virtual antenna array and focus their transmission in the direction of the intended receiver, potentially achieving orders of magnitude improvements in energy efficiency [87]. Also, this technique maximizes the lifetime of the network while satisfying the requirements of Quality of Service (QoS). There are other uses of this cooperation, for example, the cooperative transmission technique is also used in wireless acoustic sensor networks. These networks are used in reverberant environments with limited loudspeakers.

An example with practical considerations in the implementation of collaborative beamforming on WSN can be found in [88]. In that work, they conduct a thorough study of the related work and analyze the requirements to do collaborative beamforming. The factors that affect the implementation of that type of technique in real scenarios are considered in the work. Therefore, they consider that the nodes to be used can be generic commercial nodes with their associated constraints and that there is no central node that synchronizes the system.

Figure 4.4. Beamforming scenario between wireless nodes and the far-away base station showing the system model for carrier synchronization. Extracted from work [89].

MAC schemes

All those time-synchronization strategies that aim to synchronize communications to avoid collisions when access to the medium are included within MAC schemes. We define it this way because the MAC schemes' implementation, generally, is done in the MAC layer of the WSN protocol stack. This type of strategy can also be known by other names, such as power-saving schemes, power management, scheduling algorithms, transmission scheduling, sleep schedule, etc. By using these coordinated MAC schemes, such as TDMA, or any of its variants, the transmission medium can be shared, for example, the air.

From all the names presented above, it can be extracted that the objective of these strategies is to coordinate the periods of sleep, reception, and transmission of the different wireless nodes that compose the network. That is to say, the sleep cycles and the activity cycles of the wireless sensor nodes are coordinated. Therefore, if the nodes remain as long as possible in a sleep state, and only wake up, in a coordinated way, when it corresponds to them to send or receive information, it allows the node to save energy and extend the lifetime of the network. But while a node is sleeping it can neither send nor receive packets. Only when they wake up, they can re-establish communications. With TDMA, these sleeping, receiving and sending cycles are coordinated. In this way, there be only one sending node at the time and therefore there be no collisions through the shared medium. Only those nodes to which a message is destined are in reception mode and the rest of nodes are in a state of energy saving in sleep mode or performing tasks that do not require radio communication. In this way collisions in the process of sending packets are avoided, the network lifetime is increased, and networks of greater efficiency in the spectrum and energy can be designed.

The TSMP [72] is a protocol that makes use of TDMA. The TSMP devices stay synchronized with each other and communicate in timeslots, like other Time Division Multiplexing (TDM) systems. Such deterministic communication allows the devices to stay extremely low power, as the radios only turn on for the periods of scheduled communication. This kind of protocol uses channel hopping to avoid interference, the packets between TSMP devices get sent on different radio channels depending on the time of transmission.

4.2. Software implementation

The second big group in which we separate the time synchronization strategies is according to their software implementation. As we commented at the beginning of the chapter, this taxonomy should include both software strategies and hardware strategies, but we have considered that whenever a hardware strategy is implemented, it has some associated software. Therefore, hardware strategies are a subgroup of this software implementation and therefore are seen in a subsection.

The organization tree according to its software implementation is divided into six subgroups, where three of them keep branching. Each of the six branches of the software implementation is defined below.

Security

Figure 4.5. Detail of the taxonomy showing only the security block.

In this security group, we catalog the synchronization strategies depending on whether they implement some security level that protects the synchronization algorithm (Figure 4.5). We do not want to catalog security strategies in wireless communications that are synchronized. But we want to catalog time synchronization strategies that within their implementation have considered the peculiarities of security.

Depending on what security measures are implemented in the synchronization protocol we can tell if the protocol has considered the security in its design or not. An example of a secure time synchronization model for large-scale IoT networks based on a spanning tree topology was presented in [79]. The work introduces a secure time synchronization protocol that can work in large-scale IoT networks, which are in a hostile environment where exists malicious nodes. The authors point out that malicious sensor nodes can decrease the accuracy of the entire network by broadcasting fake timestamps messages.

Energy efficiency

Figure 4.6. Detail of the taxonomy showing only the energy efficiency block.

Like the security group, we catalog synchronization strategies depending on whether the authors have considered the energy efficiency of the protocol (Figure 4.6). In this case, we are not considering whether the network as a whole or, in particular, a wireless sensor node, is energy efficient. What interests us in classifying protocols in this branch is whether energy efficiency has been considered when designing the protocol. Therefore, the implementation of those protocols that have taken into consideration this parameter makes better use of the energy resources.

In this way, those synchronization protocols that need a few cycles of CPU execution to work are more efficient than those that require many cycles to carry out their tasks. Therefore, protocols that require many mathematical operations to make estimates, for example, statistical protocols, are less energy efficient than those protocols that only keep a timestamp in memory when a package is received. Similarly, those synchronization strategies that require additional hardware modules and those that make high use of energy are less efficient than those that do not need additional hardware and power consumption.

In the development presented in [62], it is shown how to carry out the synchronization strategy making use of an external module with high power consumption, such as a GPS module.

Complexity

Figure 4.7. Detail of the taxonomy showing only the complexity block.

This group of complexity, like the two previous ones, security, and energy efficiency, catalogs the time-synchronization strategies according to whether this parameter has been considered in its design (Figure 4.7). By the complexity parameter, we mean computational load, not the complexity of having to use many lines of code for its implementation. It may be the case that a few lines of code generate a high computational complexity.

For example, there some protocols, like [75], where the authors make use of multi-model Kalman filter algorithms which increase the complexity of the protocol implementation.

Hardware requirements

Figure 4.8. Detail of the taxonomy showing only the hardware requirements block.

In this classification branch, the algorithms are sorted according to the hardware requirements they need to function properly (Figure 4.8).

One of the branches is integrated with all those protocols that do not need any additional hardware to execute the time-synchronization tasks. In other of the branches, the strategies of time synchronization are integrated where external hardware modules are necessary. In other words, the time-synchronization strategy is based, for example, on an external RTC clock or on a specific GPS module for synchronization which is the main clock source for time management. In our work about clock sources and their influence on time synchronization in WSNs, there are several examples of external modules used with this purpose [90]. As a third case, the most extreme case, there are time-synchronization strategies that develop specific hardware platforms to obtain the desired time synchronization results. An example of this case is found in one of our works where an innovative wireless end device is presented to monitor structures in a synchronized way [91].

Clock correction

Figure 4.9. Detail of the taxonomy showing only the clock correction block.

All time-synchronization strategies can be classified depending on whether they carry out clock value correction in the synchronization process (Figure 4.9). Those strategies that do not perform any kind of clock value correction usually synchronize events once the measurement is finished. These strategies are called post-facto synchronization. An example of a post-facto synchronization protocol is [8]. In that work, the authors present an implementation of a low-power synchronization scheme called post-facto synchronization.

There are the strategies that do correct the clock values, i.e. they correct the offset that accumulates over time. This correction can be carried out on demand, at specific moments, or periodically at regular intervals. As a rule, when the correction is made on-demand, an offset correction is made. When it is done periodically and regularly, you can calculate and estimate the values of the drift and from this value calculate the offset to be corrected in each iteration. Synchronization protocols that implement calculations, estimations, and corrections regularly provide stability to the

clock against changes in environmental conditions, such as changes in ambient temperature. An example of the importance of temperature changes in the frequency difference in clocks is seen in work [74]. An example of a calculation and correction of the skew and offset is presented in the MTS [69].

Software abstraction

The synchronization protocols can be cataloged according to their software abstraction (Figure 4.10). A protocol is more abstract when it abstracts more the programmer from the particularities of its implementation. A protocol is less abstract when it is necessary to know more details about its implementation.

Within this branch of software abstraction, we divide the classification according to three characteristics: the implementation layer in the software protocol stack, the network topology and the distribution of messages in the network.

Figure 4.10. Detail of the taxonomy showing only the software abstraction block.

Implementation layer

In the scope of the WSNs, a software interconnection model based on the Open Systems Interconnection (OSI) [92] model is used but reduced. In addition to the physical layer and the data link layer, following the architecture presented by Callaway in [9], the International Standards Organization (ISO) model for OSI states that the network layer “provides functional and procedural means to exchange network service data units between two transport entities over a network connection. It provides transport entities with independence from routing and switching considerations.” The network layer controls the operation of the network and routes packets and it also addresses flow control. In WSNs there is no specific transport layer because the point-to-point connection is not assured. The session layer in WSN is also dispensed with since in this field there are usually no user sessions, but the network usually has a specific task. In this document we differentiate four layers (see Figure 4.11): physical (PHY), medium access control (MAC), network (NET) and application (APP). In the lower layers

are the PHY layer and the MAC layer while in the higher layer is the APP layer.

The application layer is considered the layer of greater abstraction concerning the physical world, but the one closest to human understanding and therefore where the software develops more easily. The physical layer, which is the layer of least abstraction, is closest to the physical world and farthest from human understanding. In lower abstraction layers the task of developing software is more complex, although once developed it is much more efficient.

Figure 4.11. Diagram of the software abstraction layers in WSN and the associated communication paths delays.

We refer to low abstraction when the implementation of the protocol needs software modifications in the lower layers of the protocol stack. We refer to high abstraction when its implementation can be carried out in the application layer.

In the classification tree, a low abstraction is equivalent to needing a specific implementation layer, while a high abstraction indicates that a non-specific implementation layer can be used. A specific implementation layer means that it is necessary to implement the strategy at a specific layer for its proper functioning, for example, implement the protocol at the MAC layer. An implementation in a non-specific layer means that the strategy has the freedom to be implemented in the layer that the developer considers most appropriate. Carrying out an implementation in the APP layer makes it easier to implement from the programmer. But its disadvantage is that it is a protocol that will drag more non-deterministic time deviations. And these non-deterministic deviations are one of the great enemies of accuracy in time synchronization. Therefore, generally, the protocols that are implemented in the lower layers of the protocol tower tend to be more accurate. In short, there is a compromise between the precision pursued and the abstraction of its implementation.

An example of a time-synchronization protocol that needs a MAC-layer time-stamping technique is the RTSP [77].

Network topology

Network topology group classification refers to whether a synchronization protocol needs a specific network topology to function or whether it is a protocol that can be implemented in any type of network topology. As in the previous case, making use of a specific topology means that for the correct functioning of the strategy, a specific network topology must be used. If the strategy is encompassed within non-specific topology it means that it works correctly regardless of the type of topology used.

As has been said repeatedly, many of the synchronization strategies are ad-hoc solutions. It is for this reason that many synchronization strategies require specific network topology for their proper functioning. For example, the Timing-sync Protocol for Sensor Networks (TPSN) [65] is an algorithm that only works under a tree topology.

Message distribution

The last branch of classification in the taxonomy that we propose catalogs the synchronization protocols according to the distribution of the messages in the network. In the same way, as we have classified the synchronization protocols according to their topology, they can be classified according to the message distribution.

In this classification block, we use the specific and non-specific terms in the same way as in the previous cases of the implementation layer and network topology. Therefore, we have synchronization strategies that only work using specific distribution schemes and those that work independently of the chosen distribution scheme. An example of a protocol that needs a specific message distribution is Glossy. The efficient network flooding and time synchronization with Glossy [68] not only exploits the constructive interference of IEEE 802.15.4 symbols but also needs fast network flooding as a message distribution mechanism.

Once the taxonomy is defined, the development of the methodology is greatly facilitated and can be seen in the following Chapter.

Chapter 5

Developed Methodology

Contents

5.1. Difference between ‘method’ and ‘methodology’	63
5.2. Methodology for implementation of synchronization strategies for WSN	65
5.3. Time-Synchronization Search Tool (TSST)	87

*Para estar absolutamente seguro de algo,
uno debe saber todo o nada al respecto*

Henry Kissinger

This chapter presents both the methodology developed and its associated tools. We begin the chapter by giving a brief overview of the difference between the "method" and the "methodology" and by showing the benefits of using the latter. After that, we explain all the details of the developed methodology based on a block diagram. Once the methodology is presented, we explain the tools developed: the administration tool and the user tool and the database they share. We explain in detail all the necessary parameters to evaluate each time-synchronization strategy and we see how the results are shown to an end-user who makes use of the methodology and the associated tools.

5.1. Difference between ‘method’ and ‘methodology’

Methodology is a word generated from three words of Greek origin: *metà* ("beyond"), *odòs* ("way") and *logos* ("study"). The concept refers to the research plan that allows meeting certain objectives within the framework of science. Therefore, the methodology can be understood as the set of procedures that determine a scientific type of investigation or mark the course of a doctrinal exposition.

It is important to distinguish between the method (name received by each plan selected to achieve an objective) and the methodology (the branch that studies the

method). While method and methodology are closely intertwined, they serve different roles in the research process and should be treated accordingly.

A method is simply the tool used to answer your research questions, for example, a time-synchronization strategy. The term method, also known as research techniques, can be defined as the path to an end; concerning the methodology consists of the procedures that must be carried out to comply with the stipulated by it and obtain true conclusions about the phenomenon or problem that is analyzed.

The methodology is not dedicated to analyzing or verifying knowledge already obtained and accepted by science: its task is to track and adopt valid strategies to increase this knowledge. A methodology is a rationale for the research approach and the lens through which the analysis occurs. In other words, a methodology describes the general research strategy that outlines how research is to be undertaken [93]. The methodology should impact which method(s) for a research endeavor is selected to generate compelling data. With one methodology, you can apply several different methods to support or reject the research hypothesis. The methodology is the systematic, theoretical analysis of the methods applied to a field of study. It comprises the theoretical analysis of the body of methods and principles associated with a branch of knowledge. A methodology does not set out to provide solutions. It is, therefore, not the same as a method. Instead, a methodology offers the theoretical underpinning for understanding which method, set of methods, or best practices can be applied to a specific case. The methodology is a concrete resource for the selection of specific research techniques. The methodology to be efficient must be disciplined and systematic and allow an approach that allows analyzing a problem in its entirety.

In a nutshell, while the methodology is what unites the subject to the object of knowledge and is essential to achieve scientific knowledge, the method is the path or instrument that leads us to it.

5.1.1. Benefits of a methodology

Although the motivation that has led us to develop a methodology has already been introduced, we would like to go a little deeper into the benefits that this methodology can bring to all the researchers in the WSN field who wish to implement time-synchronization processes in their networks.

A methodology allows the developers of WSNs to have consistency between projects allowing an objective comparison between evaluations, priorities, and improvements in the process. The methodology gives the developers consistency in metrics and expectations and gives them the flexibility to reach the complexity of the projects. The methodology allows developers to have better planning based on common experience. Thanks to a methodology a faster implementation of methods can be achieved as it can abstract developers from certain details, at the same time as it allows them to focus their attention on smaller and easier to solve objectives. A methodology often improves the ability to get the job done.

Moreover, the methodology together with the tool, that is also presented in this chapter, allows developers to build a knowledge base that can be easily shared

worldwide. In this way, the entire scientific community can benefit from the community experience.

Our methodological approach is based on quantitative methods since we aim to categorize, rank and identify patterns among these methods. If our objective were to describe, interpret, contextualize and gain in-depth insight into specific concepts, it would be better to approach them through qualitative methods.

In this thesis, we have developed a methodology to address practical problems, not just theoretical ones. As we have seen in Chapter 3 through the state of the art, there is not a standard methodology, not even a methodology, for the implementation of time-synchronization strategies for WSNs. The methodology we present here is intended to offer the researchers a range of possibilities among which they can choose the method that best fits his problem. The aim is to develop, not only a methodology but also a tool that helps researchers to choose between multiple synchronization protocol options, the one that best fits the resources available to obtain a specific result. The tool we have developed aims to facilitate the development of the methodology and to support it, allowing an end-user to apply the methodology effectively.

5.2. Methodology for implementation of synchronization strategies for WSN

The methodology presented in this thesis offers WSN developers a way to choose a time-synchronization strategy in a methodological way. Developers looking for a time-synchronization strategy can make use of this methodology. The proposed methodology provides developers with a set of strategies that fit the needs in each case. Thus, developers do not need to be experts in the specific topic of time synchronization to choose an appropriate strategy. Each time-synchronization strategy has been previously evaluated by a synchronization expert and has been catalogued based on multiple parameters.

Thanks to the taxonomy presented in the previous chapter, depending on the parameters that are important for each application and developer, the methodology leads the developer through a series of steps to obtain a solution, or a group of solutions, adapted to the specific case. It is ensured that, regardless of the level of expertise of the final users of the methodology, the solution that is finally chosen has been technically validated. The methodology compensates for the lack of experience that developers may have as it adapts the level of abstraction to the developer's experience. In these cases, in combination with the tool, the methodology reduces development time by offering solutions graphically pre-evaluated by experts. These features ensure a successful decision for everyone.

The methodology we propose is based on a series of steps that can be observed in Figure 5.1. In each of the steps, which we call blocks, the user must evaluate a series of characteristics or parameters. All synchronization strategies are available at the beginning of the cycle. Developers select certain characteristics or requirements in each of the blocks. Since not all the strategies fit the requirements imposed in each

of the blocks, the number of available strategies is reduced as one advances through the blocks. Once the entire process is completed, all the remaining time-synchronization strategies meet the commitments established between the requirements and the available resources. This procedure is called "time-synchronization strategy filtering". This filtering process makes it possible to select time-synchronization strategies from all the possibilities available in the tool. In this way, the user can find one, or a set of time-synchronization strategies, adapted to the needs of the application and the scenario.

The set of blocks that constitute the methodology guides the user in analyzing all the important issues when choosing a time-synchronization strategy. Associated with this methodology, a set of tools has been created. These tools are dynamic and can handle the appearance of new solutions, such as new synchronization protocols or situations, or changes in the needs or the environment. This is because the filtering process discussed above is done on a set of strategies that can grow or shrink at any time. This means that if, in the future, new time-synchronization strategies appear, they can be included in the set of available strategies to be filtered.

The methodology guides the developer through a sequence of topics related to time synchronization from the most abstract to the most concrete. In this way, the methodology can filter, through multiple interrelated parameters and according to the interests of the developers, and select the most appropriate synchronization strategy in each case.

The methodology does not only lead the user to automate the selection of a time-synchronization algorithm; it can also offer non-expert developers in the field of time-synchronization solutions adapted to their needs. If the needs of the application change, this methodological process is carried out again and a new adapted solution is obtained. An ad-hoc solution would usually be a more fitting solution, but most WSN application developers are not experts in the field of time synchronization, and the development of an ad-hoc synchronization protocol would take a lot of effort and development time on their part.

In addition, the use of the methodology allows final users to save time. On the one hand, the user does not need to take additional tests to check if the strategy the user wants to implement works correctly, because other users before him/her have already made those tests. On the other hand, having a methodology and a tool associated with a set of synchronization strategies saves us time in choosing them, since users do not have to spend time searching for the available options.

In this section, every step of the methodology is explained. The objective of our methodology is not to specify what should be done in the evaluation, but rather how to do it. However, in this work, we also present a set of tools that make the execution of the methodology possible.

The methodology we propose can be represented by a flowchart, as shown in Figure 5.1. This flowchart can be divided into six main blocks of inputs (green) plus the output blocks (blue). The white blocks represent the phases where decisions must be made, that is, where the combination of methodology and tools compare the input requirements with the available time-synchronization strategies and filters them.

Figure 5.1. Flowchart of the proposed methodology, where the input blocks are green, and the output blocks are blue. White blocks are where the methodology makes decisions.

The flow of the methodology shows, one input block after another, the evaluation of a block of parameters and how the possibility of finding a viable solution is evaluated. If a possible solution is not found given the requirements, a report is generated. This report informs the developers what the parameters are, or the trade-offs between them, that make a viable solution impossible. The cycle ends after determining the non-feasibility. On the contrary, if all the input blocks are evaluated, and there are still viable solutions available, a spider or radar chart is generated (see Figure 5.2). This chart shows the relationship between the parameters and the possible solutions that meet the requirements that the developers have introduced. If the developers are not satisfied with the solutions offered by the methodology, they can always start the process again and modify the input parameters.

The order of the input blocks, as shown in Figure 5.1, can be changed. The established order ensures a method of filtering the time-synchronization strategies from the most abstract to the most concrete characteristics. The first block of inputs in the methodology evaluates the target of the time-synchronization strategies. We call that block the “synchronization target” block. Following the taxonomy presented in the previous chapter, we have chosen three of the most common targets pursued when developers want to synchronize a WSN. Most time-synchronization strategies belong to one of these targets: spectral or energy efficiency making use of MAC schemes, data fusion, or communications with cooperative transmissions. If the reader returns to Chapter 3, where we looked at the state of the art and the different existing works, he/she would see that all the works can be included in one of these three objectives.

The second input block is the “power consumption and monetary cost” block. In this block, the methodology makes the users evaluate the additional monetary cost

and the power consumption of those tasks related to the time-synchronization strategy.

The third block evaluates the scenario where the WSN is to be deployed. This block is called “scenario and security analysis” because it is here that the methodology makes the user evaluate whether the scenario facilitates or hinders the time-synchronization tasks and how critical the security and authenticity of the time source are. The variation of the environmental conditions, especially changes in temperature, can affect the behavior of the clock sources to a greater or lesser extent. Time-synchronization strategies can be different if the network is oriented to indoor spaces, or if it is a network that is planned to be installed in open and outdoor spaces. In this block, other concepts, such as the network size or the convergence time, are also evaluated. The network size is the number of nodes that must be synchronized in the network. The greater the number of nodes, the more complex and/or slow the message distribution is likely to be. As previously mentioned, each application may have specific requirements that need to be known. The block that makes the user evaluate those specific “application requirements” is given that same name. In this block, the time accuracy needed for the synchronization strategy for the proper execution of the application is evaluated. The maximum time-synchronization error allowed is a key factor for the correct operation of the application. This is one of the most important parameters in terms of synchronization strategies. It sets the error limit that the network can tolerate and, if it is exceeded, that time-synchronization strategy is considered not valid. The synchronization convergence time is the time the strategy needs from the start of its execution until it reaches the synchronization objective. This convergence time can be affected by the network size. Certain synchronization strategies run correctly for a small number of nodes, but if the number of nodes in the network grows, the accumulated error also grows.

In the last two blocks of inputs, the user must evaluate the software and hardware resources. The tools used to execute the methodology can filter the available synchronization strategies based on the software and hardware limitations. The number of time-synchronization strategies is reduced when the software or hardware required is more demanding. Some protocols need to be implemented at specific layers of the software stack, such as the MAC layer. Other protocols only work on certain operating systems or by making use of specific radio modulations. Some protocols do not require any specific hardware, while others need external modules or specific transceivers. Each new requirement may reduce the number of time-synchronization strategies, and it may be that no one option meets all the requirements.

Any of the input blocks can be skipped, considering that there are no limitations or, in other words, that there is freedom to choose any available option.

Once all the input blocks have been evaluated, the tool generates a radar chart. This radar chart is a graphical method of displaying nine parameters that characterize a time-synchronization strategy in the form of a two-dimensional chart. Each of the parameters has a quantitative value, from 0 to 10, and these values are represented on axes starting from the same point at the center, as shown in Figure 5.2. We delve into the meaning of each parameter and its values in Section 5.2.3. Generally, each time-synchronization strategy has just one main “synchronization

target”. For each target, a set of solutions can be considered as valid. The rest of the input blocks of Figure 5.1 are represented by the nine radar parameters. The power consumption and the monetary cost are related to the second input block, the security; while the topology of the network and the message distribution represent the “analysis of the scenario and security” and the “application requirements” is represented by the precision and accuracy and stability of the clock. The “software requirements” is represented by the software abstraction parameter. Similarly, the “hardware requirements” is represented by the hardware abstraction parameter. All the available time-synchronization strategies have numerical values for each of the evaluated parameters. All the strategies that can be plotted, whose area stays inside the area defined by the filter, are offered to the users as the synchronization strategies that meet the WSN requirements. When developers select any of the solutions proposed by the radar chart, the methodology cycle is considered finished. In certain cases, the tools that execute the methodology can propose what they consider to be the best option among all the proposals. If the developers agree with this solution, the methodology cycle finishes. In the end, in special cases, a piece of pseudocode is offered to facilitate the implementation task of the strategy in the WSN. The methodology does not define time-synchronization protocols, it catalogs them and offers the ones that are adjusted to the requirements of a specific application.

Figure 5.2. Graphical representation of the insertion of a new time-synchronization strategy into the tool database with all the possible solutions. This example shows the insertion of solution 4.

Through this methodology, we can standardize, structure, and organize how time-synchronization strategies are filtered and selected, based on the application and the user decisions or needs. This methodology is repeatable, reproducible, and undoubtedly reports results. The methodology helps to tackle different projects in the same way and allows any developer to select the appropriate time-synchronization strategy for his or her WSN, regardless of his or her experience in the subject, abstracting the developer from the details of the protocols and generally improving the development time.

The methodology can be used from two points of view: the administration point of view and the user's point of view. From the administrator's point of view, the objective is to incorporate new time-synchronization strategies into the system so that the options are more and more extensive. From the user's point of view, the aim is to find a time-synchronization strategy that meets the requirements imposed by a specific application.

5.2.1. Administration view

From the administrator's point of view, a WSN synchronization expert is responsible for evaluating the new synchronization strategy. As will be seen in Section 5.2.3, the administrator must parameterize the new strategy and assign a value to each of the parameters. To go through all the parameters, the administrator follows the flowchart shown in Figure 5.1. A graphic representation of the insertion of a new time-synchronization strategy into the tool database can be seen in Figure 5.2.

Whether the point of view is the administrator's or the user's does not change anything about the methodology. The communication between both points of view is a table or database that must store the parameterized strategies that are added. The database of possible solutions simply grows with every new incorporation. The new strategy should be analyzed by the experts and broken down and cataloged according to a parameter evaluation. In this administration view, time-synchronization strategies are defined based on nine numerical values representing nine parameters. The parameters can use values from 0 to 9. We reserve the value of 0 to indicate that the parameter has not been defined and therefore no value has been assigned to it. For parameters that have been evaluated, values from 1 to 9 are used. It is thus possible to know the level of the parameters by a fast, visual means; knowing that the closer to the center it is, the lower its value will be and, generally, the more demanding the parameter. The farther from the center, the higher its value and, generally, the laxer the parameter.

We go deeper into the evaluation of each of the parameters in Section 5.2.3, where we analyze the meaning of each numerical value for each parameter. Although, as a rule, the level associated with the parameter indicates its level of demand, there are some special cases where the numerical value is associated with a specific type of behavior.

In short, we can say that the lower the parameter levels, the smaller the area enclosed when the points are joined. A small area indicates that, in general, it is a very demanding time-synchronization strategy. However, the higher the values associated with the parameters, the greater the area enclosed in the radar chart. A larger area visually indicates that the protocol is more relaxed and therefore less demanding. This visual approach is represented in Figure 5.3.

Figure 5.3. Graphic representation of a very demanding time-synchronization strategy (left) and therefore with a small area and a very relaxed time-synchronization strategy (right) and therefore with a much larger area.

On the lefthand side of Figure 5.3, we can see a representation of a demanding synchronization strategy, where all its parameters have values close to 1. On the righthand side of the figure, we can see a relaxed synchronization strategy in all its parameters with values close to 9. Visually and quickly, we can see how the left area is much smaller than the right area. Similarly, quickly and visually, without looking at the details, we can extrapolate that the strategy on the right is much more relaxed and the one on the left much more demanding.

Once incorporated into the database, this new strategy is part of the possible solutions that are offered to developers who make use of the methodology and the selection tool.

5.2.2. User view

From the user's point of view, the aim is to create a filter to reduce the number of valid options for a specific case. Following the order of the blocks of the methodology, the concepts on which the end-user is guided analyze all the fundamental characteristics of a synchronization protocol. From this perspective, it must be considered that the user does not have to be an expert in the field. Therefore, the way to obtain the necessary information to set the filter must be done through simple questions that every user may know. Some of the parameters to be evaluated can have a direct question, since it is assumed that the user has enough knowledge to answer them. For example, the end-user may be asked whether he/she wants to have a secure protocol (level 1) or whether the security parameter should be more relaxed (level 9). From the user's point of view, the special value 10 can be used. This value indicates that this parameter is not relevant for the creation of this specific filter. A value of 10 does not perform any filtering and all strategies will be valid concerning that parameter. In some parameters, the questions should be masked and approached from an end-user point of view, as she/he does not need to know the existence or implementation details of certain features. For example, instead of asking about the specific network topology, the user may be asked about the number of

nodes in the network, how they are physically distributed and whether they can communicate freely with each other. In Figure 5.4, the flowchart followed by this input tool is shown. Each of the topics of the input blocks is evaluated.

Figure 5.4. Flowchart of the user tool where all the input blocks are evaluated and queries to the time-synchronization strategies database are generated. The cycle ends when there is no solution available to fulfill the requirements or when all the input blocks are evaluated, and the radar chart is generated.

Each time a block from Figure 5.1 is evaluated, one or more parameters are assigned a value. Each time a parameter is evaluated, the solutions available up to that point are offered. If there are still available solutions after the evaluation, another input block is processed. The radar chart is regenerated every time a new input block is processed. If there are no remaining input blocks, the solution is chosen before the cycle ends. If there are no available synchronization strategies to fulfill the requirements after all queries have been made to the database, a report is generated and the cycle ends.

As in several of the cases, the questions about the parameters should be masked by several simpler questions, the input forms contain both general and detailed questions. Depending on the answers to the general questions, some of the detailed questions can be considered irrelevant. The more information obtained from these input forms, the more accurate the result will be. The more details are provided, the narrower the selection filter of time-synchronization strategies. The narrower the filter is, the fewer strategies appear as available. Different users may have a relative interpretation of the same metric, for example, the accuracy of an oscillator. This process, and the table proposed in Section 5.2.3, are responsible for transforming the relative into the absolute.

There is a direct relationship between the requirements and the number of available solutions. If, on the one hand, due to the requirements introduced in the input blocks, the filter is too selective and restrictive (smaller area in the chart), it may be that no solution meets all the desired parameters. If, on the other hand, few details are provided, the selection filter is too flexible and tolerant (bigger area in the

chart), so a large number of possible solutions are offered, many of which are very general and poorly adjusted to a specific problem.

Figure 5.5. Graphic representation of the application requirements creating a user filter.

Figure 5.5 shows the graphic representation of an example of application requirements, or as we say, the definition of the user filter. In this case, it is a large area filter, so it is known beforehand that it is a relaxed filter. We can see how the parameters of monetary cost, security, network topology, and message distribution, are not important for either the user or the application. These parameters have been assigned the value 10, indicating that any solution is acceptable. In the power consumption parameter, a value of 8 has been set, which indicates that, although it is not very demanding, it is considered. The same applies to the value 7 for clock stability. Software abstraction is starting to be of interest to the user and the parameters of accuracy and hardware abstraction, with a value of 4, are already important features for the user and the application.

All the details of the parameter evaluation, as well as the meaning of the different levels, are explained in the following sub-sections.

5.2.3. Parameter evaluation

There are nine parameters in total plus a target selection. All the parameters, except the synchronization target, have a numeric value ranging from 1 to 9. The value 0 means that the parameter has not been defined (from the administrator's point of view) and the value 10 indicates that this parameter has no importance in the definition of a filter (from the user's point of view). These nine parameters are represented on the radar chart. In the numerical scale, with which the parameters are evaluated on a level scale, the value 1 is the most demanding and 9 the most relaxed. In this level scale, the user, when defining the filter for selecting synchronization strategies, has a graphic view of whether the filter is very demanding or relaxed. The larger the area of the filter, the more relaxed it is; while the smaller its area, the more demanding it is. Table 5.1 shows a summary of the correspondence between the blocks to be followed by the methodology and the parameters that are evaluated in the radar chart. Each of the parameters, and the meaning of their level scale, are explained below.

Table 5.1. Correspondence between the blocks to be followed by the methodology and the parameters that are evaluated in the radar chart.

Methodology 1st Block	Name	Synchronization Target	Radar Chart Axes	Synchronization Target
Methodology 2nd Block	Name	Power Consumption & Monetary Cost	Radar Chart Axes	Power Consumption Monetary Cost
Methodology 3rd Block	Name	Scenario & Security Analysis	Radar Chart Axes	Security Network Topology Message Distribution
Methodology 4th Block	Name	Application Requirements	Radar Chart Axes	Accuracy Clock Stability
Methodology 5th Block	Name	Software Requirements	Radar Chart Axes	Software Abstraction
Methodology 6th Block	Name	Hardware Requirements	Radar Chart Axes	Hardware Abstraction

Synchronization target

The synchronization strategies are very different according to their purpose. Although an improvement in hardware means that all time measures improve, software solutions are very dependent on the target of the synchronization protocol. Following the taxonomy presented in Chapter 4, we differentiate three types of targets: MAC schemes, cooperative transmission, and data fusion. The synchronization strategies of the MAC schemes pursue the objective of synchronizing the MAC layer for the successful operation of some communication protocols. In this way, all the communication protocols can use differentiated time slots to transmit packets and avoid collisions. The cooperative transmission applications synchronize their communications to achieve collective goals, for example, to make constructive interference. The data fusion applications need synchronization to merge all the data collected by the individual nodes to make up a general picture of the whole network. In the administration view, one or more targets can be selected as the strategy may meet several targets simultaneously. In the user section, only one target can be selected.

We can find examples of protocols that pursue each of the targets: for instance, an example of a MAC scheme is the TSMP [72], examples of cooperative transmissions are the efficient network flooding and time synchronization with Glossy [68], which make use of constructive interferences, while an example of data fusion is the TPSN [65].

Power consumption

Table 5.2. Values associated with the level scale for the power consumption parameter ranging from 1 to 9.

Power Consumption	1 <5 μ W	2 <50 μ W	3 <500 μ W	4 <1mW	5 <5mW	6 <50mW	7 <500mW	8 <1W	9 <5W
-------------------	---------------------	----------------------	-----------------------	---------------	---------------	----------------	-----------------	--------------	--------------

Regarding power consumption, if there are transitions in the microcontroller state between an active mode and low-power mode in the implementation of the node's software, it is necessary to take into account the clocks that can remain active in low-power mode and their accuracy. The same applies to the transitions between an active mode and sleep mode in the radio transceiver. Those transitions can affect the time-synchronization strategy to follow. If the user is looking for a solution with limitations in power consumption, or the power consumption should be as small as possible, only certain synchronization strategies will be viable. One piece of information that the administrator may know is the minimum and maximum distance between the network nodes. With these data, and depending on the radio transceiver, it is possible to estimate the radio transmission power needed to communicate the nodes, and therefore the power consumption of wireless communications.

If the administrator has evaluated the power consumption associated with the synchronization strategy at the nodes, a numerical value can be associated with each consumption range. The administrator must consider that the use of radio transmissions is usually very consuming; therefore, in many cases, doing more operations locally and performing fewer radio communications is usually more energy-efficient. To translate the relative evaluation of each administrator and user into an absolute scale, we have created Table 5.2, where numerical values are associated with the scales of each parameter. For the case of power consumption, we have a scale where the most demanding power consumption, the one associated with value 1, is the one that consumes less than $5\mu\text{W}$ and where value 9 is associated with the most relaxed situation, and where there can be consumptions up to 5W . The most demanding and relaxed situation has been calculated based on a system with a 5-volt power supply and current consumption of $1\mu\text{A}$ and 1A respectively. These measures are based on realistic scenarios where the communication messages and the necessary execution time have been considered. This power consumption parameter refers only to the additional power consumption provided by a single-node synchronization strategy. This parameter is an absolute value and is not a percentage value of the total power consumption of the node. We have used units of watts so that the ranges were independent of the value of the system's power supply.

As commented before, for the data input and the graphical representation of the term power consumption, we must bear in mind that a greater value indicates that the strategy has a greater consumption of energy. Therefore, the lower the value of the level, the lower the power consumption of that protocol. From the user point of view, small values filter out those protocols that consume a lot of energy. Large level values accept protocols with both small and large consumption. An example of a synchronization strategy that makes use of GPS, and therefore high power consumption, is [62].

Monetary cost

The administrator must evaluate the monetary cost associated with implementing a synchronization strategy. Thus, if some external module is needed to implement the synchronization strategy, it is necessary to know its price. For certain

users, the monetary cost is of vital importance, since they will have to discard some solutions whose cost exceeds certain limits. For example, the use of GPS modules or some expensive TXCO.

Table 5.3. Values associated with the level scale for the monetary cost parameter ranging from 1 to 9.

Monetary Cost	1 < 1%	2 <15%	3 <30%	4 <40%	5 <50%	6 <60%	7 <75%	8 <90%	9 <100%
---------------	---------------	---------------	---------------	---------------	---------------	---------------	---------------	---------------	----------------

As can be seen in Table 5.3, for the monetary cost parameter, we have defined limits in percentages. In this case, we preferred to set the parameter values as a percentage of the price of an end node. In this case, if the end product has high costs, it may not matter so much to assume higher costs in the synchronization strategy. On the other hand, if the cost of the node is very low, the cost of the strategy may have a significant impact. This percentage refers to the cost of the node, with the cost of the node being 100% of its price. The value 1 is associated with those strategies that have no extra cost or a very low cost, that is, those strategies that have an associated cost lower than 1% of the node cost. The value 9 is associated with those strategies that have a very high cost, that is, whose cost is between 90 and 100% of the node cost. If, for example, we have a node whose cost is 10 euros, those strategies with a cost of less than 0.1 euros would be assigned the value 1, while those with a cost of more than 9 euros would be assigned the value 9 or 10. The rest of the values are defined in Table 5.3. If the nodes of a network with wireless communications must be connected to wired power, then the administrator must give them a value of 10. With this scale of values, from the user point of view, the radar chart will have a very large area for those values that are more relaxed, that is, higher consumption, and will have a smaller area for those more demanding values that seek less power consumption.

An example where an administrator can value this parameter could be the case of including an external component such as an OCXO [43]. This external module increases the price of the node and, depending on the number of nodes that make up the network, this monetary cost can be assumed or not.

In this parameter, only the monetary cost is evaluated. The cost of resources, for example, the cost of implementing highly complex strategies, are evaluated in a distributed manner. That is, if a strategy is computationally complex, with high computational overhead, it is probably associated with a higher power consumption, higher monetary cost or require specific hardware. These three parameters are already evaluated separately.

Security

The security of the time-synchronization strategy indicates how critical it is to ensure the authenticity of the time source. Some applications require a secure and reliable time-synchronization scheme for the WSN because they are placed in hostile environments. Those time-synchronization protocols incorporate special strategies to

fight against fake timestamp broadcasters or they have some prevention strategies against malicious attacks.

Table 5.4. Values associated with the level scale for the security parameter ranging from 1 to 9.

Security	1	Very secure. Security as a priority	2	3	4	5	6	7	8	Not secure. No security strategy has been implemented	9
----------	---	--	---	---	---	---	---	---	---	--	---

An administrator, when incorporating a new synchronization strategy into the database, must assess whether it has been designed with security in mind. For the graphic representation of the security term, it must be considered that higher values indicate lower safety in the protocol. Similarly, small level values indicate that the protocol has a higher level of security (Table 5.4). The value 1 has been associated with those protocols which are stricter in terms of security and, therefore, where security is a priority. On the other hand, the value 9 has been associated with those protocols that are not very secure or not at all secure; therefore, security is not considered in their design. Intermediate values can be extrapolated from both ends.

By filtering the available protocols, a user can allow all protocols regardless of their security by assigning a value of 10 to the filter or he/she can search only for secure protocols by lowering the value of the filter.

An example of the security parameter is the secure time-synchronization model for large-scale IoT networks based on a spanning tree topology [79].

Network Topology

The scenario where the network is deployed affects several of the parameters that must be evaluated by the administrator to set the parameter values. The network size is the number of nodes that make up the network. There are many factors, such as the network topology, the message distribution scheme, or the difficulty of synchronizing many or just a few nodes in the network, among others, that depend on the network size. The physical location where the WSN is deployed determines which synchronization strategies are more adequate in each case. The synchronization scope is the range of action where the time synchronization must be effective. The administrator should know if the time-synchronization strategy needs to synchronize all the nodes in the network or only some of them. What is more, the WSN sometimes needs to be synchronized with third-party references, such as the UTC. Another way to get relations and to make estimations is the fact that, knowing the number of nodes in the network in combination with the span of communications and the space between nodes, the administrator can get an idea of the topology of the network. Also, according to the number of nodes, the clock dispersion and therefore the error of the time-synchronization strategies vary.

Table 5.5. Types of network topologies associated with the level scale for the network topology parameter which has five options.

Network Topology	1 Mesh	3 Tree	5 Star	7 Lineal	9 ALL Valid
------------------	---------------	---------------	---------------	-----------------	--------------------

The topology of the network is also related to the scenario. Some synchronization protocols make use of specific network topologies, such as linear, star, tree or mesh. The level values for this parameter are somewhat special. Instead of having nine values, there are only five options, with values 1, 3, 5, 7 and 9 (Table 5.5). The most demanding topology, associated with the value 1, is the mesh topology. In this topology, all nodes must have communication with all nodes through multi-hop communications. The next level of demand is the tree topology, where it is necessary to establish the hierarchy of levels and where there can also be multi-hop communications. We move on to star topology, which can also be seen as a one-hop tree topology. Then we have the linear topology, which is just a tree topology reduced to a single branch. And finally, the least demanding level, to which we assign the value 9, are those strategies that can work in any type of topology.

It should be noted that there are network topologies that allow other topologies to be executed in turn. This scale has been designed valuing this property as well. For a user who has no special requirements, he or she can set the filter to value 9, allowing the selection of all strategies, both those that work in all topologies and those that need a specific topology. More demanding users who need a tree topology, for example, will set the filter to value 3. This allows the selection of both tree and mesh topology strategies. As mentioned before, the mesh topology can be reduced to a tree topology by reducing the number of routing paths. If for example, a user sets level 5, star topology, he/she is indicating that she/he wants to discard simpler topologies, such as the linear topology, but accepts strategies that make use of more complex topologies, such as tree or mesh.

Visually, large areas represented with large values accept both protocols that work with specific topologies and those that work with multiple topologies. Small values, associated with small areas, only accept specific and increasingly complex topologies, so the number of available strategy results is also reduced.

An example of a time-synchronization protocol that needs a specific topology is the secure time-synchronization model for large-scale IoT networks based on a spanning tree topology [79].

Message Distribution

The message distribution scheme is dependent on the network topology. Message distribution is usually critical in the synchronization protocols, since it is through this distribution of messages that deterministic and non-deterministic pathways appear. This is why many protocols use very specific message distributions.

Communications in the network can be multi-hop, meaning that the messages can go through the network further than the physical distance between two adjacent nodes. If the network allows multi-hop communication, the communication distance

can cover the whole network. If the administrator knows this communication distance and the physical distance between nodes, it is possible to infer whether one-hop or multi-hop communications are necessary.

Table 5.6. Message distribution types associated with the level scale for the message distribution parameter which has five options.

Message Distribution	1 Multi-Hop Broadcast	3 Multi-Hop Point-to-Point	5 Multi-Hop Flooding	7 One Hop Broadcast	9 One Hop Point-to-Point
----------------------	------------------------------	-----------------------------------	-----------------------------	----------------------------	---------------------------------

Most of the methods of time synchronization in wireless networks are based on the sharing of messages between neighboring nodes. However, the administrator must consider what Pottie and Kaiser showed in [13], that the radiofrequency energy required to transmit 1 Kb more than 100 m (i.e., 3 J) is equivalent to the energy required to execute 3 million instructions. This means that how the problem of time synchronization can be tackled is very varied, as each approach can have advantages and disadvantages.

Similar to the network topology parameter, the message distribution parameter is divided into five levels: 1, 3, 5, 7 and 9 (Table 5.6). The message distribution models we value are one-hop (point-to-point and broadcast) and multi-hop (flooding, point-to-point, and broadcast). At level 1, we associate broadcast distribution with multi-hop, that is, the most demanding of all. This type of distribution must control the forwarding of messages and their origin in order not to saturate the network. At level 3, we associate point-to-point communications with multi-hop. Routing is required for this type of distribution. The next level is flooding with multi-hop. For level 7, we have a broadcast distribution with one-hop, and finally, with level 9, we have point-to-point communication with one-hop.

From the user's point of view, if the filter is set to value 5, for example, we are saying that we are looking for synchronization strategies that allow multi-hop and discarding all strategies that are based on a single hop. If a user requires multi-hop point-to-point communications, those strategies that have multi-hop broadcast communications will also work, but not those that have one-hop communications.

Visually, as in previous cases, the smaller the area, the fewer the strategies that will be offered to the user as valid options. An example of a synchronization protocol that needs at least a level 5 is the efficient network flooding and time synchronization with Glossy [68].

Accuracy

As commented in previous sections, every application has specific requirements. The triangle of relationships between the maximum error allowed by the application, the accuracy of the clock and the period between synchronization rounds offer the administrator information of vital importance when introducing a new time-synchronization strategy in the database.

Table 5.7. Values associated with the level scale for the accuracy and precision parameter ranging from 1 to 9.

Accuracy	1 <1 μ s	2 <10 μ s	3 <500 μ s	4 <1ms	5 <10ms	6 <500ms	7 <1s	8 <10s	9 <60s
----------	---------------------	----------------------	-----------------------	---------------	----------------	-----------------	--------------	---------------	---------------

For the parameter of accuracy, we can consult Table 5.7 to know the association between the levels and the demand in accuracy. We have associated with the most demanding level, level 1, a maximum error in time synchronization of less than 1 μ s. While we have associated with the most relaxed level, at level 9, time synchronization errors between 10 and 60 seconds. Given the current state of the art, it has been considered that very few strategies achieve sub-microsecond accuracy at the cost of making use of high resources. This is why the most demanding level has been set at that range. The more relaxed value is for those applications that need to be synchronized, but that do not require much accuracy because they monitor varying parameters very slowly, for example, changes in ambient temperature. As seen in the taxonomy, there are strategies for post-facto synchronization; so this accuracy may be achieved once the measurement is completed with further data processing. For a user who sets his/her selection filter at level 4, i.e., 1-millisecond accuracy, all strategies that are capable of synchronizing with a maximum error of 1 millisecond will be suitable. But if they synchronize with fewer errors, they will also be valid. Visually, the smaller the area, the more demanding the filter, eliminating more strategies. An example of a good precision is those strategies where the MAC-layer timestamping is employed to achieve synchronization accuracy up to microseconds [65], while the Speculative Precision Time Protocol consistently synchronizes with sub-microsecond precision [94].

As with consumption, if the administrator knows the minimum and maximum distance between the network nodes, the message propagation time could be estimated. In this way, it could be known if this time has a significant contribution in comparison with the rest of the components of the communications path.

Clock stability

For the administrator, it is useful to know if the local clocks in the nodes are corrected continuously or if they can run freely (without local corrections). Furthermore, if the clocks are corrected continuously, this correction operation can be done on demand when an event is detected, or it can be done periodically at intervals. In the case of periodic clock corrections, the stability parameter of the clock is useful to estimate of the period in which the corrections should be executed.

Table 5.8 shows the different clock stability ranges that have been associated with the nine levels. We consider a clock to be very stable, and therefore very demanding when its stability is like that of cesium and rubidium clocks, having deviations of less than 10^{-11} . This deviation is equivalent to 864 nanoseconds a day, or 26 μ s a month, and is associated with the value 1. The scale grows and therefore becomes more relaxed until it reaches the value 9, where the clocks have stability

between 100- and 1000-ppm or a deviation of more than 8.64 seconds a day, or more than 259 seconds a month.

Visually, the larger the area, the less strict the parameter is, and the more options are available. As the filter is reduced, the parameter becomes more demanding and the available options will be reduced.

Table 5.8. Values associated with the level scale for the clock stability parameter ranging from 1 to 9.

Clock Stability	1	$<10^{-11}$ 0.01ppb	2	$<10^{-10}$ 0.1ppb	3	$<10^{-9}$ 1ppb	4	$<10^{-8}$ 0.01ppm	5	10^{-7} 0.1ppm	6	10^{-6} 1ppm	7	10^{-5} 10ppm	8	$<10^{-4}$ 100ppm	9	$<10^{-3}$ 1000ppm
--------------------	----------	------------------------	----------	-----------------------	----------	--------------------	----------	-----------------------	----------	---------------------	----------	-------------------	----------	--------------------	----------	----------------------	----------	-----------------------

From the user's point of view, we must consider that both experts and non-experts in the field can use the filtering tool. This is why the questions posed to the user must abstract him/her from the problem. In the case of clock stability, the user may not even know that the clocks have a stability parameter. This is why the question posed to the user is a more abstract question, such as how often can a synchronization round be run? Depending on the answer to this question, the tool assigns the level of the filter.

Examples of synchronization strategies that perform corrections continuously are those that make use of linear regression algorithms such as the RTSP [77].

For the administrator, it would be useful to know if the reference clock is static or not, that is, if it always comes from the same source or if it is a dynamic reference. For example, FTSP [66] achieves robustness to node and link failure by using ad-hoc flooding to propagate synchronization messages and dynamic topology updates. In FTSP, the node with the lowest ID number is dynamically elected/re-elected as the root node and serves as the source of the reference time.

The convergence time of the synchronization strategy is another factor to consider. The greater the time of convergence, the greater the percentage of CPU that is dedicated. Also, the longer it takes to synchronize the network, the longer it takes to perform the synchronized tasks required by the application.

Software abstraction

A wireless sensor node can have several clock sources: internal oscillators, external oscillators, or specific frequency oscillators such as RTCs. The administrator must know the requirements in terms of software abstraction of the synchronization strategy to be incorporated into the database. Some synchronization strategies require techniques that make use of multiple clocks, such as VHT [95], or switch between the working states of the microcontroller by turning off some clocks. If the administrator knows the typology and frequency of the clock sources, he or she can establish whether the clocks are affected by environmental factors or not. The administrator can also estimate the range of accuracy of a clock only by knowing its type. The administrator should know that internal RC clock sources have poor accuracy and that they are very dependent on such factors as the temperature. Clock sources based on crystal oscillators have good accuracy and precision. The accuracy

of the clocks is one of the parameters used to calculate the relationship between the maximum permissible error and the length of the intervals between rounds of time synchronization. Although the clock sources have a certain frequency, the system may work at a higher or lower frequency. The inverse of this frequency gives the administrator what we refer to as a bit time. The bit time sets some resolution limits. For example, the bit time for 32.768 kHz clocks is 30.5 μ s, but for an 8 MHz clock it is 125 ns. Besides, the frequency can be used to estimate power consumption if this has been defined as an important parameter.

Table 5.9. Software abstraction levels associated with the level scale for the software abstraction parameter that has eight options.

Software Abstraction	1 APP OS	2 APP BM	3 OS	4 NET	5 MAC	6 HAL	7 DRI	8 REG
-------------------------	-------------	-------------	------	-------	-------	-------	-------	-------

The level of software abstraction is key to choosing an adequate synchronization strategy. If the software architecture is based on an operating system or, if the nodes work on bare machines, this leads the tool to some solutions or others. An operating system can facilitate the tasks of integration or implementation of the software, but at the same time, the operating system could mask many tasks and executions. In a bare machine system, it is more difficult to implement software, but there is more control over the execution of tasks and the code is better at dealing with the hardware constraints. A system with an operating system is slower when executing code because there is code overload due to the context changes, so there is a trade-off between low-level control and efficiency in code development.

The software abstraction parameter has been divided into eight levels (Table 5.9). The most demanding level is the one that most abstracts the end-user from the specific implementation of the strategy and, therefore, level 1 has been assigned the application software abstraction layer on the operating system. The next level of abstraction would be the application software layer over the bare machine, that is, without an operating system. For level 3, the implementation would need to modify the software on the operating system level. For level 4, the software would need to be modified at the network layer level. At the next level, it would be necessary to touch software at the MAC layer level. At level 7, drivers would need to be modified and, at level 8, the ability to touch system registers would be required. Therefore, a higher level of abstraction may imply that certain strategies cannot be implemented because they belong to lower layers. This organization of layers has been done from the application developer's point of view and should not be confused with the point of view of the developer of the different layers of the protocol stack.

In this way, from a user perspective, use of the filter is defined by up to which layer of software abstraction the user has available to be modified. Users who set a level 5, for example, can modify up to the MAC layer. This excludes all protocols that need to touch the HAL, drivers or registers, but accepts all protocols that are implemented in higher layers, e.g., application. As in all other cases, a smaller filter area limits the number of available solutions.

There are many examples of synchronization strategies that have been implemented on operating systems. For example, with TinyOS [96], one of the factors that allow good results to be obtained in time-synchronization protocols is that packet-level time-synchronization interfaces provided by TinyOS are used to timestamp synchronization messages at MAC layer. Also, TinyOS has the FTSP [66] synchronization protocol integrated. A particular example, among others, of the synchronization algorithm that works with TinyOS, is the Clock Sampling Mutual Network Synchronization (CS-MNS) [97]. However, there are also many protocols implemented on FreeRTOS [98]. In the TDMA timing protocol proposed by Lennval [20], the software timer reloads introduce an error larger than what they want to compensate for; this is due to, among others, real-time OS preemptions. They also highlight that, when making the measurement setups with FreeRTOS, the synchronization uncertainties in the send time are due to delays that depend on implementation details and OS delays. Another example is the proposed MAC protocol by Val [99], which is based on FreeRTOS and has been designed and implemented for real-time SHM systems for trains. Then there are also examples of implementations on other less well known operating systems. The Flopsync scheme designed by Leva [100] was implemented as an application for the Miosix [101] microcontroller operating system; while the time-synchronization strategy behind the Trustful Space-Time Protocol (TSTP) [94] makes use of a cross-layer communication protocol initially developed for the Embedded Parallel Operating System (EPOS) [102].

Hardware abstraction

Finally, the administrator must consider the requirements of the synchronization strategy in terms of the hardware needed for its operation. Concerning the specific requirements of the strategy in these terms, other hardware parameters can affect other parameters to a greater or lesser extent. Therefore, the more knowledge the administrator has when incorporating a synchronization strategy into the database, the better defined its parameters will be. For example, it is interesting to know the types and sizes of the available memories. There could be, among others, volatile or non-volatile memories, flash memories, and RAM; and each of the memory types could have different sizes. Flash memory affects the size of the code that could be implemented, while RAM affects the amount of data that can be stored at runtime. In the case of some synchronization strategies, it could affect the number of offset timestamps that can be stored for skew calculations using such methods as the least-squares regression [103] [104]. There are many other examples where the hardware approach is used to solve, or at least minimize, the problem of time synchronization in WSNs. For example, one of the hardware platforms we have developed is a wireless end device that monitors structures in a synchronized way [91]. In this case, the hardware approach is not based on the system clock source, but on the design of a complete system where the nodes must meet strict time-synchronization requirements. The network synchronization of the system is based on a star topology where a central node emits energy pulses. The receiving nodes record the moment

when they received the energy pulse with a timestamp in the MAC layer. Once the measurement is finished, the data is merged based on the reception timestamped pulses.

Table 5.10. Hardware abstraction levels associated with the level scale for the hardware abstraction parameter which has seven options.

Hardware Abstraction	1	HW-NS Low Res	2	HW-NS High Res	3	Additional OSC	4	External Module	5	TXRX specific	6	MCU specific	7	Platform specific
-------------------------	----------	------------------	----------	-------------------	----------	-------------------	----------	--------------------	----------	------------------	----------	-----------------	----------	----------------------

The highest level of exigency is the one that allows the synchronization strategies to be executed while abstracting the user from the difficulties that the hardware entails. This is why, in level 1, we have associated non-specific hardware with constrained resources. In level 2, we have non-specific with unrestricted resources hardware where high complexity strategies can be executed. In level 3, it is necessary to add external oscillators, such as an external RTC. In level 4, an external module has to be added, for example a GPS module. In level 5, it is necessary to use a specific radio transceiver. In level 6, a specific microcontroller is needed and, finally, a complete specific hardware platform is needed in level 7. A user who sets level 2 in the selection filter indicates that she/he only wants to opt for synchronization strategies that do not require external or specific elements to work. On the other hand, a user who sets level 4 would be willing to include external clocks or modules. The complete list can be found in Table 5.10.

The modulation of the data at the time of transmission and the frequency band of the transceiver fix some characteristics, such as the wavelength, and therefore the transmission range. Considering the frequency band, an estimation of the one-hop communication span can be made. To estimate that communication span, it is also necessary to know the type of antenna (strip, external, ceramic, etc.) and the configured transmission power. An estimation of the scope of communications can be made with these data. The baud rate, the type of modulation to be used for communications and other details about the modulation would also be useful for the administrator. For example, an MSK modulation is needed to use inherent synchronization protocols, such as Glossy [68].

Another important topic is whether the synchronization in the network is done exceptionally or the time-synchronization strategy needs to have the clocks synchronized throughout the lifetime of the application. This will affect the tasks and the percentage of execution time in the microprocessor used by the synchronization strategy. If continuous local corrections are made by the nodes, it may be necessary to do skew calculations, that is, the estimation of the deviation of their local clocks from the reference one. These mathematical calculations may require more complex operations, perhaps with a floating-point, or they may need to store more variables in memory. The hardware platform may also provide the developers with some tools that facilitate the calculations.

Table 5.11. Summary of the values and levels associated with each of the parameters that form the radar chart. Through this table, the relative evaluation of each administrator and user is translated into an absolute evaluation, the same for all.

Administration View					User View													
<i>A time-synchronization strategy is defined here</i>					<i>A user filter is defined here</i>													
<i>Valid values from 0 to 9</i>					<i>Valid values from 1 to 10</i>													
<i>Value 0 represents Not Defined</i>					<i>Value 10 represents Indifferent</i>													
Power Consumption	1	<5μW	2	<50μW	3	<500μW	4	<1mW	5	<5mW	6	<50mW	7	<500mW	8	<1W	9	<5W
Monetary Cost	1	< 1%	2	<15%	3	<30%	4	<40%	5	<50%	6	<60%	7	<75%	8	<90%	9	<100%
Security	1	Very secure. Security as a priority					2	3	4	5	6	7	8	Not secure. No security strategy has been implemented			9	
Network Topology	1	Mesh		3	Tree			5	Star			7	Lineal			9	ALL Valid	
Message Distribution	1	Multi-Hop Broadcast		3	Multi-Hop Point-to-Point			5	Multi-Hop Flooding			7	One Hop Broadcast			9	One Hop Point-to-Point	
Accuracy	1	<1us	2	<10us	3	<500us	4	<1ms	5	<10ms	6	<500ms	7	<1s	8	<10s	9	<60s
Clock Stability	1	<10 ⁻¹¹ 0.01ppb	2	<10 ⁻¹⁰ 0.1ppb	3	<10 ⁻⁹ 1ppb	4	<10 ⁻⁸ 0.01ppm	5	10 ⁻⁷ 0.1ppm	6	10 ⁻⁶ 1ppm	7	10 ⁻⁵ 10ppm	8	<10 ⁻⁴ 100ppm	9	<10 ⁻³ 1000ppm
Software Abstraction	1	APP OS	2	APP BM	3	OS	4	NET	5	MAC	6	HAL	7	DRI	8	REG		
Hardware Abstraction	1	HW-NS Low Res	2	HW-NS High Res	3	Additional OSC	4	External Module	5	TXRX specific	6	MCU specific	7	Platform specific				

As shown in Table 5.11, all parameters are evaluated from 1 to 9 and values 0 and 10 are used to indicate special cases from the administrator and user perspectives. The ranges from 0 to 1 and 9 to 10 have been intentionally left empty to accommodate possible new developments that may occur in the future. In this way, the methodology would continue to be valid in the future, where, for example, synchronization strategies appear with lower power consumption than those we value in the table. These new strategies could be assigned values between 0 and 1, indicating that they consume less. Furthermore, with this growth, we would have a perspective on how the state of the art has evolved in a specific parameter. This capacity to incorporate new solutions in the future would not affect the solutions already incorporated in the database, since all the previous values and ranges are maintained with the same meaning. When incorporating these new ranges of values, at the time they appear, the associated tool will have to be adapted.

5.2.4. Report results

When the methodology is used, a result must always be generated. This result can be satisfactory by offering one or several synchronization strategies that meet the user's requirements, or it can be an unsatisfactory result where no strategy that meets the requirements can be offered. In both cases, the methodology generates a report. Each time the user modifies one of the filtering parameters, the tool calculates the number of possible remaining solutions.

The methodology generates a report in table form that is accompanied by a radar chart. The table shows a list of all the synchronization strategies that comply with the filter defined by the user. This same table is represented graphically through the radar chart. In some cases, depending on the requirements of the application and the decisions of the user, a group of time-synchronization strategies is proposed as a solution. In other cases, when there is a harder dependency between the parameters and the degrees of freedom are reduced, the result of unfeasibility can be reached. In this case, the result is accompanied by the causes and trade-offs that make it unviable.

From the user's point of view, it should be noted that there are dependencies between the parameters evaluated. Therefore, the user must consider that, perhaps, when he/she seeks to maximize some parameters, others must be minimized to obtain viable solutions. It is probable that, when trying to maximize all the parameters, no solutions will be available.

All the possible solutions, hardware and/or software, fulfill a series of trade-offs between the parameters. A radar chart is formed with all these trade-offs that graphically shows the dependency between the parameters and the quantitative value of each parameter. Figure 5.6 shows a graphic representation of how a solution is chosen with the user filter. Figure 5.6 is a specific figure designed to clarify how the tool filters the different solutions available. The application requirements (Figure 5.5) are superimposed over the available solutions in the database (Figure 5.2), giving Figure 5.6 as a result.

Figure 5.6. Graphical representation of how a valid solution is selected. Here the application requirements shape (blue area) is superimposed over all the available solutions in the database. Only the solutions 3 and 4 are valid (grey area) because they are fully contained in the user filter area. The solutions 1 and 2 are with a pointed line and are discarded.

As can be seen, only solutions 3 and 4 are completely contained within the user-defined filter area. The filter is a definition of the application requirements. Therefore, only solutions 3 and 4 are valid in this example case. As the area of solutions 1 and 2 is not completely contained by the filter area, it means that some of its parameters do not meet the requirements imposed and are therefore excluded. The radar chart offers a visual way to know, through areas, which solutions are valid, and which are not. The smaller the area of the radar chart, the more selective and

demanding the requirements to choose a valid synchronization strategy. In contrast, the greater the area of the radar chart, the less restrictive or the greater the freedom when choosing a synchronization strategy. All the viable solutions that pass the filter are included in the chart. Any modification in the input parameters causes a modification in the requirements or on the freedom to choose an appropriate strategy, and therefore the tool generates a different radar chart.

As commented before, all the time-synchronization strategies have previously been included in a database with a value assigned to each of their parameters. In this way, all the synchronization strategies are parameterized and can be represented on a radar chart. When setting a user filter, each parameter value is compared. If the value of the strategy is less than or equal to the value of the filter, it is kept, if it is greater, it is discarded as it does not meet the requirements imposed by the user.

Using this procedure of graphic representation, an optimal solution can be defined as that one which coincides with the perimeter of the filter. That is, we consider the optimal solution to be the one in which the values of its parameters coincide, individually, with the values defined in the filter. Therefore, the solution that coincides with the filter will be an optimal solution for the definition of the needs. At this point, the user may see that other solutions offer better values in certain parameters and therefore change his/her definition of needs, being stricter in those parameters, also making the optimal solution change.

5.3. Time-Synchronization Search Tool (TSST)

The tool helps both the administrator and the user to follow the workflow of the methodology. The tool can be something as simple as a table in a notebook where the new strategies that are incorporated are noted and parameterized. But thanks to new technologies, and the use of computers, this tool can evolve to use an excel table and where the radar chart is also generated. Or it could also be developed on other platforms such as applications for PCs or mobile devices.

We have decided to develop a tool based on web development. Specifically, we have created four web pages in JavaScript [105] and HTML [106], which connect to a web server created in NodeJS [107] and which makes use of a database developed with Cockroach DB [108]. The elements and developments have been chosen so that the web tool is easily accessible, and that the worldwide scientific community can benefit from it. Our tool is released in a [public repository](#) for anyone to download and run for use. In the following sections, we will detail each of the developments.

The tool we have created to go through this methodology allows the evaluation and the introduction of the necessary data in the input blocks. Four web interfaces and a database have been developed. The tool is called Time-Synchronization Search Tool (TSST). The four web interfaces are general index (Figure 5.7), administration form, user form, and associated calculators. One of the forms is used by experts in time-synchronization strategies to introduce each of the strategies in a parametrized way into the database. We call this form the administration TSST (aTSST). The aTSST allows experts to quickly and easily enter new time-synchronization strategies

into the database. In this way, every protocol found in the database is available in the filtering phase. The database provides the dynamism to incorporate, edit or delete strategies, but only from the administrator mode. The user TSST (uTSST) is where the end-user enters their requirements seeking to find a synchronization strategy that suits their needs. This is the web form that abstracts the end-user from the details of the strategies. As a rule, the use of a tool like ours improves the decision time to choose a synchronization strategy. Furthermore, regardless of the user's level of experience in this field, the tool guides them in their decision making so that the result is always a valid one. The web interface with associated calculators helps administrators and end-users to perform calculations related to clock sources or synchronization strategies, for example, by doing deviation calculations and converting ppm to seconds a day.

Figure 5.7. Index page of the Time-Synchronization Search Tool (TSST).

5.3.1. Database

The chosen database is from Cockroach labs [108]. CockroachDB is an open-source distributed Structured Query Language (SQL) database that offers the user many benefits. Its state-of-the-art technology is attractive to businesses of all sizes and draws them in. Once there, the database flexibility and extensive features for database management encourage them to stay. In their own words, this database has been architected for the cloud, allowing them to deliver resilient, consistent, distributed SQL at any scale. This database is scalable and distributed and offers the familiarity and power of SQL with the comfort of your existing Object Relational

Mapping (ORMs). Automated sharding (data fragmentation for scalability) ensures great performance as the database scale. CockroachDB scales horizontally, very quickly, and easily. This scaling capability is unique in that it has this specialized built-in functionality. Although this database is an SQL database, its scaling is considered non-SQL-ish. This database has also been designed for resilience and latency. The database offers a multi-region database that operates with single-region latency and ensures data is always available and a Recovery Point Objective (RPO) of zero. This means that when a failure occurs the data that can be lost is zero.

In our case, we have used a self-hosted distributed SQL for all our testing and development. But the aim is to transform it into a distributed SQL service that the whole scientific community can take advantage of. A key benefit of this database is straightforward horizontal scaling across nodes.

During development and testing, the database was installed on a computer with a Windows operating system. Once installed, a script is executed that starts the database where we assign a listening port and an HTTP command port. Once the database is launched, we obtain an output as shown in Figure 5.8

```
CockroachDB node starting at 2020-01-23 12:15:21.4892889 +0000 UTC (took 4.8s)
build:      CCL v19.1.4 @ 2019/08/06 15:43:11 (go1.11.6)
webui:      http://localhost:6081
sql:        postgresql://root@localhost:6001?sslmode=disable
client flags: bin\cockroach <client cmd> --host=localhost:6001 --insecure
logs:       D:\Documentos\git\phd\nodejs\cockroachdb\toolsDB\db\data\logs
temp dir:   D:\Documentos\git\phd\nodejs\cockroachdb\toolsDB\db\data\cockroach-temp454432911
external I/O path: D:\Documentos\git\phd\nodejs\cockroachdb\toolsDB\db\data\extern
store[0]:   path=D:\Documentos\git\phd\nodejs\cockroachdb\toolsDB\db\data
status:     restarted pre-existing node
clusterID:  dbb771f9-442f-4d57-8926-1492c4d99511
nodeID:     1
```

Figure 5.8. CockroachDB database log after initializing

Once the database server is initialized, we can create a new database document, and within this create a new table. This table is where the synchronization strategies are stored. We have created a table with 22 fields. In addition to the 10 fields presented in Section 5.2.3, we have fields for the internal management of the database and information about the published strategy. The information about the strategy stores data like the title of the strategy, if it has a Digital Object Identifier (DOI) or Uniform Resource Locator (URL) number, date of its publication in the scientific community, name of the journal or congress where it was published, names of the authors, an abstract of the strategy and a field to store a pseudo code. The fields that we have created for the internal management of the database are a unique identifier field, an internal name or name reduced to a single word, a field that saves a timestamp of when the strategy has been inserted in the database and a field that saves the contact information of the administrator who has made the insertion.

This table structure allows administrators to store new time-synchronization strategies while allowing users to filter according to multiple parameters.

5.3.2. Webserver

The server has been created using NodeJS. NodeJS is an asynchronous event-oriented JavaScript runtime environment built with the Chrome V8 JavaScript engine. NodeJS is designed to build scalable networked applications.

The server starts by importing all the necessary packages for its proper functioning. Among others, it imports the HTML package, the URL address package, and the CockroachDB package. Then the code tries to initialize the database. First, a login is made with the corresponding data to have write permissions. Then it is checked that the database document exists, and if not, a new one is created. The same check is made to see if a previous table already exists or if a new one must be created. If there is an error in the database initialization, this error is shown on the server terminal and the server execution is finished. In case the database initialization is satisfactory we can go on to create an HTTP server.

5.3.3. Administration TSST

The administration TSST (aTSST) website is based on two files. One file contains an HTML structure and the other file contains all the functionality of the website developed in JavaScript.

In the HTML file, we define the style of the web page and the places where the different web blocks are found. In this specific case, we are going to define a web form with multiple parameters and several buttons to interact with the web. We create an input block for each of the database fields.

The JavaScript file is the one that contains all the functionality control of the page and where its behavior is defined. To facilitate the work jQWidgets [109] has been used. jQWidgets is a software framework with widgets, themes, input validation, drag and drop plug-ins, data adapters, internationalization, and many other advantages. It is built on open standards and HTML5, CSS, JavaScript, and jQuery technologies.

The administration website can be divided into three sections. The first section collects information about the details of the time-synchronization strategy. The second section is where the synchronization strategy is parameterized. The third section is where the results are shown in table mode, and in a graphic way, using a radar chart.

Figure 5.9 shows the first section of the aTSST. In this section, the mandatory fields are highlighted in red. These mandatory fields are the minimum fields that the administrator must complete to include a new strategy in the database. At the beginning of this first section, there are brief instructions for new administrators.

Administrator TSST

This is the Administration Time-Synchronization Search Tool

In this section a WSN synchronization expert, from now the administrator, is responsible for evaluating a new synchronization strategy.

The administrator must parameterize the new strategy and assign a value to each of the parameters following the table below.

The database of possible solutions grows with every new incorporation.

Time-synchronization strategies are defined based on **nine numerical values** representing nine parameters.

The parameters can use values from 0 to 10.

We reserve the **value of 0 to indicate that the parameter has not been defined** and therefore no value has been assigned to it.

For parameters that have been evaluated, values from 1 to 10 are used.

Administrator Email

admin@email.com

Tag Name

ShortName

Title

The full strategy title

Authors

Author1, Author2,

DOI

10.1000/xyz123

URL

http://www.myaddresss.com

Publication Date

Thursday, February 27, 2020

Publisher / Journal / Conference

Journal name

Abstract

Abstract information

Pseudocode

Pseudocode

Figure 5.9. Display of the first section of the aTSST where they ask about publication details of the synchronization strategy.

Figure 5.10 shows the second section of the aTSST. This section is where the administrator should parameterize the synchronization strategy. All fields in this section are mandatory. The target or the set of targets that the strategy pursues must be indicated. Besides, a value must be assigned to each of the nine parameters. All parameters have a numerical value between 0 and 9. A value of 0 indicates that the parameter has not been defined. To make it easier to assign values to the parameters, a slider has been included for each parameter. Each slider has been configured so that only valid values can be chosen according to Table 5.11. This way, all parameters go from 0 to 1 without intermediate values. There are parameters such as software or hardware abstraction whose maximum values are 8 and 7 respectively. Moreover, in these parameters, as with the network topology and message distribution parameters, their values are discrete and therefore the slider only allows certain values in each case. In the rest of the parameters, the values in the slider have been configured to allow values with jumps of 0.25 units between 1 and 9. Also, the

tooltips shown when moving each slider show the values to which each scale corresponds (Table 5.11). In this way, when moving the slider, the limit or real value that is being set is always shown. A small button with the "+" symbol has also been added to each parameter description. Pressing the "+" button displays a popover with detailed explanations on how to interpret and evaluate each parameter.

Target

Choose (one or several) the synchronization target that best suits the needs of your application. +

☐ Data Fusion☐ Cooperative Transmissions☐ MAC Schemes

Power Consumption

Choose the value that best fits the Power Consumption needs. Set any value between 1 and 9, or 0 if you do not want to rate this parameter. +

09

<50mW

09

Monetary Cost

Choose the value that best fits the Monetary Cost needs. Set any value between 1 and 9, or 0 if you do not want to rate this parameter. +

09

09

Security

Choose the value that best fits the Security needs. Set any value between 1 and 9, or 0 if you do not want to rate this parameter. +

09

09

Network Topology

Choose the value that best fits the Network Topology needs. There are only five options, with values 1, 3, 5, 7 and 9, or 0 if you do not want to rate this parameter. +

09

09

Message Distribution

Choose the value that best fits the Message Distribution needs. There are only five options, with values 1, 3, 5, 7 and 9, or 0 if you do not want to rate this parameter. +

09

Multi-Hop Flooding

09

Accuracy

Choose the value that best fits the Accuracy needs. Set any value between 1 and 9, or 0 if you do not want to rate this parameter. +

09

09

Clock Stability

Choose the value that best fits the Clock Stability needs. Set any value between 1 and 9, or 0 if you do not want to rate this parameter. +

09

09

Software Abstraction

Choose the value that best fits the Software Abstraction needs. Set any value between 1 and 8, or 0 if you do not want to rate this parameter. +

08

08

Hardware Abstraction

Choose the value that best fits the Software Abstraction needs. Set any value between 1 and 7, or 0 if you do not want to rate this parameter. +

07

07

Save

Search

Clean

Delete

Back

Administration TSST

This work is licensed under a [Creative Commons Attribution 4.0 International License](#).

Figure 5.10. Display of the second section of the aTSST where the target is selected with checkboxes and the value of the parameters through sliders. The user interaction buttons are also shown.

At the bottom of the second section of the aTSST there are four buttons: save, search, clean and delete. The buttons are disabled when you first enter the page. Every time any of the form fields are filled in, they are validated. The content of the field is validated, and it is checked if any of the buttons should be enabled or disabled when filling in any field. For example, the search button is enabled when the administrator selects one or several targets. If after selecting a target we click on the

search button, a search is performed in the database of all strategies that match the selected target.

After a search, the results appear in a table, as seen in the third section of aTSST shown in Figure 5.11. The table shows a reduced number of the saved parameters to make their display more user-friendly. The internal name, the title of the publication, the target and the values of the nine parameters are kept in the table. If the administrator clicks on any of the synchronization strategies that appear in the table, its radar chart is shown below the table.

The clean button is enabled every time one of the input fields has been filled in. This button is used to clear the whole form and the lower section where the results table and the radar chart are shown. By pressing this button, the whole page is cleaned.

The save button, as its name indicates, serves to save the data entered in the form in the database. When a strategy is saved in the database it is associated with a unique identifier number.

Figure 5.11. Display of the third section of the aTSST where when interacting with the buttons a table with the saved strategies is shown. Clicking on one of the strategies in the table displays its associated radar chart showing the value of its parameters.

When a search is made and one of the time-synchronization strategies is selected, in addition to painting its radar chart, the form data is filled with the data saved in the database. Besides, the save button changes its text to “Edit”, indicating that if a modification of the form is made, this button can be pressed to save the changes. This way, any strategy saved in the database can be edited later.

Finally, the delete button is a button that is enabled when selecting a strategy from the table. If we click on the delete button, this strategy is deleted from the database table.

As additional details the table is dynamic, so the administrator can change the size of the columns or can sort the strategies shown by any of those columns. In this way, the administrator can sort the display as he or she likes best. Also, parameter values that were assigned as 0 in the table are represented as Not Defined (N/D) making it easier to identify those parameters.

5.3.4. User TSST

The user TSST (uTSST) is thought and designed so that any WSN application developer, with much or little experience in the field of time synchronization, can complete it without difficulties. The most direct and effective way to know the requirements and needs of the application regarding time synchronization is to ask the final users, the application developers.

The user page has the same structure as the administration page as can be seen in Figure 5.12. This page is composed of two files, one containing the HTML structure and the other containing the behavior defined in JavaScript.

In this case, all fields are mandatory, both the target and the values of the nine parameters. Unlike in the aTSST case where the administrator could set several targets, in this case, the user can select a single target to perform the filtering. In uTSST, the number of values that are necessary to perform the strategy search is also implemented using sliders. The reader can see a difference at first sight between aTSST and uTSST looking at the limits of the sliders. In the aTSST, the general limits were between 0 and 9, while in the uTSST the values range from 1 to 10. For a user, the value 10 represents that this parameter has no importance in the search as commented in Section 5.2.2. In the case of the uTSST, the default value of the sliders when loading the page is 10.

In this web, we have only included two buttons: search and clean. The clean button has the same behavior as in aTSST, it cleans the contents of the forms and the page.

The search button is disabled until a target is selected. Once a target is selected the user can press the search button, which makes a search request in the database. The search criteria are the selected target and all those strategies whose parameter values are lower than the variables entered in the form. In Figure 5.13 the reader can see the results of a search where values have been assigned to the parameters and the data merger target has been chosen. The results are shown both in table mode and in graphical mode via the radar chart. In the uTSST, there are some differences from the aTSST. In this case, when you click on a strategy in the table its information is

displayed for the user to consult. The radar chart shows in a single graph all the viable solutions that meet the requirements imposed. That is all the solutions that are kept within the imposed filter appear in the graph. The filter is represented in the graph as a blue line with a semi-transparent blue filling. Each of the possible solutions appears as a colored line. If the number of possible solutions is too large, all the lines overlap and the display of the results on the radar chart is confusing. That is why the radar chart is interactive, allowing the user to activate and deactivate lines, or to highlight the line that interests the user most, as shown in Figure 5.14.

User TSST

This is the User Time-Synchronization Search Tool

From the user's point of view, the aim is to create a filter to reduce the number of valid options for a specific case. There is a direct relationship between the requirements and the number of available solutions. If, on the one hand, due to the requirements introduced in the input forms, the filter is too selective and restrictive (smaller values and smaller area in the chart), it would be possible that no solutions meet all the desired parameters. If, on the other hand, few details are provided, the selection filter is too flexible and tolerant (bigger values and bigger area in the chart), and therefore, a large number of possible solutions are offered, many of which are very general and poorly adjusted to a specific problem.

To define the parameters the user can use values from 1 to 10. The value 10 indicates that the parameter is unimportant in this specific search. The acronym N/D in the table of results indicates that the parameter has not been defined and therefore no value has been assigned to it.

Target

Choose the synchronization target that best suits the needs of your application.

☐ Data Fusion

☐ Cooperative Transmissions

☐ MAC Schemes

Power Consumption

Choose the value that best fits the Power Consumption needs. Set any value between 1 and 9, or 10 if you do not want to rate this parameter.

1

10

1

10

Monetary Cost

Choose the value that best fits the Monetary Cost needs. Set any value between 1 and 9, or 10 if you do not want to rate this parameter.

1

10

1

10

Security

Choose the value that best fits the Security needs. Set any value between 1 and 9, or 10 if you do not want to rate this parameter.

1

10

1

10

Network Topology

Choose the value that best fits the Network Topology needs. There are only five options, with values 1, 3, 5, 7 and 9, or 10 if you do not want to rate this parameter.

1

10

1

10

Message Distribution

Choose the value that best fits the Message Distribution needs. There are only five options, with values 1, 3, 5, 7 and 9, or 10 if you do not want to rate this parameter.

1

10

1

10

Accuracy

Choose the value that best fits the Accuracy needs. Set any value between 1 and 9, or 10 if you do not want to rate this parameter.

1

10

1

10

Clock Stability

Choose the value that best fits the Clock Stability needs. Set any value between 1 and 9, or 10 if you do not want to rate this parameter.

1

10

1

10

Software Abstraction

Choose the value that best fits the Software Abstraction needs. Set any value between 1 and 8, or 10 if you do not want to rate this parameter.

1

10

1

10

Hardware Abstraction

Choose the value that best fits the Software Abstraction needs. Set any value between 1 and 7, or 10 if you do not want to rate this parameter.

1

10

1

10

Figure 5.12. The user interface of the uTSST

The uTSST forces the user to follow the evaluation blocks of the methodology. This ensures that any user, regardless of their experience in the field of WSN time synchronization, meets the evaluation requirements.

Figure 5.13. The output result of user TSST where strategies are shown as colored lines that comply with the filter represented in a semi-transparent blue fill.

Once the radar chart has been generated with all possible solutions, there are several possibilities before finalizing the methodology cycle.

- The developers can directly select one of the possible solutions offered by the tool.
- When the time-synchronization strategies are inserted in the database, some of them have associated pseudocode. If any of them are included in the set of available solutions, the pseudocode is also offered to the developers to facilitate the implementation task. An example of pseudocode can be seen in Figure 5.15.

Figure 5.14. The output result of user TSST where one strategy is highlighted and four are disabled.

Algorithm 1 Initialization for Time Synchronization

```

1: if the node is root node then
2:   nodeType  $\leftarrow$  BN
3:   broadcast  $\leftarrow -1, ID, Init, Null, -1, -1, 0 >$ 
4: else
5:   nodeType  $\leftarrow$  UN
6: end if
7: myLevel  $\leftarrow$  0
8: isFatherAtk  $\leftarrow$  false

```

Figure 5.15. Initialization pseudo-code for time synchronization of the secure time synchronization protocol against fake timestamps for Large-Scale IoT [79]

5.3.5. Display of the results

The display of the results, as mentioned above, is done through a table and a graph. The table is created using one of the jQueryWidgets modules. The module is a lightweight jQuery widget that represents data in a grid-like structure. This table supports the multi-column display of hierarchical data, data paging, sorting and filtering, data editing, columns resizing, fixed columns, conditional formatting, aggregates, and rows selection. It can read and display the data from data sources like XML, JSON, Array, CSV or TSV. Therefore, it offers multiple alternatives for table mode representation, organizing the data dynamically and facilitating the interaction of the administrator and user.

For the graphic representation, we have made use of Highcharts [110]. Highcharts is a software library for graphics written in pure JavaScript, offering an easy way of adding interactive charts to a web site or web application. Within the wide range of possibilities of Highcharts, we have made use of polar data representation. From there we have defined the rest of the details, such as the number of variables to represent, the number of divisions to show, or if the data represented has a fill or just stroke. Besides, as it is an interactive graphic, the user can see the parameter values in a pop-up when moving the mouse over the different represented strategies. Besides, in the legend, the representation of the different strategies can be activated or deactivated, or simply highlighted as shown in Figure 5.14.

As commented in the uTSST, the radar chart is interactive. For example, when the administrator or user places the mouse over the graph, it shows the value of each of the parameters. The radar chart also has a sub-menu to display the chart on full screen, to download the chart in image format, or to download it in table format. Besides, the user can interact with the graph by activating, deactivating or highlighting each of the data series represented.

5.3.6. Associated tools

Along with the main tool (TSST), other small tools have been developed that have been of great help to us in the development of the validation. Three types of calculators have been developed. As in the previous tools, this set of calculators has been developed as an integrated website where the user can make calculations online and the results are offered instantly.

- In the clock deviation calculator (Figure 5.16), the time deviation (seconds per day and minutes per year) can be calculated given the quality of a clock in ppm. By entering any of the three parameters, the other two can be calculated. This calculator also allows the user to add a temperature coefficient to the clock source and therefore calculates the additional ppm, sec/day and min/year that the clock could deviate as a function of the temperature. Finally, this calculator gives the accumulated values of the clock deviations by adding the two previous calculations.

Clock deviation calculator

The calculator expresses the deviations in ppm, seconds/day and minutes/year. It also has a calculator to know the deviation of the clock source associated with temperature changes.

A) Clock quality deviation

20

ppm

1,728

Seconds/Day

10,512

Minutes/Year

Calculate

Calculate

Calculate

B) Temperature dependent calculations

Enter Operating Temperature:

20

°C

Crystal Frequency Error

Error (ppm) =

0,042

$\times (25 - \text{OperTemp})^2 =$

1,05

ppm

0,09

Seconds/Day

0,55

Minutes/Year

Calculate

Accumulated deviation: A) + B)

21,05

ppm

1,818

Seconds/Day

11,062

Minutes/Year

Figure 5.16. A calculator that converts the quality of a clock expressed in ppm into time deviations and can also add a deviation term according to a temperature coefficient.

- The ppm conversion calculator (Figure 5.17) is a calculator that allows conversion between different units: decimal, percentage, permille, ppm, ppb and parts per trillion (ppt). Entering a value in any input field directly converts it to all the others.

PPM converter

Decimal, percent, permille, ppm, ppb, ppt conversion calculator
Enter proportion part in one of the text boxed and press the convert button

Enter decimal:

0,00001

Convert

Enter percent:

0,001

Convert

Enter permille:

0,01

Convert

Enter ppm:

10,0

Convert

Enter ppb:

10000

Convert

Enter ppt:

10000000

Convert

Figure 5.17. A calculator that converts between multiple units: decimal values, percentage, per miles, ppm, ppb, and ppt.

- The calculator Clock quality vs Error tradeoff (Figure 5.18) calculates dynamically what is presented in Table 6.6. Thus, by entering two values between the maximum allowed synchronization error in an application, the

quality of a used clock source or the intervals between synchronization rounds, the third value is calculated.

Clock quality vs Error tradeoff

Relation between minimum oscillator quality, maximum synchronization error allowed and maximum round interval between synchronization rounds.

Clock quality:	<input type="text" value="20"/>	ppm	Calculate
Max time synchronization error allowed:	<input type="text" value="1"/>	seconds	Calculate
Round interval:	<input type="text" value="50000"/>	seconds	Calculate

Associated calculators for TSST

This work is licensed under a [Creative Commons Attribution 4.0 International License](#).

Figure 5.18. Development that dynamically calculates the compromise between the maximum allowed synchronization error in an application, the quality of a clock source and the intervals between synchronization rounds.

Chapter 6

Validation

Contents

6.1. The methodology of the validation	101
6.2. Method validation	103
6.3. Methodology validation	144
6.4. Conclusion	153

*Una vez descartado lo imposible,
lo que queda, por improbable que parezca,
debe ser la verdad*

Arthur Conan Doyle

Throughout this chapter, we validate the developments of the methodology. We have carried out the validation of the methodology and the validation of the method. For the validation of the methodology, a group of test users has used it and we have collected all their comments and answers in a form. To carry out the method validation we make use of several use cases. We present several hardware platforms that we have developed during this thesis to be able to implement and execute experiments and synchronization strategies in real environments. We validate the methodology and the associated tool, but we also implement the proposed strategies in real applications. We show the real tests carried out in the different scenarios. We conclude the chapter corroborating with results the correct functionality of the developed methodology.

6.1. The methodology of the validation

To check that our developments, both the methodology and the tools, comply with the proposed objectives, we have carried out validation of the methodology and the method.

To validate the methodology, we have used a group of expert WSNs test users who have been asked to use the methodology and the tool. Each of the users has

worked independently and we have collected and classified all their answers. Also, we have asked them to answer a form to evaluate their previous knowledge about WSNs and time synchronization and their level of satisfaction with the application of the methodology and the use of the tool. Finally, thanks to the use made by these users, and based on their comments, a series of improvements have been implemented in the tool to make it more user-friendly and simpler to use and understand.

For the validation of the method, we have employed use cases. Specifically, two use cases have been conducted for two applications with different requirements but with the same target, data fusion. The number of applications with the target of data fusion is higher than those developments that pursue one of the other objectives: cooperative transmissions or MAC schemes. The applications where data fusion is required have grown exponentially with the arrival of IoT and Industry 4.0, where sensors of the same type or a different nature cooperate to obtain a complete image of the environment under monitoring. Both use cases require a correct time synchronization to carry out a proper data fusion.

The verification and validation presented in this chapter focus on the process of choosing one, or a group, of time-synchronization strategies that meet the requirements of each of the proposed applications and implementing it in a real application environment.

Verification and validation are independent procedures that are used together for checking that a system meets requirements and specifications and that it fulfills its intended purpose. Verification is the action of checking that what we say we are doing we are doing. The verification task is carried out in the development phase and provides objective confirmation that the results meet the design requirements. Validation confirms that what we are doing is being done right and that the intended goal is being achieved.

The validation task is performed once the system has been developed and is therefore after verification. With the validation the result is evaluated, it is checked that it complies with what was initially agreed and that it meets the requirements of the end-user. Validation, as its name suggests, serves to validate the correct functioning of a development. In this way, its validity is assured. Through validation, the correspondence between the initial descriptions and the model is checked to see if it responds to the initial approach. Validation is a necessary phase, otherwise, there is a risk of implementing a bad specification, or of implementing a development that does not meet the needs. The validation should verify that different users with the same requirements should get the same set of answers. That is why the two use cases developed in this chapter have been brought to the attention of a group of researchers who work with WSNs. This group, individually, has been provided with the same assumptions for each of the cases and their responses and behavior have been evaluated with the uTSST.

The methodology, the user interface, the administration interface and the functionality of all of them have been validated through use cases. For web development, it can be said that both the frontend and the backend have been validated.

Besides, to complete the validation, a "tour session" has been carried out. In this session, the methodology and associated tools are reviewed in detail, allowing other researchers to express their opinions on it. This allows future users to understand the meaning of each requirement and express their agreement or disagreement with them. Making use of all constructive criticism, the final published version has been developed.

As commented before, for this validation we are going to make use of two use cases. For each of the use cases, we have proposed an application or scenario where we use a different hardware platform. With this environment, we set up and develop a series of tests from which we obtain results. The results we obtain must validate the correct functioning of the methodology and the tool for choosing a time-synchronization strategy. But also, the application must work correctly with the implementation of the selected time-synchronization strategy.

With these two use cases, we can validate the methodology (through the test group's answers to the questionnaires) and the method (through use cases).

6.2. Method validation

To validate the correct functioning of the methodology, the use case method has been used. This method is a "*way of using*" the system, usually described through a set of "*typical uses*". A use case describes how an actor uses a system to achieve a goal, and what the system does to help him. This method tells how the system and the interaction with it collaborate to produce a value output, complete use of the system. This approach is a practical way to functionally evaluate both the methodology and the tool presented in Chapter 5. A use case not only helps to validate the functioning of the entire system but in many cases helps to understand and clarify the functioning of the process.

One of the proposed use cases has relaxed requirements in the parameters and the other has more demanding parameters. In this way, we show how the methodology works in different circumstances and applications giving an appropriate solution in every case.

The first application on which we have worked is the application that motivated the development of this thesis: SHM. For this application, we developed an ad-hoc hardware platform called the "SHM platform". With this hardware, numerous tests were carried out in different structures, but the main one was a Spanish road bridge.

The second application is an environmental monitoring application. A modular ad-hoc hardware platform has been developed for this application with the name of the "Multi-purpose HW Platform". We have carried out multiple tests checking the influence of temperature on the deviation of the clock sources. The results of the behavior of different types of clocks are presented.

These use cases have been chosen because they are representative of different degrees of severity in terms of requirements. An undemanding time-synchronization is required in environmental monitoring applications. In this kind of application, accuracy errors in the range of seconds in the data fusion do not have a great influence

on the result. From this application, it should be noted that it is carried out in large outdoor spaces where temperature variations between day and night and different times of the year are continuous. Accuracy errors in the range of milliseconds in the time-synchronization of the data fusion do have an important and direct influence on the results in SHM applications. In this specific use case, we go into how synchronization errors affect SHM results.

The use case analysis has been presented in a structured manner: application introduction, hardware platform, assumptions, methodology, tests, and results as can be seen in Figure 6.1.

Figure 6.1. Flow diagram showing the analysis structure of each use case.

In each case, we introduce the application followed by a subsection about the hardware platform developed and used. In the subsection of assumptions, the situation in which the application is developed has been raised. In this subsection, it is possible to introduce additional restrictions to those imposed by the application, such as restrictions imposed by the user or the environment. The methodology subsection has been divided into two phases. First, the flow of the methodology diagram presented in Section 5.2 is used. As a second phase, the tool presented in Section 5.3 is executed, where values have been assigned to the different parameters and a set of valid synchronization strategies have been obtained that meet the application requirements. The time-synchronization strategy is then implemented in the hardware platform and some test is carried out obtaining their corresponding results.

6.2.1. Structural Health Monitoring application

The objective of modal analysis in structural mechanics is to determine the natural frequencies and modes of vibration of an object or structure during free vibration. We can describe the dynamic behavior of any linear mechanical structure in terms of parameters that describe its structural resonance. These modal parameters are the resonance frequency, the damping and the vibration pattern (modal form) of the resonance. The mathematical model based on these parameters is a linear model that gives a complete description of the linear behavior of the structure. An

Operational Modal Analysis (OMA) is then a type of experimental method that determines the characteristics of a dynamic system and defines the dynamic model of a structure.

Structural Health Monitoring (SHM) refers to the process of implementing a damage detection and characterization strategy for engineering structures. In this framework, the damage is defined as changes to the material and/or properties of a structural system. The SHM process involves the observation of a system over time using periodically sampled response measurements from an array of sensors (often inertial accelerometers). For an SHM wireless system, all sensors collecting acceleration samples must be correctly synchronized. If any of the data loses synchronization it may be interpreted as damage to the structure, when in fact it is a failure in the process of merging the data. That is why in this use case, following the methodology, the tool needs to find a viable solution for a demanding application regarding time synchronization. In this use case, most of the input forms in the tool are filled with details because the requirements imposed by the application and scenario are rigorous.

The tests and results performed in this use case are divided into two. In the first instance, a theoretical study is made of how synchronization errors affect the frequencies and modal forms of structures. Then this situation is simulated on a beam where, in a controlled way, synchronization errors are introduced. After that, fieldwork is done on a bridge of a Spanish road using the developed hardware.

Hardware platform

The first developed hardware platform was aimed for the structural health monitoring of bridges [91]. To carry out the application and collect the necessary data, both the hardware platform and the corresponding software were developed in-house.

This hardware platform was designed, implemented and tested by us. The hardware architecture is shown in Figure 6.2 left. The internal modules that compose the system are the power module, acquisition module, wireless communication module, synchronization module, control module, and manager module. The designed system is composed of two types of devices: the server and the clients. Clients have power, acquisition, wireless communication, control, and synchronization modules. The server consists of wireless communication, synchronization and manager modules. In Figure 6.2 right is shown a diagram of the scenario where the hardware is going to be used. As can be seen, on a structure such as a road bridge, the measurement nodes are placed on the structure to detect the vibrations of the structure.

Figure 6.2. Block diagram of the developed hardware platform (left) and a simplified diagram of a real measurement scenario on a road bridge (right).

To know the details of each one of the modules, the reader is directed to consult the publication [91] where detailed information can be found. In this section, we only focus on the development of the time-synchronization module.

In Figure 6.3 we are showing four representative images of a real scenario where vertical vibrations are being monitored on a bridge located on a Spanish road. From left to right and from top to bottom:

- a) The image shows an overview of the location of multiple nodes and their corresponding accelerometers.
- b) Image of one of the clients used to record the vibrations of the road over the bridge.
- c) One of the accelerometers used to record the accelerations. These models of the accelerometer are high precision with a broadband resolution of 0.000001g rms and a sensitivity of 10.0 V/g.
- d) The server consists of a laptop computer where the wireless communications module is connected to send commands and energy pulses to the clients and receive the collected data from them.

Figure 6.3. From left to right and from top to bottom. a) Image of the tests made on a road in a bridge in Spain. b) Image of one of the clients used to record the vibrations of the road over the bridge. c) One of the accelerometers used to record the accelerations. d) Server with a wireless module used to send commands and synchronization pulses to the clients.

As commented before, the system is based on two components: a personal computer where the data is merged and which act as a server and the monitoring nodes which act as clients. The server has the hardware and software for network configuration, the transmission of measurement parameters, the transmission of synchronization pulses, and data reception. Clients have the software necessary to understand the measurement instructions, the data acquisition card, and the appropriate sensors (between one to four per node) to measure.

As discussed in previous chapters, when a developer needs a time-synchronization protocol and does not have a methodology, there are two options: to make use of an existing protocol that may not be adapted to the needs of the application or to develop one of their own. In our case, when this application was carried out, we did not have any methodology and we decided to develop our own time-synchronization protocol. Moreover, when we detected this need, we took advantage of it and decided to start developing this thesis.

The published work [111] shows an analysis of how errors between 30 μ s and 1 ms affect the detection of different modal forms. Normally in SHM applications over bridges, the frequencies and modal forms to be detected are below 25 Hz. As the

objective of this application is data fusion applied to SHM, for a target-sampling rate of 50 Hz, a total jitter of 1 millisecond or 5% of the sampling interval is selected as the gap on total jitter [112]. If this limit is exceeded, the results will be compromised, always considering that synchronization errors have a greater effect when the frequency is higher. For example, a 1 ms synchronization error between two measured acceleration responses will result in 3.6° error in phase angle at 10 Hz and 36° error at 100 Hz [113].

The proposal that was made for the wireless time synchronization was based on the standard protocol IEEE 802.15.4 to meet the time requirements. A master device inside the server sent energy pulses to all clients on the network distributed through a star topology. The synchronization modules in the clients had the feature to detect energy in the 2.4 GHz band and acted upon detection. The transceiver used was a Texas Instruments CC2420.

For the implementation of the time-synchronization strategy, the main challenge in this platform was the software stack layer overhead. Function calls could generate non-deterministic delays from the detection of the energy pulse in the physical layer until that warning reaches the application layer. Our solution was a direct call from the physical layer to the application layer, making the timestamp in the MAC layer, avoiding all the intermediate stack layers. In this way, the client module generates an interruption at the exact moment that it detects energy in its operation band. The interruption is attended immediately in the PIC32 microcontroller, and a couple of time reference-number of samples are stored. We used a 32-bit timer to get this reference, which is activated in the first pulse. The 32-bit timers allow us to detect differences in the signal under 1 nanosecond. By removing the different layers of protocol communication, and making the timestamping in the MAC layer, the energy pulse that marks the synchronization is much faster and does not require any additional processing.

Synchronization pulses could be transmitted to the clients at any time to set sync marks that would be used in the post process. This process corrects small deviations between internal oscillator-based hardware clocks. This way, the system offers full flexibility to create synchronization marks at any moment or in any setup. These pulses could be used to synchronize the clients before placing them over the structure. Therefore, the system allowed for the measurement of structures that are thousands of meters in size. We also included a 9-dBi external antenna which increased the coverage of the synchronization module. More sync marks could be requested during the measurement process to increase accuracy using a simple postprocessing algorithm with decimation and interpolation operations.

The first pulse synchronizes the activation of all ADCs in the boxes. That makes all the ADCs start their measurements with a minimum clock error. Tests carried out in the laboratory assure that the difference between receiving pulses in the nodes is under 120 ns. With all those pulses, the system can fix the minimum derivations in the signals provoked by differences in the frequency clock of the clients which have an accuracy of 25-ppm.

Finally, when the measurement of one setup finalizes, the clients send the couples' samples-timer number to the server where they are analyzed. Because of the

oversampling and the sync marks, the server can process the data getting the synchronization accuracy pursued. Sync marks are sent by the server; thus, an exact time between marks is known. The frequency rate is fixed and higher than the system's needs. Therefore, two files of different clients should have the same number of samples between marks. If a little jitter is detected, the server can interpolate or decimate samples to avoid that jitter.

Following our own taxonomy, this protocol is included within the protocols with a data fusion objective. No security has been implemented in its development. It is an energy-efficient protocol as it only needs to store a timestamp after an interruption. Therefore, it is a protocol with low complexity. The hardware requirements it has only require a radio transceiver that can act after detecting energy pulses. It has no clock correction and therefore falls within the post-facto protocols. The protocol makes use of a star network topology, with timestamping at the MAC layer and without a specific message distribution.

To summarize, the sync module provided a great initial reference to the system, with less than 125-ns difference between all clients. The next pulses that the clients receive are used to fix derivations in the frequency clock. Using that approach, we obtain good first initial pulse synchronization and enough information to correct the internal deviation.

Assumptions

The tool faces an application for structural health monitoring. The time synchronization requirement in these applications is strict and requires a maximum deviation between the clocks in the network of 1 millisecond [114] [115]. However, the more precise it is, the better the results are. In these applications, the main task is the acquisition of accelerations and therefore the task of synchronization, cannot, under any circumstances, interrupt the acquisition.

In this use case, it is intended to monitor a road bridge. The system must be portable to be installed in different types of bridges and it should be battery-powered to allow its deployment in different scenarios. In this case, we have a star network topology, and we make use of a specific hardware platform.

The monitoring of a structure, such as a bridge, is usually carried out with multiple measurements lasting between 10 and 20 minutes. A full set of measurements can take up to 8 hours. In the end, the total set of measurements can then be merged to obtain complete information about the structure or to compare its behavior at different times.

Methodology

The objective of this use case is data fusion with a demanding time-synchronization accuracy. The power consumption is a parameter (value of 8) that has relative importance. The WSN end devices used in this application are battery-powered but they can be charged every day. The batteries used by the nodes in this application must last at least 8 hours, one working day. The monetary cost (value of 8) depends only on the number of end devices needed for the specific

scenario. The network size and the localization of the scenario are variable. The scope of time synchronization extends to all nodes that collect samples since the desynchronization of only one of the nodes affects the overall outcome. In the application, there are no special requirements regarding security because this is a mobile system and it is considered that no attacks are going to occur (value 10). The network topology (value of 7) have been previously defined. In this case, there is a specific network topology, a star topology, to use. Regarding message distribution, although it is not previously defined it must work correctly in the star topology (value of 7). As commented in the assumptions, the maximum error allowed for this application is 1 millisecond. That makes compulsory the use of a precise and accurate (value of 4) time-synchronization strategy where the period of synchronization rounds depends on the quality of the clock sources used. In this case, the stability parameter (value of 8) loses some importance since multiple measurements of about 10 or 20 minutes are made. In these periods, even if the clock sources are not very stable, their deviation is not noticeable. Considering the requirements imposed by the application, and the requirements imposed by the user makes the complexity of the time-synchronization strategy increase. Parameters as the convergence or the use of static reference nodes to distribute the reference time are not defined. But the clock sources to use are the ones available in the specific hardware platform imposed by the user (value of 7). The synchronization messages must be timestamped in the MAC layer to get accurate results and that is the reason for a low level of software abstraction. The memory available, the hardware platform and the transceiver for the radio communications are specific and have been previously defined by the user (value of 8). But the hardware platform should not be a limitation because it has the computing capacity necessary to carry out the necessary calculations of most of the protocols that exist to date. All the parameter values are summarized in Table 6.1.

Table 6.1. Summary of the values assigned to the nine parameters of the methodology for a specific application of Structural Health Monitoring

Power Consumption	8	Monetary Cost	8	Security	10
Network Topology	7	Message Distribution	7	Accuracy	4
Clock Stability	8	SW Abstraction	8	HW Abstraction	7

With the values that we, as end-user, have set to each of the parameters, we can make use of the tool and set a filter as shown in Figure 6.4. The area, in this case, shows how the most demanding parameter is precision, and the parameter that is indifferent to the user is security. The other parameters make the area not small indicating, that many of those protocols that comply with the accuracy parameter will be valid for this application. On the left side of Figure 6.4, we can see the real data from many of the wireless sensors that took the samples in a synchronized way even though they were spatially distributed throughout the structure.

Figure 6.4. Radar chart generated based on the input form for a specific Structural Health Monitoring application. The most demanding parameter is precision with a value of 4 and the most relaxed is the security with a value of 10.

In this case, the report generated by the tool, containing the radar chart with the viable solutions, is shown in Figure 6.5. As a result, those synchronization protocols that keep the time-synchronization error below 1 millisecond would be offered.

Figure 6.5. User selection tool showing a table with all the viable time-synchronization that fulfills the user filter and a graphical representation with a radar chart.

As mentioned above, when this application was carried out, and when the synchronization protocol was developed, the methodology did not yet exist. But once

the methodology was developed, this protocol was incorporated into the tool selection database. Given the requirements imposed, the tool, as can be seen, offers our own development as one of the possible solutions to be implemented (the “PulseStar”), and is the one that we select to carry out the next test.

Test

As already mentioned, we will first see the simulations. Then we will analyze what we call Practical case: a simulation of a beam with real data introducing controlled synchronization errors. Finally, we will see the tests carried out on a real bridge located on a Spanish road.

In this experiment, the time synchronization between distributed measurement nodes is analyzed. This issue becomes more important as information gathered by each node must be combined to form global information subsequently analyzed. In the case of using WSNs for OMA applied to SHM, small time-synchronization errors can cause major problems in modal identification. Thus, a numerical experiment is presented in which the effects of these time-synchronization errors on the OMA estimation are shown. This controlled simulation allows us to analyze the mode shapes identified with and without error yielding important conclusions that may be used to improve estimations in future applications.

In this section, we study the influence of lack of time synchronization in OMA applied to SHM. The use of WSNs in certain applications such as those related to OMA applied to SHM implied certain challenges such as the time synchronization between distributed measurement nodes. The deviations in clock offset between measurement nodes in a distributed WSN can lead to unsuccessful estimations of modal shapes using OMA. Frequency variation between two different measures may be used as an indicator of possible structural damage, while modal shapes are used as a tool to localize damage. In most cases, only one measurement is present and frequency and modal shapes are used for finite element updating. A remarkable deviation in the modal shapes would make damage location or model updating a difficult task that would lead to wrong results. If these results are going to be used as a starting point for maintenance or structural modifications, the consequences are unpredictable.

Simulations

The stage on which the experiment is raised and on which subsequently the results are extracted considers that the only source of error of time synchronization is the phase error between different measuring points. Five spaced points on a beam extent of nine meters are considered including the two fixed points at the supports (Figure 6.6). The manually induced phase error remains constant and stable throughout the measurement. The phase error is applied to different measuring points. As a further check, the software used to OMA identifies correctly and adds no additional errors. In this way, we ensure that all differences after the modal identification are due to the induced phase error in the selected channels.

Figure 6.6. Example of first four mode shapes of a Bernoulli beam with five measurement nodes.

A two-dimensional static Bernoulli beam [116] has been studied to get its modal parameters. The OMA techniques use measurements at discrete point locations to determine the modal shape. This continuous function is suitably approximated if the measurement nodes are at points where the vibration mode is observable. The problem can arise when several modes want to be identified using the same sensor positions. For example, Figure 6.6 shows a beam with five measurement points, two fixed in the ends and three in between (a-a', b-b', c-c'). With those three two different combinations of the sensor, locations are showed to identify the first four modes, (a, b, c) at $(\frac{L}{4}, \frac{L}{2}, \frac{3L}{4})$ and (a', b', c') at $(\frac{L}{5}, \frac{3L}{5}, \frac{4L}{5})$. The first choice would be selected if the expected modal shape is not already considered. In this case, mode 4 is not detected and mode 2 is poorly represented. The second alternative has measurements in all nodes with at least half of the maximum amplitude reached for a given frequency within these four modes. Other modal shapes can be poorly represented.

All the operations and analyses relating to structures and modal forms could be carried out thanks to the help of Jaime García Palacios, Jose Manuel Soria Herrera and Roberto Ortega from the Hydraulics and Energy Laboratory of the Escuela de Ingenieros de Caminos, Canales y Puertos of the Universidad Politécnica de Madrid. In this practical use case, an ideal modal shape is compared against a modal shape obtained at given discrete points after an OMA is carried out. Therefore, the analytical acceleration series in the time domain with a sampling of 1000 Hz are generated in MATLAB from the superposition of the modal shapes at each measurement node. A first analysis compares the result of the obtained modal shapes with the theoretical ones and they completely fit. A second analysis introduces a synchronization error over two of the measurement points. The results are compared with the theoretical modal shapes obtained when this error is expected, and they both matches.

It is assumed that at each measurement node the amplitude of the registered acceleration is the same as the ideal one at the same node. The difference between independent nodes in the same position is due to small synchronization errors. To be more specific, with time-synchronization error, a phase deviation is considered. This

deviation makes that the maximum amplitude in different measurement points is reached at different times.

Therefore, at a given time, not all the channels would have reached the maximum possible value. Figure 6.7 shows the influence of phase deviation due to a time-synchronization error. It is assumed that measures are carried out at three intermediate nodes ($\frac{L}{4}, \frac{L}{2}, \frac{3L}{4}$) and channels connected at ($\frac{L}{4}, \frac{3L}{4}$) have a retarded angle of deviation. At time t_1 the maximum is reached at span center (a) while at nodes (b) and (c) the amplitude is lower. Therefore, at time t_1 the modal shape is represented by the blue line. If this approach is considered when nodes (b) and (c) reach the maximum amplitude (time t_2), node (a) has an amplitude smaller. In this case, the modal shape is represented by the magenta line. The modal shapes identified after OMA have normalized the maximum value of the theoretical displacement to be able to compare the series.

Figure 6.7. Influence of phase deviation between different measuring nodes

Since in this use case, only the influence of the phase deviation due to synchronization error between channels is studied, the ideal Bernoulli model is used for now on. The real world differs from this behavior adding errors, such as fabrications differences, fluctuations in the environment, slightly different clock speeds between measurement nodes, etc. As was commented above, in this case only, a phase error is considered. That phase error is controlled and manually added to some of the measurement nodes and the offset between measurement nodes is constant and stable during all the measurement.

Very small errors in time-synchronization cause important errors in OMA. The influence is not significant on the identified frequency, but it is on the modal shape. It should be considered that the main purpose in OMA is to detect, localize, identify and if possible, to determine the remaining span life of the structure. For detection based on frequency, variations could be a good signal, but for the remaining steps, modal shapes become very important. They can show where the structure has suffered more variation helping to localize the possible damage. Identification is more related to how this variation can be implemented in a Finite Element Model (FEM) to follow a given modal shape. As more modes are identified, more information to calibrate a

model is obtained and better radiography of the structure is obtained because some damage can only be detected for higher frequencies. A constant synchronization offset among channels introduces a phase deviation that it depends on the identified frequency and it is directly proportional to it. However, the error in the amplitude of the channel respect to the maximum possible amplitude at a given time is not linear as it is related to the cosine of the phase deviation. This introduces important errors as the angle grows to $\pi/2$ rad. In fact, with a phase deviation of π rad the modal shape can be completely different; e.g., when two sensors are located at both sides of the middle span cross-section of a bridge, the first flexural longitudinal mode is seen with both sensors accelerating up and down synchronously. If one of them has a phase deviation error for this frequency of π rad, when one moves in one direction, the other goes on the contrary one. This means that this mode is identified as a torsional one instead of flexural.

As mentioned before the sampling frequency (f_s) is a very important parameter of this model. It is assumed that synchronization between the acquisition systems allows assuring that there is no more difference among two channels than the time gap among two recorded samples. This is common to WSNs. This gap corresponds to a maximum error time given by $t_e = \frac{1}{f_s}$.

If the measurement is carried out at m points then there are two modal shape vectors than can be compared, the ideal one and the error series. The Modal Assurance Criteria (MAC) value is usually considered to compare mode shapes. A value of MAC closes to 1 means a good correlation between the two vectors, while a value of 0 shows orthogonality. Assuming the orthogonality of mode shapes the MAC, value is a good help to the difference between two orthogonal modes with close frequencies.

Practical case

A Bernoulli beam with a length of 9 m is studied. Nine modes are considered and values in Table 6.2. are given for the theoretical frequency.

In the same table for a given sampling frequency of 1000 Hz, the error of angle, as well as amplitude, are obtained for each mode. The MAC value has been obtained considering 7 and 27 equally distributed measurement nodes on the beam. In both cases, it is supposed that half of the sensors (pairs or odds) have reached the maximum amplitude and the intermediate ones have a phase deviation angle given by one sample at the given frequency rate.

For each of the modal frequencies (f_n) associated with the modal order (n), the values of phase deviation (φ), amplitude error (Am_{error}) and MAC are computed to introduce the amplitude error in the modal shape. It is supposed that half of the channels (pairs or odds locations) have reached the maximum amplitude while the other half have an amplitude given by the maximum minus the amplitude error associated with the studied frequency, as shown in figure 2. MAC_{7n} represents the MAC value obtained with 7 sensors equally spaced and MAC_{27n} with 27 measurement points. In both cases, the end sensors are located on the supports.

Table 6.2. Theoretical modes and errors associated with 1000 Hz of sampling frequency. For each of the modal frequencies (f_n) associated to the modal order (n), the values of phase deviation (φ), amplitude error (Am_{perror}), and MAC values are obtained with 7 (MAC_{7n}) and 27 (MAC_{27n}) measurement points are shown.

n	f (Hz)	φ ($^\circ$)	Am_{perror} (%)	MAC_{7n}	MAC_{27n}
1	5.9218	2.1318	0.0692	1.0000	1.0000
2	23.6871	8.5273	1.1055	1.0000	1.0000
3	53.2959	19.1865	5.5546	1.0000	0.9992
4	94.7482	34.1094	17.2031	0.9912	0.9912
5	148.0441	53.2959	40.2317	0.9404	0.9404
6	213.1835	76.7460	77.0732	0.8892	0.7178
7	290.1664	104.4599	124.9702	0.2650	0.2650
8	378.9928	136.4374	172.4622	0.0249	0.0249
9	479.6628	172.6786	199.1847	1.0000	0.0000

Figure 6.8. Comparison of mode shape 9 (red line) with 7 (blue line) and 27 (green line) measurement points

If many measurement points are considered, as is the case of 27, there is no difference between choosing the pairs or odds to add the phase error. With only a few points, as is the case of 7, the difference between choosing one or another option could be visible.

Analyzing MAC values of Table 6.2. there is no difference between the case with 7 or 27 nodes except for the case of $n = 6$ and $n = 9$. Modal shape 6 is poorly represented with only 7 equally spaced points measurement points where only those with the phase error will be identified. All these positions have zero modal displacements for this modal order. Modal shape 9 has an interesting behavior that can be seen in Figure 6.8. With only 7 points of measurement, the phase is inverted π rad. But does not happen in the case of 27 measurement points. That is why for mode 9 with a phase angle deviation close to π rad. the MAC value obtained with 27 measurement nodes becomes 0, while with 7 nodes is 1.

Road bridge in Spain

To verify that the acquisition hardware platform, together with the synchronization protocol, worked correctly, several tests were carried out on the road AP-67, north direction side, in Aguilar de Campoo bridge, Spain. During the tests, a National Instruments (NI) wired measurement equipment was taken, which acted as

a reference system since it had been used in previous measurements and its correct operation had been verified.

For these tests, a time-synchronization protocol was developed with the name PulseStar. PulseStar was based on a star topology where the central node emitted energy pulses. With the first pulse, the rest of the nodes started recording samples. More energy pulses were emitted during the measurement. Each node stored timestamp in the MAC layer when it was detected. The clocks of each node ran freely and were not corrected during the measurement. At the end of the measurement, each node sent the recorded data to the central node where the data fusion was performed, matching the recorded energy pulses. This protocol allowed us to synchronize the samples with sub-millisecond accuracy.

In the current tests, the wireless system composed of seven reference sensors and nine roving sensors measured all the 34 nodes by 3 setups, as shown in Figure 6.9. In total, four wireless boxes, each one with four available channels, were used for the tests. The 4-channel NI system was used mainly for evaluation of the time synchronization of the wireless system. There are two overlapped reference nodes #17 and #20 for the wired and wireless systems.

Figure 6.9. Diagram of the monitored road bridge with the position of the 34 measurement points.

Besides, in each setup, there were two roving sensors of the wired system overlapped with two of the roving sensors of the wireless system. Each setup had been repeated twice, for a measurement duration of 10 mins and 15 mins, respectively. The ambient vibrations on the road bridge were measured when one lane was closed while the other lane was still open to the traffics.

The identified mode shapes of Mode 1 and Mode 5 from the 10-min measurements are plotted in Figure 6.10.

Figure 6.10. Identified mode 1 (left) and mode 5 (right) of the road bridge after 10 min measurement.

The identified mode shapes from Mode 1 to Mode 5 from the 15-min measurements are plotted in Figure 6.11. The relative differences between the frequencies identified from 10-min measurements and 15-min measurements are very low, less than 2 %.

Figure 6.11. Identified mode 1 (left) and mode 5 (right) of the road bridge after 15 min measurement.

Results

Practical case

The selected nine modes are used to generate a time history series of the beam acceleration at each considered node. This time histories are generated for the theoretical perfect modes. The calculation focusses only on the case of 7 nodes. A phase deviation corresponding to 1 sample over 1000 Hz (1 millisecond) is added to nodes 3 and 5 (odds). The aim is to demonstrate the importance of synchronization errors associated with each of the modes identified.

These series are identified using the Expectation-Maximization SSI method. In the first analysis, the series without errors are analyzed and in the second one, the series with synchronization errors are used. Both series are compared, and in this way, the influence of the phase angle deviation is evaluated.

Table 6.3. Frequency comparison of series where the generated theoretical frequencies (f_{theo}), the identified frequencies from the signal without error (f_{wout}) and the identified frequencies from the signal with the induced offset (f_{with}). The lack of synchronization does not affect the identification of the natural frequencies of the structure.

n	f_{theo}	f_{wout}	f_{with}
1	5.9218	5.9220	5.9218
2	23.6871	23.6870	23.6871
3	53.2959	53.2959	53.2959
4	94.7482	94.7482	94.7482
5	148.0441	148.0441	148.0441
6	213.1835	0	0
7	290.1664	290.1664	290.1664
8	378.9928	378.9928	378.9928
9	479.6628	479.6628	479.6628

A first result that can be seen in Table 6.3 is that the lack of synchronization between nodes does not affect the identification of the natural frequencies of the structure. That is, the natural frequencies of the beam are properly identified taking as data source the ideal signal and taking as data source the signal with the induced offset error. The three columns showed correspond to the generated theoretical frequencies (f_{theo}), the identified frequencies from the signal without error (f_{wout}) and the identified frequencies from the signal with the induced offset (f_{with}). From these results it can be concluded that frequencies are the same. Only modal shape 6 is not identified due to the location of the sensors that generate spatial aliasing for this modal shape.

A figure of each of the eight identified modes is shown in Figure 6.12. The theoretical modal shape is drawn as a dashed red line while the blue line represents modal identified theoretical shape and green line modal identified shape when synchronization error of 1ms is introduced at nodes 3 and 5.

After the identification process, there is no difference in the modal displacements between the theoretical modal shape and the identified one. As well as between a theoretical shape within the induced error and its corresponding identified one.

Figure 6.12. The eight identified mode shapes of a Bernoulli beam with seven measurement points with the theoretical modal shape (dashed red line), the modal identified theoretical shape (blue line) and the modal identified shape when synchronization error of 1ms is introduced at nodes 3 and 5 (green line).

In Figure 6.12 the eight identified modal shapes have been represented to explain the problem. For the firsts three modes, no clear graphical difference between both identified series is seen. In Table 6.4 the numerical results show that the difference grows with the modal order, as it was expected from the theoretical background. From modal shape $n = 4$ onwards the differences are even graphically visible.

For modal shape three, there are no differences between the signals. The reason is that nodes 3 and 5 cross over zero at the same instant that the central node is at its maximum amplitude.

Looking at modal shape 4, nodes 2 and 6 in table 3 and MAC value of 0.9912 from table 1 the following conclusions are thrown. First, a good MAC value can be used to discard different modal shapes, but it is not a clear indicator of the modal

displacement of each one of the measured nodes. In this case nodes, 2 and 6 in the modal shape have a difference of 17.19% concerning its desirable amplitude.

This difference becomes important when these results are going to be used for damage location or FE updating. The current error will increase with modal order as can be seen in modal order 5 where this difference becomes 40.23% with MAC of 0.9404. Differences in modal shape 7 are huge in the graphical and numerical analysis as well as the poorly MAC value of 0.2650.

If a visual inspection of modal shape 8 is carried out to identify this as a possible mode, as it is usually done in OMA analysis to avoid spurious poles, it looks like a second bending mode while its real order is 8. Therefore, the lack of synchronization could induce a wrong visual interpretation of results.

As has explained before, mode 9 is a case, where the graphical and numerical information tells that the identification with and without phase error is the same. This is only when seven nodes are considered. This is because the phase error inducted in this case is near π radians making the signal with error turn over in the nodes with no error. Nodes 3 and 5 have no modal displacement, therefore, the identified modal shape coincides with the theoretical one.

Table 6.4. Normalized amplitude of each of the seven measurement points (N_{dx}) between the modal identified theoretical shape (ϕ_{wout}) and the modal identified shape when synchronization error of 1ms is introduced (ϕ_{with}).

Mode	Node	Nd_1	Nd_2	Nd_3	Nd_4	Nd_5	Nd_7	Nd_7
1	ϕ_{wout}	0	0.5000	0.8660	1.0000	0.8660	0.5000	0
	ϕ_{with}	0	0.5000	0.8654	1.0000	0.8654	0.5000	0
2	ϕ_{wout}	0	0.8660	0.8660	0.0000	-0.8660	-0.8660	0
	ϕ_{with}	0	0.8565	0.8660	0.0000	-0.8660	-0.8565	0
3	ϕ_{wout}	0	1.0000	0.0000	-1.0000	0.0000	1.0000	0
	ϕ_{with}	0	1.0000	0.0000	-1.0000	0.0000	1.0000	0
4	ϕ_{wout}	0	0.8660	-0.8660	-0.0000	0.8660	-0.8660	0
	ϕ_{with}	0	0.7170	-0.8660	-0.0000	0.8660	-0.7170	0
5	ϕ_{wout}	0	0.5000	-0.8660	1.0000	-0.8660	0.5000	0
	ϕ_{with}	0	0.5000	-0.5176	1.0000	-0.5176	0.5000	0
7	ϕ_{wout}	0	-0.5000	0.8660	-1.0000	0.8660	-0.5000	0
	ϕ_{with}	0	0.1249	0.8660	0.2497	0.8660	0.1249	0
8	ϕ_{wout}	0	-0.8660	0.8660	-0.0000	-0.8660	0.8660	0
	ϕ_{with}	0	-0.8660	-0.6275	-0.0000	0.6275	0.8660	0
9	ϕ_{wout}	0	-1.0000	0.0000	1.0000	0.0000	-1.0000	0
	ϕ_{with}	0	-1.0000	-0.0000	1.0000	-0.0000	-1.0000	0

Road bridge in Spain

The wireless results are compared to the wired results obtained in previous tests. The wired results were already checked, and their validity was known. By comparing the wireless and wired results we validated and ensured that there was no time-synchronization errors. In general, the relative difference between the frequencies is low, less than 5 %. In Figure 6.13 the reader can see the overlap of the results obtained with the wired system (W) and the wireless system developed (WL), taking measurements of 10 and 15 minutes. In the horizontal axis are represented each of the measurement nodes and in the vertical axis are the modal displacement.

Figure 6.13. Superposition of the results obtained with the wired system (black line) and the wireless system developed, taking measurements of 10 (red line) and 15 (blue line) minutes.

Conclusion

In this practical use case, the importance of time-synchronization has been remarked. A very easy theoretical procedure has been introduced to evaluate the amplitude error of each measuring node, associated with each frequency when the maximum synchronization admissible error is known. This is the case of many actual wireless systems. An example of an application with an analytical solution has been used to avoid extra errors. Also, an OMA has been carried out on the analytical constructed time history series to check that the identified modal shapes correspond to the theoretical ones they are derived from. The time-synchronization error has been introduced in a controlled way, and the influence over the modal shapes corresponding to nine modal orders is studied. Within these nine modes, different situations have been analyzed and several conclusions may be drawn.

First, is that a small deviation of time-synchronization does not affect the identification of frequencies. Second, the amplitude error increases with the frequency and this error is maintained after the OMA process. However, in some cases this error can be considered as negligible while in others becomes crucial for modal identification, leading to a wrong visual interpretation of results, impossibility to identify a real mode, correct identification of a mode that is one radian out of phase, etc. The relative importance of this error can be easily evaluated as already mentioned.

It can be noted that the theoretical Bernoulli model used has the frequencies clearly marked and well detached, and the values become very large for orders bigger than 4. These values are difficult to identify in a real structure. However, this example has been selected to point out the importance of time-synchronization, and it should be considered that only 1 millisecond has been considered in 2 measured nodes to get these results.

Tests have shown how the lack of time synchronization in SHM applications directly affects the results. The protocol chosen through the methodology was successfully implemented on a bridge, obtaining both the structure's own frequencies and its modal forms. The identification results correspond accurately with the theoretical results and the wired results.

The test subjects who have used the tool and faced the SHM use case have obtained a set of valid solutions. This experiment shows that starting from the same premises if they are not too strict, the set of solutions can be wide. Within that set of results, each user chooses the solution that he or she finds best. In our case, all the solutions chosen by the users would be valid for the application under study.

In this way, the use of the methodology and the tool is validated in this case of use, where a certain degree of accuracy in time synchronization was necessary.

6.2.2. Environmental Monitoring application

In this use case, we face the selection of a time-synchronization strategy with large freedom for decisions. Multiple details of the user form of inputs are irrelevant because of that lack of restrictions. High synchronization accuracy is not required for this application, as time-synchronization errors in the range of seconds do not have a great influence on the result. This is thanks to the fact that the variables that are monitored are of very slow variation, such as temperature, that their variation is noticeable in minutes or hours, but that in windows of a few seconds it is not noticeable. In this use case, we talk about freedom of decision since there is no imposition of use of any type of hardware platform, nor the type of network, nor scheme of distribution of messages, nor cost.

Hardware platform

The hardware platform presented in this section was designed, implemented and tested by us in a modular way to be useful in multiple environments and applications. In this way, different types of experiments can be carried out on this hardware platform. In this section, we focus on experiments about environmental conditions and how temperature changes affect the stability of the clock sources in WSN. But later, we will see how, by adding new modules, we can improve the previous hardware platform, the “SHM platform v1.0” and extend its coverage to be able to perform SHM tasks in other structures. To achieve these objectives it was thought that the best way to implement them was in a modular way, where the complete system was formed by different boards with different functionalities that could be stacked on top of each other making the global functionality of the system grow.

Stackable modules

The initial design, environmental applications-oriented, was carried out with two hardware layers: supply and regulation board, and control and communications board. The assembly of a basic system can be seen in Figure 6.14.

Figure 6.14. Modular stackable hardware platform. The initial basic system composed of two layers: power management, and control and communications (left). Detail of different clock sources (right): a) external 32.768 kHz, b) external 8 MHz for PLL, c) external DS3231 TCXO.

The **power regulator board** (Figure 6.15) contains four parallel power down regulators that transform the input signal to 12v, 5v, 3.3v, and 1.8 volts and filter the signal to both input and output. All regulators are switched with the capacity to supply 3 amps at their output.

Figure 6.15. Schematic diagram of the board containing the four parallel voltage regulators.

The **control, storage and communications board** (Figure 6.16) is responsible for managing the entire system and where we implement the strategies of time synchronization. This board has as general and main objectives the storage of the collected data, as well as the synchronization between the different nodes with sensors that are used to perform the measurement. This hardware module is where the different clock sources to be tested are located.

Figure 6.16. Schematic diagram of the control, storage, and communications board.

The hardware radio frequency module and the software module for time synchronization were used for time synchronization. The dual RF module is the one needed for the wireless communications, to exchange commands and instructions between nodes. This radiofrequency module works in sub-GHz frequencies having the ability to switch between the 433 MHz and 868 MHz bands. The implemented module is based on the Semtech SX1276 radio transceiver with LoRa compatibility. The control of this transceiver is used as a 32F411-DISCOVERY board that incorporates a microcontroller STM32F411 of the same manufacturer. This microcontroller is based on an ARM Cortex M4 32-bit core.

The design of this hardware platform was made with the idea of being able to implement different time synchronization strategies in it. This platform is a generic platform where, in multiple cases, it can be considered to be oversized.

For the application of environmental monitoring, we have used only the two first stackable modules with some modifications such as an additional 32.768 kHz external oscillator, or an external DS3231 TXCO. For the STM32F4xx board's family, the LSI clock accuracy is not recommendable for calendar applications [117]. The LSI clock is neither factory calibrated nor very accurate. Typically, its frequency is between 30 - 60 kHz. The LSI clock incorporated in the STM32F4xx boards is the example of CMOS oscillator we are going to test. This kind of oscillator, unlike the other oscillators in the system, acts as a low-power clock source that can be kept running in Stop and Standby modes. In this board, all internal clocks except LSI are stopped in Stop mode [118]. A standard external 32.768 kHz 20-ppm XO, a DS3231 TCXO and the main HSE at 96 MHz are the other three clocks sources under comparison.

As commented before, a similar device from a different vendor has been tested to compare the behavior between vendors. Most of the tests have been done with an STMicroelectronics platform. In this case, to evaluate whether the microcontrollers' manufacturers have any effect on the behavior of external oscillators, a platform from a different vendor has been selected. The platform selected is an Arduino UNO R3 which is based on a Microchip ATmega328P with an external crystal oscillator of 16 MHz.

Assumptions

In this use case, the methodology pursues the goal of finding a synchronization mechanism for an environmental monitoring application. The application monitors multiple parameters, such as temperature or humidity, covering a large area of terrain using a WSN. It is necessary to aggregate all the data to have a global vision of how the monitored variables in the area evolve. The aggregated data must be time-synchronized to obtain an accurate picture of the environment. The information must be transmitted periodically and continuously to a processing center. As the parameters under monitoring vary slowly with time, their acquisition is done every 10 minutes.

The nodes that make up this monitoring network need to have low power consumption for several reasons. On the one hand, some of them are going to be installed in areas of difficult access, so replacing their batteries will not be a quick task. On the other hand, the number of nodes that make up this network is large and replacing or recharging the batteries of all the nodes is a task that should be reduced to a maximum of once a year. In this case, the application is intended for users who only want to modify the software at the application level. Furthermore, non-specific hardware platforms are to be used, where, at most, an external clock source is added.

Methodology

Regarding the first input block, in this case, the objective of the time-synchronization strategy is data fusion. There is a special interest in the power consumption (value of 3) because the WSN end devices are scattered throughout an extensive area and they are battery powered. The monetary cost (value of 8) is interesting because in this case, the number of end devices is big. The application is developed in a barely hostile environment and therefore it has no special security requirements (value of 10). The network size is imposed by the size of the area under monitoring and the localization is mainly outdoors. The scope of the network to be synchronized is the full network. Neither the topology (value of 10) nor the distribution of messages (value of 10) is very demanding in their requirements. Several options like star or mesh topologies are valid. And the message distribution only needs multi-hop communications. There is no need to distribute the UTC through the nodes because is enough to have a worldwide time in the sink node where the data is merged. The application requirements regarding the maximum error of time between nodes are lax because environmental changes are slow, and the sampling period is appropriate (value of 8). On the other hand, it is interesting that the stability of the clock sources is good so that during these long intervals between samples, the clocks of the nodes do not deviate too much (value of 7). In this application, the synchronization complexity seems low. The reference clock in the sink node where the data is merged is static. There are no established requirements regarding the synchronization convergence time, the type of clock sources used, the use of memory in the microcontroller, the frequency band to be. In this use case, regarding software abstraction, the user can only touch the software at the application level (value 2). And finally, concerning the hardware abstraction, the user at most wants to use external clock sources (value 3). All parameters are summarized in Table 6.5.

Table 6.5. Summary of the values assigned to the nine parameters of the methodology for a specific environmental monitoring application.

Power Consumption	3	Monetary Cost	8	Security	10
Network Topology	10	Message Distribution	10	Accuracy	8
Clock Stability	7	SW Abstraction	2	HW Abstraction	3

In this case, three parameters are irrelevant for the user: security, network topology, and message distribution. This makes the filter, in all those parameters, allow the input of all the protocols. The accuracy and monetary cost parameters are also lax and make the filter area large. On the other hand, there are three parameters with very low values that we must interpret according to their levels. The value 3 in power consumption indicates that we are looking for very low consumption protocols. The low values of software and hardware abstraction indicate that the user is only willing to use protocols with a high abstraction for the user, that is, programming in

the upper layers of the software stack, specifically in the application layer, and generic hardware platforms where only external clock sources need to be added. The user tool generates a radar chart with the area shown in Figure 6.17.

Making use of the user filter in the tool, the report generated can be seen in Figure 6.18. From all the possible solutions offered by the radar chart, we have chosen the use of RTC modules based on TCXO in each of the nodes. Considering the scenario, it would be recommended to use a sub-gigahertz transceiver because the distances between nodes can be large and because the data transferred from one node to another is small. The network topology recommended would be mesh topology because it is the most adaptable to the scenario. The RTCs values are corrected on-demand with reference timestamps reducing in this way the message exchange due to time synchronization.

Figure 6.17. The radar chart generated for the specific requirements of an environmental monitoring application. Big values represent few constraints or big freedom in the selection, and low values represent high demands or very restrictive conditions.

As in this case, there are three parameters to which the user does not give importance, the filter supports any solution regarding them. Therefore, for those parameters, the filter is the largest possible and therefore increasing its area. In Figure 6.18 you can see how for security, network topology, and message distribution the filter has a value of 10.

Figure 6.18. User selection tool showing a table with all the viable time-synchronization that fulfills the user filter and a graphical representation with a radar chart.

Test

To check that the strategies proposed by the tool following the guidelines of the methodology meet the requirements, not only a real application has been implemented, but also an intensive study of different clock sources.

The developed hardware platform serves to carry out stability tests with different clock sources [90]. We conducted real experiments and theoretical analyses of the behavior of different clock sources over this platform. To corroborate the results, the same experiments were also carried out on another commercial hardware platform from a different manufacturer. We highlighted and demonstrated the importance of the early stages of design, especially the selection of the clock source because that decision has a great impact on the performance of the time synchronization in wireless sensor networks. On our hardware platform, we performed real experiments with CMOS clock sources without calibration, with CMOS clock sources with calibration, with clock sources based on external 32.768 kHz oscillators, with clock sources based on TCXOs and with clock sources based on the internal HSE oscillator increasing its frequency through a PLL.

The representation of the analysis of the theoretical behavior of the different clock sources shows how, in addition to the basic quality of the clock source, it is necessary to consider the influence of the temperature. Depending on the type of clock source, the temperature influence can provoke high-drift variations, which makes the software correction of the TS more complicated. Those theoretical analyses have been corroborated conducting experiments of performance, where the CMOS, XO, and TCXO clock sources have been tested at room temperature and with abrupt changes in temperature. The results show that in the future, with good modeling of the working temperature and clock sources, the theoretical calculations can give us a very good approximation of the behavior of clock sources in real environments.

This work shows that a good choice of the clock source in the early phases of the design of a WSN can lead the developers from failure to success. The clock source choice in the early stages of design has a great impact on the performance of the TS in WSN. The developers should always choose the clock source that best suits their needs and consider the influence of the temperature.

There are several types of clocks sources that can be used in WSNs. None of them are perfect, and their oscillator can vary its frequency over time due to changes, mainly, in temperature. The clock sources have a direct influence on time synchronization tasks, and those tasks of time synchronization, in some cases, have a direct impact on the results and behavior of the main application of the WSN. Knowing the clock source used by the nodes in a WSN is a key factor in the design of applications that require time synchronization.

In an environmental monitoring application, for example, to monitor the temperature of a vineyard, different clock sources were tested to check their stability against temperature changes.

For this case of application, real experiments and theoretical analysis of the behavior of the clock sources most used in WSNs have been carried out. CMOS oscillators have low accuracy, bigger than 500-ppm and a high dependency with temperature. External crystal oscillators have good accuracy, around 20-ppm, and are stable with temperature. TCXO is very accurate, around 5-ppm, and the temperature does not influence their drift. The use of PLL circuits minimizes the impact of temperature and stabilizes oscillators. We highlight and demonstrate the importance of the early stages of design, especially the selection of the clock source because that decision has a great impact on the performance of the time synchronization in WSNs.

The objective of this empirical study is to prove that if we can improve the stability and accuracy of the clock source on which our synchronization is based, the synchronization protocols will need to perform fewer re-synchronization rounds and, therefore, our networks will be synchronized more accurately for longer periods. Knowing the performance and behavior of the clock source of a WSN allows the developers to address certain TS problems at an early design stage. In this way, it will be easier for developers to solve future problems with the implementation of software protocols. Addressing the problem in the early phases of design will allow the developers to have a WSN with a more robust and stable TS. Also, analyzing the clock sources of our sensor nodes and understanding what type of clock is appropriate in each case helps the developers to better design and understand the operation of TS in WSNs. This use case is aimed to enable designers of WSNs to compare and evaluate the consequences of using different clock sources, based on their empirical behavior. Start with early design decisions is always better, for example, starting in the hardware design phase, to minimize defects in the later stages, such as, in software developments.

Analysis of theoretical behavior

We use the following three groups to cover the most used clock sources in WSN:

- RTC based on the internal CMOS oscillator of a microcontroller.
- RTC based on an external 32.768 kHz XO.
- RTC based on an external TCXO.

Therefore, an analysis of the theoretical behavior and the practical experiments have been carried out for each of these groups. The behavior of these groups must coincide with the analysis carried out by the manufacturers in their respective datasheets.

The purpose of these analyses is to be able to compare these theoretical data with the empirical results that we will show in the next subsection. All clock sources have been used as the basis of an RTC. In this way, the theoretical analysis and experiments are more easily comparable and the conclusions more clear. The results obtained are easily extrapolated to any other clock source. Two sources of the clock with a different frequency of oscillation, that is to say, with different granularity between generated events, but with the same accuracy in ppm, cause the same deviation after an equal interval of time.

Among all the environmental factors that can affect the behavior of clock sources, the most relevant parameter is the temperature. Considering the temperature as the fundamental parameter that causes changes in the skew, the rest of the parameters can be considered negligible [22]. This is why in this section we are going to consider that all skew variations are due to changes in temperature. A null offset has also been considered throughout the work, simplifying the analysis and making the results more intelligible.

Figure 6.19: Example figure of a DS3231 TCXO where the behavior of a typical uncompensated crystal oscillator and a compensated oscillator can be seen. The figure shows how an oscillator slows down at cold and hot temperatures [119].

Figure 6.19 is the example figure of a TCXO DS3231 showing the behavior of typical uncompensated crystal oscillators against temperature [119]. The figure compares how an XO and a TCXO are affected due to changes in temperature. Dashed lines represent the range of ± 20 -ppm for the XO. That figure shows the general behavior of an oscillator, and how it slows down at cold and hot temperatures and how its tolerance increases up to $\pm 15\%$. The turnover temperature is, as a rule, 25°C and its tolerance at that point is usually $\pm 5^\circ\text{C}$. The temperature coefficient is a constant in the range of 0.030 to 0.042 with a typical value of 0.036 [120]. As a numerical example, we assume a turnover temperature of 25°C and $k = -0.042 \text{ ppm}/^\circ\text{C}^2$. The fractional change of frequency can be calculated as shown in Equation 6.1:

$$\begin{aligned} \text{Error}(\text{ppm}) &= k * (T_0 - T)^2 && \text{Equation 6.1} \\ k &= -0.042 \text{ ppm}/^\circ\text{C}^2 \pm 15\% \\ T_0 &= 25^\circ\text{C} \pm 5^\circ\text{C} \end{aligned}$$

Giving a real example with a working temperature of 85°C , we obtain that the clock source deviates an additional $-151 \text{ ppm} \pm 50 \text{ ppm}$. This additional deviation due to temperature changes is independent of the factory accuracy of the clock source. That is to say, the deviation due to changes in temperature is independent of the accuracy of the oscillator.

RTC based on internal CMOS oscillators

CMOS oscillators have a high skew and drift. These are the most used and basic solutions for a clock in modern microcontrollers. They are usually able to operate in all low power consumption modes. We include in this group all the oscillators with an accuracy bigger than 100-ppm. In our theoretical analysis, we take a clock whose

skew grows linearly at an average rate of 1,000-ppm (86.40 seconds/day, 8.76 hours/year).

Deviations in an oscillator can be both positive and negative, meaning that in some cases the deviation is increased and in others is decreased. In Figure 6.20 it is shown only the absolute value of the maximum, that is the worst case. In Figure 6.20 left, we can observe the evolution of deviation in minutes of a 1,000-ppm oscillator. The theoretical analysis was done at 25°C, 35°C and 45°C. The legend shows the deviation due to the variation of temperature (ppmTmp) and the accumulated deviation (ppmTot). The deviation is calculated based on the deviation due to the accuracy of 1,000-ppm plus the result of Equation 6.1 using the aforementioned temperatures. The accuracy deviation calculation can be achieved by multiplying the frequency deviation of the clock sources by the time intervals, which gives the absolute deviation between the local and real-time. After one day, the oscillator deviates 86.40 seconds at 25°C, but if its temperature is 20°C higher, an additional 1.45 seconds can be diverted. But the real contribution here is to see how, depending on the accuracy of clocks, the influence of the temperature could be negligible or become the most important factor. Being an imprecise oscillator due to its quality of 1,000-ppm, the contribution of the deviation by temperature represents a small percentage. This is common in oscillators with an accuracy higher than 100-ppm.

Figure 6.20. The absolute maximum of the theoretical value of the deviation of an oscillator with different accuracy (yellow bars) and the influence of constant temperature ($\pm 10^\circ\text{C}$ red bars, $\pm 20^\circ\text{C}$ blue bars) with a 1,000-ppm oscillator (left) and with a 2-ppm oscillator (right).

RTC based on external crystal oscillators

Clocks based on XO to generate RTC are more stable than those based on CMOS. These kinds of clock sources are usually able to operate in low power and sleep modes, but not in stop modes.

We are including in this group all oscillators with precision between 10- and 100-ppm. The typical frequency of an oscillator that is used for an RTC is 32.768 kHz, which gives us a granularity of 30.5 μs . Its skew grows linearly, but much more slowly

than in CMOS. In this case, we are analyzing clocks with 20-ppm (1.73 seconds/day, 10.51 min/year) without changes in temperature. If the temperature grows till 45°C, the 20-ppm clock can reach 37-ppm with a deviation of around 3.2 sec/day.

RTC based on external Temperature Compensated Oscillators

A TCXO can minimize the deviation due to temperature changes, creating a clock source much more stable than a normal XO. We are including in this group all those clock sources with a precision deviation lower than 10-ppm. For this theoretical analysis, we are using clocks with 2-ppm accuracy (0.17 seconds/day and 1.05 min/year). Usually, they can operate in all modes because they have their own external battery. In these cases, the information from the clock source to the system will be only available in running modes via serial communications.

We can see in Figure 6.19 how the TCXO is much more stable than an uncompensated XO [119].

In Figure 6.20 right, we see the deviation in seconds of a 2-ppm oscillator with temperatures of 5°C, 15°C, 25°C, 35°C and 45°C. After 24 hours, there is a 0.173 seconds deviation at 25°C, but with deviations of 20°C, there is an additional 1.45 sec deviation.

The additional deviation due to temperature variations is the same regardless of the base ppm of the clock source, but the relative percentage due to temperature in the 2-ppm clock is greater than the deviation of 2-ppm itself, which means that the temperature has a bigger impact on oscillators with better accuracy.

Table 6.6 shows the maximum time intervals between re-synchronization rounds in a WSN to keep the synchronization error between nodes below a certain level (columns) based on the accuracy of the oscillator used (rows). For example, for a clock source that deviates 20-ppm, if we want an accuracy below 1 millisecond, we need to re-synchronize every 50 s. The table is created based on Equation 6.2, where the necessary time (Round Interval) to reach a certain time error using a specific oscillator accuracy is calculated.

$$\text{Round Interval} = \frac{\text{Synchronization Error[s]}}{\text{Oscillator Stability[ppm]} \times 10^{-6}} \quad \text{Equation 6.2}$$

In this way, Table 6.6 provides a quick method of evaluating the synchronization requirements of a WSN. For example, setting the premise of not wanting the synchronization messages to collapse the network, and therefore, restricting to a maximum of 1 message every 0.1 sec, we can see that the synchronization with a maximum error of 1 μ s is unfeasible for systems with clock sources over 20-ppm. In less demanding applications, where a maximum error of 1 ms can be allowed, we can use almost any clock source, always bearing in mind that the better and more stable our clock source is, the fewer messages and operations we need to maintain synchronization.

Table 6.6. The relation between oscillator quality and maximum synchronization error allowed

	1 s	100 ms	1 ms	100 μ s	1 μ s
1000 ppm	1.000 s	100 s	1 s	0.1 s	0.001 s
100 ppm	10,000 s	1,000 s	10 s	1 s	0.01 s
20 ppm	50,000 s	5,000 s	50 s	5 s	0.05 s
5 ppm	200,000 s	20,000 s	200 s	20 s	0.2 s
1 ppm	1,000,000 s	100,000 s	1,000 s	100 s	1 s

Test setup

The objective of conducting experiments on a real WSN is to verify that the hypotheses assumed in the theoretical analysis are correct. Even so, when performing experiments in real scenarios there are circumstances that are difficult to analyze in theory or that have been overlooked, such as hardware differences and interactions between nodes.

The experiments make use of several scenarios but all of them have the same basis as can be seen in Figure 6.21. The reference node has a CMOS, an XO, and a TCXO and it is connected via serial communications with a computer. The computer is running with Windows 10 and the Windows Time Service is configured with phase correction adjustments every 86,400 seconds (24 hours) against an NTP server. All nodes cohabit in two environments, where the temperature is controlled thanks to an additional two sensor nodes, one near the reference node, at an ambient temperature above 20°C, and the other near a cabin with a controlled temperature of 8°C.

Figure 6.21: Graphical representation of the WSN scenario

To test the behavior of the clock sources under temperature changes, in some of the tests, a set of nodes was placed inside and outside of the 8°C controlled cabin. In

this way, that set of nodes undergoes drastic temperature changes between the 8°C in the cabin and the more than 20°C of the room's ambient temperature.

Measurements and comparisons between the different RTC clocks have been made. We have tested and compared the behavior of CMOS clocks, based on an internal CMOS oscillator of 32 kHz without and with calibration. External oscillators of 32.768 kHz of 20-ppm and 100-ppm quality have also been tested. We have also evaluated several DS3231 TCXOs. Besides, to evaluate the behavior of the main clock in WSNs, experiments have been carried out with the HSE clock that is normally based on an external oscillator (16 MHz in our case) and some type of internal PLL that increases its speed (96 MHz in these experiments). Finally, although all the experiments have been carried out on the hardware platform explained in the next subsection, a new test has been performed on a device from a different vendor.

Each node sends the value of its local clocks every several seconds. In the reference node, all data is collected by marking the arrival time via software in the MAC layer and aggregating the data to a common file. This allows us to check the deviation between all the node clocks and the reference one. Multiple experiments have been performed to compare the three clock sources under evaluation.

Results

Experiment with CMOS clock sources without calibration

Table 6.7 shows the values of the frequencies after a performance measurement of the clock sources of the STM32F411 boards as they came from the factory using the standard pre-scalers for a 32 kHz clock source. We can see the differences and the corresponding ppm associated with each clock source are enormous. Because of that low accuracy, these are not recommended for most synchronization applications. All these clock sources exceed 1,000-ppm; some even reach 100,000-ppm.

Table 6.7. Measured accuracy of LSI clocks: the LSI clock factory deviation from an ideal 32 kHz without calibration.

	Measured Frequency (Hz)	Deviation (ppm)	Deviation (%)
Node 1	35.280	102.500	10.25
Node 2	35.950	123.437	12.34
Node 3	31.780	-6.875	-0.69
Node 4	33.670	52.187	5.22
Node 5	33.370	42.812	4.28

Figure 6.22 shows how the clock sources based on two TCXOs and four internal CMOS deviate with time. The NTP clock is the reference one and its line is almost invisible since it is superimposed by the TCXO clocks. In this case, it is remarkable

to see how much the CMOS oscillators with factory defaults pre-scalers deviate. The CMOS oscillators with the maximum and minimum deviation are the ones with 12.44% (124,441-ppm) and 0.65% (6,534-ppm), all over 1,000-ppm. Using this kind of CMOS oscillators as clock sources, as they came from the factory, makes the influence of temperature irrelevant because they deviate so much that the influence of additional factors goes unnoticed. When making the scale so large with the deviation of the CMOS, it seems that the rest of the clock sources represented, such as the external XO oscillators and the TCXO, do not deviate at all, but we show their behavior in more detail in the next figures and experiments.

Every clock has been approximated through a basic linear fitting giving an expression similar to Equation 2.1. In this way, we can differentiate their offset and skew (Figure 6.22 and Figure 6.24). The difference between the real data and the fitting line is the accumulated drift over time (Figure 6.23 and Figure 6.25). The mathematical expressions of every clock are over-impressed in the figure. It has been proven through several performance measurements that the CMOS oscillators in long-term experiments tend to deviate the same percentage. Their skews grow linearly and they are precise but not accurate.

Figure 6.22: Long-term experiment with non-calibrated internal CMOS oscillators. Two nodes are at room temperature (yellow and green lines) and the other two nodes alternate between the room and cabin temperature (cyan and magenta lines).

In Figure 6.23, the accumulated drift (skew variation) over time of the clock is shown with the temperature variation. In the experiment carried out, two of the nodes were at room temperature (left) and two were subjected to sudden changes in temperature (right).

Figure 6.23. The accumulated drift of non-calibrated internal CMOS oscillators versus temperature: room temperature in (left) and drastic temperature changes in (right). The accumulated drift follows an ascending line when the temperature falls and a downline when the temperature grows.

We can observe how in both cases, at room temperature and with drastic changes in temperature, the accumulated drift follows the same pattern. As a general rule, we can infer that the accumulated drift carries an ascending line when the temperature falls and a downline when the temperature grows.

Experiment with CMOS clock sources with calibration

Knowing the factory oscillation frequencies, and using more appropriate values for the pre-scalers for the RTC clock, allows us to adjust its frequency to be more accurate.

Once we verified that the CMOS oscillators of the STM32F4xx boards are not factory calibrated and that they deviate greatly from the theoretical 32 kHz, an adjustment of their pre-scalers was made to adjust their values to those of an RTC. Once adjusted, the previous experiment was repeated. Their results are shown in Figure 6.24.

Figure 6.24: Long-term experiment with calibrated internal CMOS oscillators. Two nodes are at room temperature (yellow and green lines) and the other two nodes alternate between the room and cabin temperature (cyan and magenta lines). A pointed line represents the linear fitting behavior of each node's clock.

We can note that the accuracy of the clocks can be now expressed in ppm, with maximum and minimum deviations of 3,008- and 507-ppm. This new experiment allows us to observe some behaviors that could not be detected in the previous ones due to the calibration.

Now, we observe how the influence of the temperature makes it possible to differentiate the skew from the accumulated drift. The skew grows linearly but the drift does not and is directly dependent on temperature changes.

As in the previous case, the accumulated drift has been represented with the temperature variation. Figure 6.25 left shows the nodes at room temperature and Figure 6.25 right corresponds to the nodes subjected to sudden changes in temperature.

The behavior of CMOS oscillators against the temperature in this experiment is very similar to the previous case but with different ranges since they have been calibrated.

Figure 6.25. The accumulated drift of calibrated internal CMOS oscillators versus temperature: room temperature in (a) and drastic temperature changes in (b). The skew grows linearly but the drift does not, and it directly depends on temperature changes.

Experiment with external 32.768 kHz clock sources

The behavior of the external clock sources used is shown in Figure 6.26. Figure 6.26 left shows a global view to see how the external oscillators vary in magnitude with a CMOS oscillator. It can be seen that the external oscillators are much more accurate and precise than those based on relaxation oscillators.

Figure 6.26. Long-term experiments with external 32.768 kHz oscillators. In Figure (left) it can be seen how the accuracy of the external oscillators is better than that of the LSI by several orders of magnitude. In Figure (right) the same figure is plotter without the LSI line.

In Figure 6.26 right, we have eliminated the CMOS line to observe the detail of the external clock sources. In this case, we can see how the external oscillators of 32.768 kHz behave as expected and within the margins marked by their datasheets, two of them of 20-ppm and another of 100-ppm. The oscillators reach deviations of -3-ppm, -17-ppm and 67-ppm. Repeating this experiment under the same conditions,

it is observed that the results are very similar demonstrating that the external oscillators are precise.

Experiment with Temperature-Compensated Crystal Oscillators

In all the experiments and all the previous figures, the TCXO has been plotted. Their results are so close to the 0-ppm reference that in most cases their lines overlap. We corroborate that their drift is very low independently of the temperature changes, resulting in a maximum and minimum deviation of 4- and 0-ppm in our experiments. An example of how much the drift deviates with temperature in the case of an XO and a TCXO is shown in Figure 6.27.

Figure 6.27: Stability of an XO and a TCXO under drastic temperature changes.

Experiment with High-Speed External Oscillator

The main clock in most of the devices chosen to create WSN is based on external crystal oscillators. Then the base frequency of the oscillator is increased through a PLL generating the final frequency at which the device works. In this experiment, the crystal oscillator has a frequency of 8 MHz and after the PLL process, the frequency is increased to 96 MHz.

This experiment aims to show the behavior of the High-Speed External (HSE) main clock versus temperature changes. The same scenario has been used as in the previous experiments, but in this case, taking the local timestamps of the nodes using the 96 MHz HSE. The results at room temperature can be seen in Figure 6.28 left and with drastic temperature changes in Figure 6.28 right.

Figure 6.28. HSE drift behavior under two different temperature environments: room temperature (left) and drastic temperature changes (right). The influence of temperature changes can be considered negligible for the HSE drift.

Experiment with ATmega328P with 16 MHz external oscillator

All previous experiments have been performed on the same hardware platform as the STMicroelectronics vendor. To have a more general idea of the behavior of clock sources in WSN, we have decided to make an additional test on a similar hardware platform from a different vendor. For this experiment, we have chosen the Arduino UNO R3 platform whose microcontroller is an ATmega328P from Microchip.

The same scenario has been maintained as in previous cases, but in this case, the main oscillator of Arduino UNO, an external oscillator of 16 MHz without PLL, has been used. The results obtained can be seen in Figure 6.29.

Figure 6.29. The behavior of an external 16 MHz oscillator in an ATmega328P device under two different temperature environments: room temperature (a) and drastic temperature changes (b). The cabin environment with drastic changes in temperature has more influence in the oscillators drift.

Conclusion

This practical use case highlights the importance of early-stage design decisions that affect the problem of time synchronization in WSNs. For that reason, we have focused this work on the hardware elements that affect the problem of TS, the clock sources.

The behavior of different types of clock sources, with different quality and under the influence of temperature has been, not only theoretical study in this section but also tested in real scenarios. The representation of the analysis of the theoretical behavior of the different clock sources shows how, in addition to the basic quality of the clock source, it is necessary to take into account the influence of the temperature. Depending on the type of clock source, the temperature influence can provoke high drift variations, which makes the software correction of the time synchronization more complicated. Those theoretical analyses have been corroborated conducting experiments of performance, where the CMOS, XO and TCXO clock sources have been tested at room temperature and with abrupt changes in temperature. The results show that in the future, with good modeling of the working temperature and clock sources, the theoretical calculations can give us a very good approximation of the behavior of clock sources in real environments.

A good choice of the clock source in the early phases of the design of WSN application developers can lead the developers from failure to success. The clock source choice in the early stages of design has a great impact on the performance of the time synchronization in WSN. The developers should always choose the clock source that best suits their needs and consider the influence of the temperature.

The use case and the uTSST has been put in the hands of six test subjects. All of them, despite using different values in some parameters to create the filter has reached the same conclusion, using the synchronization strategy of using TCXO-based RTCs. For this use case, all the users reached a valid solution that gives a solution to the synchronization problem.

With this use case, the operation of the methodology and the tool has been validated by offering a synchronization protocol that meets the requirements of the application. The application where the protocol has been implemented, as it is not very demanding in terms of accuracy, has worked correctly obtaining the expected results.

6.3. Methodology validation

To validate the methodology, we have asked a group of test users to apply the methodology and use the tool by facing two use cases. The use cases they have faced are the same two cases that we have presented in section 6.2. All users have applied the methodology and made use of the tool individually and without being influenced by the answers or experience of the rest of the users. Therefore, all the answers must be evaluated from the perspective that they are users who apply the methodology and use the tool for the first time and therefore have no previous experience in either the methodology or the tool. To deal with the use cases, all test users have been provided with the same information. This information is the assumptions presented in sections 6.2.1 and 6.2.2.

The researchers had different degrees of experience in WSNs to make use of our system, as can be seen in the top section of Figure 6.29. As shown in the figure, half of the participants consider themselves to have a low level of experience in working with WSNs, while the other half considers themselves to have a high level of experience. This allows us to evaluate and validate the methodology and the tool from both points of view.

Figure 6.30. Level of expertise in WSNs (top), and degree of knowledge in the field of time synchronization in WSNs (bottom), being 1 the lowest degree of knowledge and being 5 the expert level.

We have also asked participants about their level of knowledge in the field of time synchronization in WSNs as shown in the bottom section of Figure 6.29. As the histogram reflects, almost all participants consider themselves to have little knowledge in this field, although some consider themselves to have a high level. The tests carried out with this heterogeneous group of users reinforce the motivation and development of this thesis, since as will be seen throughout the following sections the methodology and tools have been useful to all of them.

The test user group, therefore, covers the entire range of end-users, from those who have no experience with WSNs and do not know much about time synchronization, to those who are experts in both subjects.

It is to this varied group of test users that we present the two use cases: application for structural health monitoring and application for environmental monitoring. All of them have been given access to the uTSST without any further explanation than the one included in the tool itself.

In the next two sections we show the answers given by the test users to the ten parameters to be evaluated in the uTSST, that is, synchronization target, power consumption, monetary cost, security, network topology, message distribution, accuracy, clock stability, software abstraction, and hardware abstraction. We also comment on which has been the final strategy selected and if these strategies are valid in each use case.

6.3.1. Structural Health Monitoring application

For this use case 100% of the users ended up choosing the same target: data fusion, as shown in Figure 6.30 section a).

In section b) of the figure we see the values assigned to the power consumption parameter. As the reader can see, the assigned values are very dispersed, from users who do not give it any importance (value 10), to users who look for consumptions below 500uW (value 3). From the assumptions offered to the users, it was possible to extrapolate information concerning the power consumption of the system. But the histogram shows, that given some data, the interpretation of each user can be significantly different.

The values assigned to the monetary cost parameter can be seen in section c) of the figure. In this case, the users have tended to be undemanding since most of the users have chosen values above 8 and the most demanding have selected the value 5.

In section d) we can see the histogram that shows the answers regarding the security parameter. The users tend to consider that the application does not need to be secure (values above 8), although one of the users answered the opposite considering security as a priority (value 1).

One of the clearest ideas in the assumptions was the use of a star network topology. Most users have selected that, as shown in section e) of Figure 6.30. But even with the information provided, there was one user who marked all the topologies as valid (value 9).

Regarding the message distribution parameter, section f) of the figure, we find very different ratings. About this parameter, nothing specific was commented on in

the assumptions and therefore all the information had to be extracted from the comments related to the environment. Perhaps that is why most users have rated this parameter as undemanding, with one-hop distributions (values above 7), while another pair of users have opted for multi-hop distributions.

Structural Health Monitoring application

Figure 6.31. Values assigned by test users to all parameters in the case of the SHM application. All parameters except the target are represented in histogram graphs.

The assumptions also gave precise information on the maximum permissible error in this MHS application. Therefore, most users have marked the option of 1ms (value 4) as seen in section g) of the figure. Two other users have marked more demanding levels, while another user has marked a more relaxed level.

Section h) of the figure shows the histogram of user response values. Most users have selected undemanding clocks, 10-ppm and above. But two users have checked the 0.1- and 1-ppm options, two types of the clock that are rare in generic WSN applications.

The response histogram of the software abstraction is shown in section i) of Figure 6.30. Most users have marked the option with a value of 10 so that the filter will not limit this parameter. The rest of the users have marked values, from 5 to 8, indicating that the users can touch the lower layers of the software stack.

Regarding the hardware abstraction, shown in section j) of the figure, most users have indicated that hardware is not a limiting factor when defining the filter (value 10). Four other users have marked the specific platform option (value 7) which is what was indicated in the assumptions. Another user has marked the option to use a specific microcontroller, while another user has marked the option to use a non-specific hardware platform with unlimited resources.

Each of the users, with the values assigned to all the parameters, has obtained different results. In Figure 6.31 left we show the results of a user with little experience in SHM and little knowledge of time synchronization and on the right the results of a user with experience in WSN and high knowledge of time synchronization.

Figure 6.32. Results obtained by two users who have applied the methodology and made use of the uTSST to choose a suitable synchronization protocol for an SHM application. On the left are the results of a non-expert user and on the right of an expert user.

Figure 6.31 shows that every user has obtained one or a set of solutions, regardless of their degree of experience and knowledge in WSNs and time synchronization. The choice of all users is reduced to the three protocols shown in Figure 6.31 right. We can, therefore, conclude that all users have chosen a valid synchronization strategy for this SHM use case. The differences between some users

in their choice are due to the different weights they have given to each of the parameters.

6.3.2. Environmental Monitoring application

As in the case of previous use, for this environmental monitoring application, 100% of the users have selected data fusion as a synchronization target. We can see the results in section a) of Figure 6.32.

The response histogram for the power consumption parameter can be seen in section b) of the figure. Most users have selected very low power ranges, as indicated in the assumptions (values below 4). Except for one user who has marked the option not to assign a value to this filter parameter (value 10).

The set of answers to the monetary cost parameter valuation can be seen in section c) of Figure 6.32. In this case, the responses are very diverse, from users who have assigned very strict values (value 1) to users who do not give it any importance (value 10). For this parameter there was no specific information in the assumptions, it was only commented that this application required a high number of nodes to monitor a large area. This monetary cost parameter is a parameter that can be refined once a set of possible solutions is obtained, for example, by choosing those that minimize the cost.

Regarding the security parameter, the trend in user responses is to be undemanding and to create filters where security is not a critical parameter. Even so, as the reader can see in section d) of the figure some users marked the value 1 indicating that security was a priority.

In section e) of the figure can be seen that most users have marked that they are looking for a mesh network topology (value 1), others have preferred tree topology, and another large percentage has marked that any topology would suit them.

Regarding message distribution, most users considered this parameter to be not limiting and therefore chose the value 10. Another percentage of users have opted for more demanding values where they were looking for more complex message distribution systems with routing and multi-hop. These results can be seen in section f) of Figure 6.32.

The accuracy value did have a reference in the assumptions, indicating that a very exact synchronization is not required since the parameters to be measured are of very slow variation. Most of the users understood this as shown in the histogram in section g) of the figure. The rest of the users were somewhat more demanding, going down to maximum errors of 10ms (value 5).

Environmental Monitoring application

Figure 6.33. Values assigned by test users to all parameters in the case of the Environmental Monitoring application. All parameters except the target are represented in histogram mode.

The clock stability parameter is a parameter that most users have marked as not limiting when creating the filter (value 10) as can be seen in the figure. However, some users have marked stabilities of 0.1-ppm (value 5), which means using very good quality oscillators.

The assumptions also commented that they were looking for protocols that could be implemented in the upper layers of the software stack. That is why most users

have used values 1 and 2 as can be seen in section i) of Figure 6.32. One of the users has marked the value 10.

Also, in the assumptions, it was commented that it was sought to be able to execute the synchronization strategy in not-specific hardware platforms. Therefore, as the histogram j) in the figure shows, most users have chosen the value 3 or lower. A small percentage has left the default value of 10.

Given the individual responses of each user, a filter has been created with the values assigned to each parameter. These filters have generated one or a set of solutions for each user. It should be noted that in this case, a high percentage of the answers have marked the same solution. We can see in Figure 6.33 left the filter generated by a user with little experience in WSNs and on the right the filter created for an expert user.

Figure 6.34. Results of the filters created for an environmental monitoring application. The figure shows the values assigned to each parameter by a non-expert user (left) and an expert user (right).

6.3.3. User experience assessment

Through a form, we wanted to know the users' opinions about the methodology and their experience with the use of the TSST.

In Figure 6.34 we can see the histograms of users' answers when we have asked them about the methodology.

A value of 1 means "Not at all", that is, that they do not agree with what they are asked. A value of 5 means "Yes, of course" or that they completely agree with what they are asked.

Section a) of the figure shows the answers of the users when they were asked if they found useful the existence of a methodology like the one developed in this doctoral thesis. As can be seen from the answers, all the test users considered it to be useful. Some of the comments provided by the users' highlight that due to their low knowledge of the subject, the use of the methodology and the tool has saved them a lot of time both to form an idea of the needs of their application and subsequently of the solutions available. Another comment highlights that the greatest

usefulness is to have a complete list of strategies where all aspects and characteristics are evaluated and therefore comparable.

Test users were asked whether they considered that the methodology used took into account all relevant aspects of WSNs. The test users' answers are shown in the histogram in section b) of Figure 6.34. As can be seen in the graph, all of them consider that the most relevant aspects have been taken into account. In the comments of some users, they have left us new proposals of parameters, like for example, to value to include the number of nodes in the network or if the environment is going to be noisy since they could be useful parameters at the time of selecting the most suitable strategy. Other users have asked us to include some parameter that allows us to discretize according to the nature of the deployment of the network. That is, new Body Area Networks (BANs) are appearing that use new methods of synchronization, such as heartbeat-based synchronization.

Methodology valoration

Figure 6.35. Histogram of users' answers when asked their opinion about the methodology developed.

Test users were also asked if they felt that the methodology and associated tool saved them time in choosing a synchronization strategy. The answers can be seen in the histogram in section c) of the figure. As can be seen, most users considered that it did save them time. These users have left us comments saying that it would also be interesting to show strategies similar to the one chosen as secondary options so

that once one is chosen, they can be compared. The user who has marked a value 2 tells us that this tool is useful and reduces the selection time in those people who have the least knowledge on the synchronization state of the art. But that for expert users, the time saving is less.

Users have also been asked if they find it useful to have a public database with a list of parameterized synchronization strategies. As can be seen in the histogram in section d) of Figure 6.34 everyone has answered yes that it is useful for them. Several of the users' comments have highlighted the usefulness, not only of having a consultation database, but also the possibilities of being able to order them according to some parameters or being able to represent them graphically.

Finally, about the methodology, users were asked whether the solutions proposed to them met the requirements of the application. The set of answers can be seen in section e) of Figure 6.34 where the trend of responses is positive. Indeed, these users have not had time to verify if these proposed solutions, once implemented, comply with the requirements, but all of them, a priori, have ended up satisfied.

TSST valuation

Figure 6.36. Histogram of users' answers when asked their opinion about the developed Time Synchronization Search Tool (TSST).

To finalize the user ratings they were asked about their user experience with the tool. The results can be seen in Figure 6.35. Section a) of the figure shows the histogram of users' responses to the question of whether they considered an interactive web platform to be a good method for implementing the methodology. All users have considered that our implementation is a good way to execute the methodology. Some of the users have highlighted the table where the results can be sorted according to parameters. Others have highlighted the visual part of the radar chart.

Users were also asked whether the mechanisms for assigning values to parameters (slide bars) were adequate. The answers can be seen in section b) of the

figure. This time the answers are more distributed although the tendency is that they are satisfied with the solution included. Some of the users have suggested changing some slide bars to radio buttons in parameters such as network topology or message distribution. They have also suggested including the table of values near each slide bar.

Finally, users were asked whether the mechanisms for displaying the results seemed appropriate (table mode and graphical mode). The results can be seen in section c) of Figure 6.35 where it can be seen that most of them found the representation modes adequate. The suggestions in this question were generous. Some proposed to show all the results at the beginning and to have them disappear dynamically as values were entered into the filter. Others proposed that strategies that do not comply with the filter but are close to doing so, could be shown in some way. Others proposed to make a dynamic activation/deactivation of the axes in the radar chart. And others that in the table, instead of appearing the values from 1 to 10, would appear the data to which those values represent.

6.4. Conclusion

To check the correct performance of new development, it is necessary to go through a validation process. In our case, the complete validation has been done by differentiating the method validation and the methodology validation. For method validation, the process has been done through use cases. The use cases generate inputs, which after being processed using the methodology and associated tools, result in outputs. By checking whether the outputs meet the initial objectives, the correct functioning of the system can be validated. In this chapter we have made use of two use cases, all of them with the target of data fusion, but with different requirements.

In the use case of SHM, solutions were sought that would work by fulfilling a series of premises. The solution had to be implemented in specific hardware, with specific network topology and its maximum synchronization error had to be below 1 ms. The obtained solution was implemented in a real environment, fulfilling the imposed requirements and giving good results in the acquisition of the natural frequencies and the modal forms of the structure. Furthermore, it has been shown how synchronization errors affect the identification of modal forms in SHM applications. In the case of environmental monitoring, the requirements were lax, and not only has it been validated that the application works correctly with the solution offered by the methodology, but a study has been made of how the clock sources vary with temperature variations and how these variations influence the time synchronization. After the test carried out, we conclude that the method has been validated.

For the validation of the methodology, we have asked a group of test users to put into practice the application of the methodology and the use of the tool. This group of users has faced the two use cases described above in a theoretical way. Through some forms, we have been able to analyze their degree of satisfaction and the behavior of the methodology and the tool. Given the results obtained with the

group of test users, we conclude that the methodology has been validated. The use of the methodology and the tool can be extended to any user, expert or not in the field.

Therefore, after performing the verification and validation of the methodology and method we can conclude that the developed system works as expected and that the results obtained by different users have been successful.

Chapter 7

Conclusion

Contents

7.1. Dissertation conclusion	155
7.2. Future work	159

*El verdadero progreso es el que pone
la tecnología al alcance de todos*

Henry Ford

In this chapter, we present the conclusions we have reached with the development of the thesis. Also, we present possible future work that we and future researchers can carry out to improve the work of selecting time-synchronization strategies for WSN.

7.1. Dissertation conclusion

Time synchronization in WSNs and wireless IoT networks is a key element for the proper operation of many applications. Each application has different requirements related to time synchronization. This is why many synchronization strategies have been developed and published throughout the years. When WSN application developers need a synchronization strategy, they must face the difficulty of this selection. This said selection must consider many parameters, and in many cases requires prior knowledge in the subject of synchronization in WSN. Given this difficulty, we detected the need to develop a methodology to facilitate the task of selecting a time-synchronization strategy adapted to the needs of each IoT application.

After analyzing the state of the art in Chapter 3, we concluded that there are many and varied works that provide solutions to the issue of time synchronization in wireless sensor networks. The problem arises when one realizes that no one solution fits all possible applications. Although many solutions have been proposed over the

last few years, practically all of them are only applicable to specific cases. Therefore, every time new users or application developers for WSNs need a synchronization strategy, they conclude that they must either develop their own strategy adapted to their needs or make use of a third-party protocol that may not fit the requirements.

With the increase in applications and options available, the problem of time synchronization becomes increasingly complex. This variety ranges from applications that have very lax requirements concerning time synchronization to others that have very demanding requirements. A wider range of parameters must also be considered. There are trade-offs between these parameters, so in trying to maximize one another may be diminished. Due to this increase in the variety of applications and the level of demand, as well as the increase in the parameters to be considered, the number of time-synchronization experts is reduced and therefore the probability of choosing the right strategy becomes smaller.

Although all these methods of solving the time-synchronization issue are different, some agreements have been reached to address these challenges:

- The most representative targets pursued when implementing a time-synchronization strategy are three: data fusion, MAC schemes, and cooperative transmissions.
- The critical path of communication in WSN can be divided into six components: send time, access time, transmission time, propagation time, reception time and receive time.
- The hardware timestamping-based solution, that is, timestamping at the MAC layer, is currently the most successful timestamping technique.
- Until now, there has been no methodology to help users select the synchronization strategy that best suits their applications and requirements.

Thanks to the methodology developed, the user who develops applications for WSN is freed from the worry of choosing the most appropriate synchronization strategy for his/her specific case. This divides the complex problem of time synchronization in WSN into several smaller challenges. On the one hand, users will have to detect and analyze the needs and requirements of their application regarding time synchronization; while on the other hand, they will have to interpret the published work to implement such a strategy in their own network.

After the development of this doctoral thesis, the objectives proposed in the first chapter have been fulfilled. Of these objectives, we would like to highlight the following:

- An analysis of the most used clock sources in WSN has been carried out. From this analysis, it has been concluded that decisions in the early stages of design are very important since they affect future action to a great extent. For example, the selection of the appropriate clock source for the scenario and application in WSN greatly helps to reduce or simplify the software strategies that must be implemented to obtain the required time synchronization. In addition to this analysis of clock sources, the important

influence that temperature variation has on certain clock sources is also extracted.

- Several hardware platforms have been developed. These platforms have a double utility. On the one hand, they serve as support for carrying out experiments and development tests, and on the other hand, they are used as platforms where real applications with time-synchronization requirements have been developed.
- The application that motivated this work was Structural Health Monitoring. Given the special interest in these applications, a time-synchronization protocol was developed, giving good results when put into practice.
- To go deeper into the specific issue of time-synchronization effects in SHM applications, a theoretical study with associated practical simulations was made. This work allowed us to verify that synchronization errors do not affect the detection of the frequencies of a structure, but they do have a direct influence on the detection of its modal forms.
- A taxonomy has been developed to catalog both existing and future synchronization strategies. This taxonomy has allowed us to organize the many existing protocols according to certain parameters that characterize them. This taxonomy was necessary for the complete development of the methodology.
- To help in the decision to choose the most appropriate synchronization strategy in each specific application, it has been necessary to develop the Time-Synchronization Search Tool (TSST) that allows us to compare synchronization strategies. For this purpose, a radar chart has been developed where nine parameters are represented through which the synchronization strategies are characterized. The parameters we consider are power consumption, monetary cost, security, network topology, message distribution, accuracy, clock stability, software abstraction, and hardware abstraction.
- Several tools have been developed to help both the administrator and the user to execute the methodology. In the administrator's case, a form has been created from which an expert in time synchronization can incorporate new strategies. By using the radar chart and giving the appropriate values to each parameter, all the strategies are cataloged under the same scale. From the user's point of view, a WSN application developer can filter graphically, based on the radar chart parameters, the strategies that do not meet their requirements. Graphically, larger area strategies will be more flexible in their requirements, while smaller area strategies are more demanding.
- As an integrated development shared by both the administration tool and the user tool, a database has been created to store all the synchronization strategies. This database is, therefore, a point of reference for all the developments regarding time synchronization that allows the comparison of

strategies based on parameters that have been defined on a standardized scale. This database, together with the developed tools, generally improves the selection time for a synchronization strategy. If the database is sufficiently complete and updated with all the new developments that appear, expert and non-expert users can quickly evaluate the options available.

All the above milestones have served and helped in the development of a methodology that guides WSN users in the selection of a synchronization strategy that suits their needs. The methodology developed guides the users through a series of blocks that make them question the most relevant parameters that best represent a synchronization strategy. With the methodology developed, the WSN application developers focus on the problem of developing the necessary services of the application, abstracting them from the problem of choosing a time-synchronization strategy. Using an analogous example, if developers want to have a secure application, developers do not need to be security experts if they have a methodology that indicates the steps and techniques to be applied.

The combination of the methodology and the tool creates an abstraction layer for the developers of WSN applications concerning the time-synchronization strategies. WSN application developers do not need to be experts in the field of time synchronization to select the appropriate solution for their specific case. Experts and non-experts in the field get the same technical solution making use of the methodology cycle. This is possible because all the possible time-synchronization strategies have previously been parametrized and inserted into the tool database by experts. Furthermore, the use of the filtering tool reduces the time for selecting synchronization strategies. If an expert makes use of the tool, he or she would save the time used in having to search for those published works and evaluate them one by one to see which one best suits his or her needs. Through the tool, the experts who have previously inserted the protocols, have already evaluated and parameterized them, saving this time for the end-user

The main advantages of applying this methodology to WSNs are:

- It generates a fast response regarding viability. As the methodology is based on a series of steps, and the viability of a valid solution is evaluated in each step, it is not necessary to wait until the end of implementation to know if it is possible to obtain a solution that fulfills the application's requirements.
- It gives greater control to the developer. Developers participate actively in the different stages of the process, for example, introducing all the requirements and needs of the WSN application under development in the input forms of the tool.
- The tool used to execute the methodology establishes a relationship between the different parameters and their effect on time synchronization. When prioritizing the parameters, the developers know which ones have more impact than others.

It clarifies the relationship between the parameters of each synchronization strategy and the requirements of each application, making use of graphic representations in the shape of a radar chart.

A validation process has been conducted, through use cases, to check that all the developments carried out throughout this thesis are correct. Two examples of use cases of the methodology and the tool have been given with two well-differentiated scenarios and with different characteristics and requirements. The behavior of the methodology has been shown, in both cases, to give valid results to the developer. For the validation of the methodology and the tool to be complete, several researchers were asked to make use of it by dealing with the same cases of use. The researchers, with different degrees of experience in WSNs and different degrees of knowledge in the field of time synchronization, were confronted with the use cases. In all cases, all the researchers came up with solutions that fitted their needs. One of the use cases was based on an SHM application and the other on an environmental monitoring application. These applications were chosen because they are representative of applications with different time-synchronization requirements.

This thesis demonstrates that the proposed methodology guides WSN application developers in the selection of a proper time-synchronization strategy that fulfills the requirements of their application.

7.2. Future work

Once we have concluded this work of developing a methodology for the implementation of synchronization strategies for wireless sensor networks, the door is open to future work that can improve what has already been developed.

As with any global methodology and tool, its globalized use by the scientific community will allow for continuous feedback and improvement. We would like to emphasize that the scope of this doctoral thesis should not be limited to the current state of the tool. The methodology is independent of the tool, and therefore, even if the tool evolves and improves in the future, the methodology must remain valid.

At the time of publication of this thesis, the tool has a couple of dozen synchronization strategies incorporated into the database. All the parameterization of these incorporated strategies and their insertion in the database has been done by us. This has been so to allow the validation of the methodology and the tool. We would like that in the future all the strategies were filled in by the real experts, that is, the developers of the synchronization strategies themselves. If each author parameterizes his/her work and shares it through this database, in the future we could have a global database with thousands of strategies. Therefore, we propose the integration of more synchronization strategies in the tool and database. The better fed the database is, the more useful it will be for all users as there will be more options to choose from. To promote the use of the tool and make it accessible to all WSN application developers, we have released the source code and its content under the Attribution 4.0 International license (CC BY 4.0). This license allows sharing and adaptation under the terms of attribution with no additional restrictions. We

have also published all the source code of our developments in a public repository (<https://bitbucket.org/repoB105/tsst>) and the tool that executes the methodology can be accessed through a public website (<http://elb105.com/tsst>).

To be able to integrate more strategies it would be necessary to define how to become part of the community of application administrators. If any expert in the field, regardless of their residence, could remotely authenticate themselves and add strategies, the platform would grow and be useful globally. That is why in the future it is planned to develop an authentication system from which users of the tool can register. Besides, once the users are registered, they will be able to acquire different roles, either the administrator/expert or the developer/end-user.

Another improvement that could be addressed in the future would be the management of feedback from all roles. This way end-users who implement synchronization strategies could help synchronization strategy developers with their feedback and experiences. But they would also be helping other application developers who may be facing the same implementation difficulties they have.

As in every new development, the best way to debug and improve it is through its use. That is why we believe that making use of the methodology and its associated tools in real and practical cases would improve its usability and functionality. Making practical executions allows us to evolve, either with changes, improvements or new developments that help increase its functionality. The increase of the functionality could go in the line of increasing the automation of decisions or improving the parameterization detailing more certain aspects or parameters.

Improvements in the display of results could be implemented in the current tool. A method could be implemented to show those strategies that have been discarded for not complying with the limits set but are very close to the filter created. In this way, the user can weigh up whether by reducing the level of demand on certain parameters he/she can obtain a solution more in line with his/her needs. Also, considering that in the future there will be thousands of possible solutions, a mechanism could be developed so that only a limited number of solutions are shown on the radar graph. In this way, only the solutions closest to the defined filter would be shown.

The tool could also be extended by giving it more autonomy, for example, by incorporating algorithms that help the user in the choice of solution to be implemented. These algorithms, in combination with the above-mentioned feedback, could suggest solutions based on success stories or the experiences of other users.

Also, we would like to highlight the high growth that the issue of security in time synchronization in WSN is having. Given this growing interest, probably prompted by the resolution of other more basic challenges, we would like to improve the assessment of the security parameter for our methodology and tools. Probably during the next years, there will be many advances in this line that should be considered.

Finally, given the evolution of WSNs in the past, it is not surprising that they continue to evolve and create more powerful microcontrollers with greater resources and lower power consumption. These new developments in combination with other technologies not assessed in this thesis, such as 5G, may give rise to future new

solutions concerning time synchronization. It is, therefore, an important task to keep an eye on future new developments that may appear. We can imagine that cognitive synchronization strategies will appear in the future. That is, synchronization strategies capable of dynamically modifying their parameters at run time. The methodology presented in this doctoral thesis would still be valid, but the tool would have to be adapted to support these types of dynamic synchronization strategies.

References

Teme al hombre de un solo libro
Santo Tomás de Aquino

- [1] “Reloj - Wikipedia, la enciclopedia libre.” [Online]. Available: <https://es.wikipedia.org/wiki/Reloj>. [Accessed: 15-Jan-2020].
- [2] “Category:Clocks - Wikipedia.” [Online]. Available: <https://en.wikipedia.org/wiki/Category:Clocks>. [Accessed: 15-Jan-2020].
- [3] “Second - Wikipedia.” [Online]. Available: <https://en.wikipedia.org/wiki/Second>. [Accessed: 15-Jan-2020].
- [4] “Relativity of simultaneity - Wikipedia.” [Online]. Available: https://en.wikipedia.org/wiki/Relativity_of_simultaneity. [Accessed: 15-Jan-2020].
- [5] “IEEE 802.11 - Wikipedia, la enciclopedia libre.” [Online]. Available: https://es.wikipedia.org/wiki/IEEE_802.11. [Accessed: 15-Jan-2020].
- [6] “IEEE 802.15 - Wikipedia.” [Online]. Available: https://en.wikipedia.org/wiki/IEEE_802.15. [Accessed: 15-Jan-2020].
- [7] P. Ranganathan and K. Nygard, “Time synchronization in wireless sensor networks: a survey 2,” *Int. J. UbiComp*, vol. 1, no. 2, p. 11, 2010.
- [8] J. Elson and D. Estrin, “Time synchronization for wireless sensor networks,” in *Proceedings 15th International Parallel and Distributed Processing Symposium. IPDPS 2001*, 2001, vol. 14, no. C, pp. 1965–1970, doi: 10.1109/IPDPS.2001.925191.
- [9] E. H. Callaway, “Ch 1. Introduction to Wireless Sensor Networks,” in *Wireless Sensor Networks: Architectures and Protocols*, 2004.
- [10] “Strategy Analytics: Internet of Things Now Numbers 22 Billion Devices But Where Is The Revenue? | Strategy Analytics Online Newsroom.” [Online]. Available: <https://news.strategyanalytics.com/press-release/iot-ecosystem/strategy-analytics-internet-things-now-numbers-22-billion-devices-where>. [Accessed: 15-Jan-2020].
- [11] “Gartner Says 8.4 Billion Connected ‘Things’ Will Be in Use in 2017, Up 31 Percent From 2016.” [Online]. Available: <https://www.gartner.com/en/newsroom/press-releases/2017-02-07-gartner-says-8-billion-connected-things-will-be-in-use-in-2017-up-31-percent-from-2016>. [Accessed: 15-Jan-2020].

- [12] “State of the IoT 2018: Number of IoT devices now at 7B – Market accelerating.” [Online]. Available: <https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/>. [Accessed: 15-Jan-2020].
- [13] “40 Internet Of Things Statistics From 2020 To Justify The Rise Of IoT.” [Online]. Available: <https://techjury.net/stats-about/internet-of-things-statistics/>. [Accessed: 15-Jan-2020].
- [14] “2018 Roundup Of Internet Of Things Forecasts And Market Estimates.” [Online]. Available: <https://www.forbes.com/sites/louiscolombus/2018/12/13/2018-roundup-of-internet-of-things-forecasts-and-market-estimates/#4359e7a27d83>. [Accessed: 15-Jan-2020].
- [15] M. T. Lazarescu, “Design of a WSN Platform for Long-Term Environmental Monitoring for IoT Applications,” *IEEE J. Emerg. Sel. Top. Circuits Syst.*, vol. 3, no. 1, pp. 45–54, Mar. 2013, doi: 10.1109/JETCAS.2013.2243032.
- [16] L. Liu and W. Wu, “ECTSA: Energy-Efficient Clustering Time Synchronization Algorithm for Sleeping-Based MAC Protocol,” in *2009 International Conference on Advanced Computer Control*, 2009, pp. 111–115, doi: 10.1109/ICACC.2009.95.
- [17] M. Anusha, S. Vemuru, and T. Gunasekhar, “TDMA-based MAC protocols for scheduling channel allocation in multi-channel wireless mesh networks using cognitive radio,” in *2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015]*, 2015, vol. 34, no. 7, pp. 1–5, doi: 10.1109/ICCPCT.2015.7159517.
- [18] M. Buevich, N. Rajagopal, and A. Rowe, “Hardware Assisted Clock Synchronization for Real-Time Sensor Networks,” in *2013 IEEE 34th Real-Time Systems Symposium*, 2013, pp. 268–277, doi: 10.1109/RTSS.2013.34.
- [19] A. Marco, R. Casas, J. Sevillano Ramos, V. Coarasa, A. Asensio, and M. Obaidat, “Synchronization of multihop wireless sensor networks at the application layer,” *IEEE Wirel. Commun.*, vol. 18, no. 1, pp. 82–88, Feb. 2011, doi: 10.1109/MWC.2011.5714029.
- [20] T. Lennvall, J. Akerberg, E. Hansen, and K. Yu, “A new wireless sensor network TDMA timing synchronization protocol,” in *2016 IEEE 14th International Conference on Industrial Informatics (INDIN)*, 2016, pp. 606–611, doi: 10.1109/INDIN.2016.7819233.
- [21] D. Capriglione, D. Casinelli, and L. Ferrigno, “Analysis of quantities influencing the performance of time synchronization based on linear regression in low cost WSNs,” *Measurement*, vol. 77, pp. 105–116, Jan. 2016, doi: 10.1016/j.measurement.2015.08.039.
- [22] Hewlett-Packard, “Fundamentals of Quartz Oscillators,” 1997.
- [23] D. Djenouri and M. Bagaa, “Synchronization Protocols and Implementation Issues in Wireless Sensor Networks: A Review,” *IEEE Syst. J.*, vol. 10, no. 2, pp. 617–627, Jun. 2016, doi: 10.1109/JSYST.2014.2360460.
- [24] A. R. Swain and R. C. Hansdah, “A model for the classification and survey of clock synchronization protocols in WSNs,” *Ad Hoc Networks*, vol. 27, pp. 219–241, Apr. 2015, doi: 10.1016/j.adhoc.2014.11.021.

- [25] V. Paxson, G. Almes, J. Mahdavi, and M. Mathis, "RFC2330," 1998.
- [26] B. Sundararaman, U. Buy, A. D. Kshemkalyani, and S. M. Street, "Clock Synchronization for Wireless Sensor Networks : A Survey," *Ad Hoc Networks*, pp. 281–323, 2005.
- [27] E. A. Lee and S. A. Seshia, *Introduction to embedded systems*. 2011.
- [28] P. Horowitz and W. Hill, *The Art of Electronics*. 2015.
- [29] J. R. Schodowski, S. S.; Filler, R. L.; Messina, J. A.; Rosati, V. J.; Vig, "Microcomputer-Compensated crystal oscillator for low-power clocks," *U.S. Army Electron. Technol. Devices Lab*, 1989.
- [30] K. E. Wojciechowski and R. H. Olsson, "A Fully Integrated Oven Controlled Microelectromechanical Oscillator—Part II: Characterization and Measurement," *J. Microelectromechanical Syst.*, vol. 24, no. 6, pp. 1795–1802, Dec. 2015, doi: 10.1109/JMEMS.2015.2441045.
- [31] "Crystal Oscillators: The Complete Guide (OCXO, TCXO, VCXO, & Clocks)." [Online]. Available: <https://blog.bliley.com/choosing-best-quartz-crystal-oscillator>. [Accessed: 15-Jan-2020].
- [32] "PPM Clock accuracy examples." [Online]. Available: <https://www.best-microcontroller-projects.com/ppm.html>. [Accessed: 15-Jan-2020].
- [33] M. L. Sichitiu and C. Veerarittiphan, "Simple, accurate time synchronization for wireless sensor networks," in *2003 IEEE Wireless Communications and Networking, 2003. WCNC 2003.*, 2003, vol. 2, no. C, pp. 1266–1273, doi: 10.1109/WCNC.2003.1200555.
- [34] R. T. Rajan, M. Bentum, and A. J. Boonstra, "Synchronization for space based ultra low frequency interferometry," *IEEE Aerosp. Conf. Proc.*, 2013, doi: 10.1109/AERO.2013.6496931.
- [35] 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, doi: 10.1109/MSP.2010.938757.
- [36] "Network Time Protocol - Wikipedia." [Online]. Available: https://en.wikipedia.org/wiki/Network_Time_Protocol. [Accessed: 15-Jan-2020].
- [37] "Global Positioning System - Wikipedia." [Online]. Available: https://en.wikipedia.org/wiki/Global_Positioning_System. [Accessed: 15-Jan-2020].
- [38] "Coordinated Universal Time - Wikipedia." [Online]. Available: https://en.wikipedia.org/wiki/Coordinated_Universal_Time. [Accessed: 15-Jan-2020].
- [39] S. M. Lasassmeh and J. M. Conrad, "Time Synchronization in Wireless Sensor Networks: A Survey," *Comput. Eng.*, p. 4, 2010.
- [40] D. R. Fuhrmann, J. Stomberg, S. Nooshabadi, D. McIntire, and W. Merrill, "Node Synchronization in a Wireless Sensor Network Using Unreliable GPS Signals," in *2014 IEEE Military Communications Conference*, 2014, pp. 630–636, doi: 10.1109/MILCOM.2014.111.
- [41] R. Kim, T. Nagayama, H. Jo, and B. F. Spencer, Jr., "Preliminary study of low-cost GPS receivers for time synchronization of wireless sensors," in

- Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems 2012*, 2012, vol. 8345, p. 83451A, doi: 10.1117/12.915394.
- [42] V. Le Cam, A. Bouche, and D. Pallier, “Wireless sensors synchronization: An accurate and deterministic GPS-based algorithm,” *Struct. Heal. Monit. 2017 Real-Time Mater. State Aware. Data-Driven Saf. Assur. - Proc. 11th Int. Work. Struct. Heal. Monit. IWSHM 2017*, vol. 1, pp. 1346–1354, 2017.
- [43] K. Y. Koo, D. Hester, and S. Kim, “Time Synchronization for Wireless Sensors Using Low-Cost GPS Module and Arduino,” *Front. Built Environ.*, vol. 4, no. January, pp. 1–16, Jan. 2019, doi: 10.3389/fbuil.2018.00082.
- [44] “Precision Time Protocol - Wikipedia.” [Online]. Available: https://en.wikipedia.org/wiki/Precision_Time_Protocol. [Accessed: 15-Jan-2020].
- [45] “Time-Sensitive Networking - Wikipedia.” [Online]. Available: https://en.wikipedia.org/wiki/Time-Sensitive_Networking. [Accessed: 15-Jan-2020].
- [46] H. Cho, J. Jung, B. Cho, Y. Jin, S. W. Lee, and Y. Baek, “Precision time synchronization using IEEE 1588 for wireless sensor networks,” *Proc. - 12th IEEE Int. Conf. Comput. Sci. Eng. CSE 2009*, vol. 2, pp. 579–586, 2009, doi: 10.1109/CSE.2009.264.
- [47] B. Lv *et al.*, “Simulation and Performance Analysis of the IEEE1588 PTP with Kalman Filtering in Multi-hop Wireless Sensor Networks,” *J. Networks*, vol. 9, no. 12, pp. 3445–3453, Dec. 2014, doi: 10.4304/jnw.9.12.3445-3453.
- [48] A. Sharma and A. Malhotra, “Towards Standardization in Time Synchronization of Wireless Sensor Networks,” in *Reengineering & Reinventing 4Ms (Men, Machinery Material and Money) for 5Ms (Manufacturing, Merging, Modifying and Managing Global Organizations)*, 2015.
- [49] A. Garg, A. Yadav, A. Sikora, and A. S. Sairam, “Wireless Precision Time Protocol,” *IEEE Commun. Lett.*, vol. 22, no. 4, pp. 812–815, 2018, doi: 10.1109/LCOMM.2017.2781706.
- [50] S. F. Bush and G. Mantelet, “Industrial Wireless Time-Sensitive Networking: RFC on the Path Forward,” *Avnu Alliance® White Pap.*, pp. 1–12, 2018.
- [51] A. Mildner, “Time Sensitive Networking for Wireless Networks -- A State of the Art Analysis,” in *Proceedings Of The Seminar Innovative Internet Technologies And Mobile Communications (Iitm), Winter Semester 2018/2019*, 2019, doi: 10.2313/net-2019-06-1_07.
- [52] M. Aickin, “The Importance of Early Phase Research,” *J. Altern. Complement. Med.*, vol. 13, no. 4, pp. 447–450, May 2007, doi: 10.1089/acm.2007.6370.
- [53] B. J. Kolltveit and K. Grønhaug, “The importance of the early phase: The case of construction and building projects,” *Int. J. Proj. Manag.*, vol. 22, no. 7, pp. 545–551, Oct. 2004, doi: 10.1016/j.ijproman.2004.03.002.
- [54] M. Navia, J. C. Campelo, A. Bonastre, R. Ors, and J. V. Capella, “Drift clock analysis on distributed embedded systems for IoT applications,” *Work. Innov. Inf. Commun. Technol. (ITACA-WIICT 2016)*, no. June, pp. 42–49, 2016.

- [55] J.-J. Gagnepain, "Sensitivity of quartz oscillators to the environment: characterization methods and pitfalls," *IEEE Trans. Ultrason. Ferroelectr. Freq. Control*, vol. 37, no. 5, pp. 347–354, Sep. 1990, doi: 10.1109/58.105240.
- [56] K. Sundaresan, P. E. Allen, and F. Ayazi, "Process and Temperature Compensation in a 7-MHz CMOS Clock Oscillator," *IEEE J. Solid-State Circuits*, vol. 41, no. 2, pp. 433–442, Feb. 2006, doi: 10.1109/JSSC.2005.863149.
- [57] J. M. Castillo-Secilla, J. Manuel Palomares, and J. Olivares, "Temperature-compensated clock skew adjustment," *Sensors (Switzerland)*, vol. 13, no. 8, pp. 10981–11006, 2013, doi: 10.3390/s130810981.
- [58] J. A. Boyle, J. S. Reeve, and A. S. Weddell, "DiStiNCT: Synchronizing Nodes with Imprecise Timers in Distributed Wireless Sensor Networks," *IEEE Trans. Ind. Informatics*, vol. 13, no. 3, pp. 938–946, 2017, doi: 10.1109/TII.2017.2689330.
- [59] E. COCA and V. POPA, "A Practical Solution for Time Synchronization in Wireless Sensor Networks," *Adv. Electr. Comput. Eng.*, vol. 12, no. 4, pp. 57–62, 2012, doi: 10.4316/AECE.2012.04009.
- [60] H. Mohammadmoradi, O. Gnawali, N. Rattner, A. Terzis, and A. Szalay, "Robust time synchronization in wireless sensor networks using real time clock," *Proc. 12th ACM Conf. Embed. Netw. Sens. Syst. - SenSys '14*, pp. 356–357, 2014, doi: 10.1145/2668332.2668365.
- [61] H. Mohammadmoradi, O. Gnawali, and A. Szalay, "Accurately initializing real time clocks to provide synchronized time in sensor networks," in *2017 International Conference on Computing, Networking and Communications (ICNC)*, 2017, pp. 486–490, doi: 10.1109/ICCNC.2017.7876177.
- [62] V. Di Lecce, A. Amato, and M. Calabrese, "GPS-aided lightweight architecture to support multi-sensor data synchronization," in *2008 IEEE Instrumentation and Measurement Technology Conference*, 2008, pp. 149–154, doi: 10.1109/IMTC.2008.4547021.
- [63] "Web of Science - Result Analysis of 'time synchronization in wireless sensor networks.'" [Online]. Available: https://wcs.webofknowledge.com/RA/analyze.do?product=UA&SID=E1ceyZ9gE1GNxfgFFkb&field=PY_PublicationYear_PublicationYear_en&yearSort=true. [Accessed: 16-Jan-2020].
- [64] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *ACM SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, p. 147, Dec. 2002, doi: 10.1145/844128.844143.
- [65] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Proceedings of the first international conference on Embedded networked sensor systems - SenSys '03*, 2003, p. 12, doi: 10.1145/958507.958508.
- [66] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi, "The flooding time synchronization protocol," in *Proceedings of the 2nd international conference on Embedded networked sensor systems SenSys 04*, 2004, p. 39, doi: 10.1145/1031495.1031501.

- [67] A. Mahmood and R. Jäntti, "Time synchronization accuracy in real-time wireless sensor networks," *Proc. - MICC 2009 2009 IEEE 9th Malaysia Int. Conf. Commun. with a Spec. Work. Digit. TV Contents*, no. December, pp. 652–657, 2009, doi: 10.1109/MICC.2009.5431415.
- [68] F. Ferrari and M. Zimmerling, "Efficient network flooding and time synchronization with Glossy," *Inf. Process. Sens. Networks (IPSN), 2011 10th Int. Conf.*, 2011.
- [69] J. He, P. Cheng, L. Shi, and J. Chen, "Time synchronization in WSNs: A maximum value based consensus approach," in *IEEE Conference on Decision and Control and European Control Conference*, 2011, vol. 59, no. 3, pp. 7882–7887, doi: 10.1109/CDC.2011.6161443.
- [70] P. Sommer and R. Wattenhofer, "Gradient Clock Synchronization in Wireless Sensor Networks," in *2009 International Conference on Information Processing in Sensor Networks*, 2009.
- [71] C. Lenzen, P. Sommer, and R. Wattenhofer, "PulseSync: An efficient and scalable clock synchronization protocol," *IEEE/ACM Trans. Netw.*, vol. 23, no. 3, pp. 717–757, 2015, doi: 10.1109/TNET.2014.2309805.
- [72] K. S. J. Pister and L. Doherty, "TSMP: Time synchronized mesh protocol," in *Proceedings of the IASTED International Symposium. Distributed Sensor Networks (DSN 2008)*, 2008, pp. 391–398.
- [73] S. Ganeriwal, I. Tsigkogiannis, H. Shim, V. Tsiatsis, M. B. Srivastava, and D. Ganesan, "Estimating Clock Uncertainty for Efficient Duty-Cycling in Sensor Networks," *IEEE/ACM Trans. Netw.*, vol. 17, no. 3, pp. 843–856, Jun. 2009, doi: 10.1109/TNET.2008.2001953.
- [74] T. Schmid, Z. Charbipercwala, R. Shea, and M. B. Srivastava, "Temperature Compensated Time Synchronization," *IEEE Embed. Syst. Lett.*, vol. 1, no. 2, pp. 37–41, Aug. 2009, doi: 10.1109/LES.2009.2028103.
- [75] Z. Yang, L. Cai, Y. Liu, and J. Pan, "Environment-aware clock skew estimation and synchronization for wireless sensor networks," in *2012 Proceedings IEEE INFOCOM*, 2012, pp. 1017–1025, doi: 10.1109/INFCOM.2012.6195457.
- [76] C. A. Boano, M. Zuniga, J. Brown, U. Roedig, C. Keppitiyagama, and K. Romer, "TempLab: A testbed infrastructure to study the impact of temperature on wireless sensor networks," in *IPSN-14 Proceedings of the 13th International Symposium on Information Processing in Sensor Networks*, 2014, pp. 95–106, doi: 10.1109/IPSN.2014.6846744.
- [77] M. Akhlaq and T. R. Sheltami, "RTSP: An accurate and energy-efficient protocol for clock synchronization in WSNs," *IEEE Trans. Instrum. Meas.*, vol. 62, no. 3, pp. 578–589, 2013, doi: 10.1109/TIM.2012.2232472.
- [78] L. Narula and T. E. Humphreys, "Requirements for Secure Clock Synchronization," *IEEE J. Sel. Top. Signal Process.*, vol. 12, no. 4, pp. 749–762, 2018, doi: 10.1109/JSTSP.2018.2835772.
- [79] T. Qiu, X. Liu, M. Han, H. Ning, and D. O. Wu, "A Secure Time Synchronization Protocol Against Fake Timestamps for Large-Scale Internet of Things," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 1879–1889, Dec. 2017,

- doi: 10.1109/JIOT.2017.2714904.
- [80] W. Dong and X. Liu, “Robust and secure time-synchronization against Sybil attacks for sensor networks,” *IEEE Trans. Ind. Informatics*, vol. 11, no. 6, pp. 1482–1491, 2015, doi: 10.1109/TII.2015.2495147.
- [81] J. He, J. Chen, P. Cheng, and X. Cao, “Secure Time Synchronization in WirelessSensor Networks: A MaximumConsensus-Based Approach,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 4, pp. 1055–1065, Apr. 2014, doi: 10.1109/TPDS.2013.150.
- [82] F. Sivrikaya and B. Yener, “Time synchronization in sensor networks: a survey,” *IEEE Netw.*, vol. 18, no. 4, pp. 45–50, Jul. 2004, doi: 10.1109/MNET.2004.1316761.
- [83] O. Simeone, U. Spagnolini, Y. Bar-Ness, and S. Strogatz, “Distributed synchronization in wireless networks,” *IEEE Signal Process. Mag.*, vol. 25, no. 5, pp. 81–97, Sep. 2008, doi: 10.1109/MSP.2008.926661.
- [84] X. Wang, D. Jeske, and E. Serpedin, “An overview of a class of clock synchronization algorithms for wireless sensor networks: A statistical signal processing perspective,” *Algorithms*, vol. 8, no. 3, pp. 590–620, 2015, doi: 10.3390/a8030590.
- [85] D. D. Geetha and N. Tabassum, “A survey on clock synchronization protocols in wireless sensor networks,” in *2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon)*, 2017, vol. 37, no. 1, pp. 504–509, doi: 10.1109/SmartTechCon.2017.8358424.
- [86] A. Mahmood, R. Exel, H. Trsek, and T. Sauter, “Clock Synchronization Over IEEE 802.11—A Survey of Methodologies and Protocols,” *IEEE Trans. Ind. Informatics*, vol. 13, no. 2, pp. 907–922, Apr. 2017, doi: 10.1109/TII.2016.2629669.
- [87] M. M. Rahman, H. E. Baidoo-Williams, R. Mudumbai, and S. Dasgupta, “Fully wireless implementation of distributed beamforming on a software-defined radio platform,” 2014, pp. 305–315, doi: 10.1109/ipsn.2012.6920945.
- [88] S. Felici-Castell, E. A. Navarro, J. J. Pérez-Solano, J. Segura-García, and M. García-Pineda, “Practical considerations in the implementation of collaborative beamforming on wireless sensor networks,” *Sensors (Switzerland)*, vol. 17, no. 2, Feb. 2017, doi: 10.3390/s17020237.
- [89] B. Béjar Haro, S. Zazo, and D. P. Palomar, “Energy efficient collaborative beamforming in wireless sensor networks,” *IEEE Trans. Signal Process.*, vol. 62, no. 2, pp. 496–510, Jan. 2014, doi: 10.1109/TSP.2013.2288080.
- [90] F. Tirado-Andrés and A. Araujo, “Performance of clock sources and their influence on time synchronization in wireless sensor networks,” *Int. J. Distrib. Sens. Networks*, vol. 15, no. 9, p. 155014771987937, Sep. 2019, doi: 10.1177/1550147719879372.
- [91] A. Araujo *et al.*, “Wireless Measurement System for Structural Health Monitoring With High Time-Synchronization Accuracy,” *IEEE Trans. Instrum. Meas.*, p. 9, 2011, doi: 10.1109/TIM.2011.2170889.
- [92] H. Zimmermann, “OSI Reference Model-The ISO Model of Architecture for Open Systems Interconnection,” *IEEE Trans. Commun.*, vol. 28, no. 4, pp.

- 425–432, 1980, doi: 10.1109/TCOM.1980.1094702.
- [93] K. E. Howell, *An introduction to the philosophy of methodology*. .
 - [94] D. Resner, A. A. Frohlich, and L. F. Wanner, “Speculative Precision Time Protocol: Submicrosecond clock synchronization for the IoT,” *IEEE Int. Conf. Emerg. Technol. Fact. Autom. ETFA*, vol. 2016-Novem, 2016, doi: 10.1109/ETFA.2016.7733533.
 - [95] T. Schmid, P. Dutta, and M. B. Srivastava, “High-resolution, low-power time synchronization an oxymoron no more,” in *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks - IPSN '10*, 2010, p. 151, doi: 10.1145/1791212.1791231.
 - [96] “TinyOS Home Page.” [Online]. Available: <http://www.tinyos.net/>. [Accessed: 16-Jan-2020].
 - [97] T. Kunz and E. McKnight-Macneil, “Implementing clock synchronization in WSN: CS-MNS vs. FTSP,” in *International Conference on Wireless and Mobile Computing, Networking and Communications*, 2011, pp. 157–164, doi: 10.1109/WiMOB.2011.6085410.
 - [98] “FreeRTOS - Market leading RTOS (Real Time Operating System) for embedded systems with Internet of Things extensions.” [Online]. Available: <https://www.freertos.org/>. [Accessed: 16-Jan-2020].
 - [99] I. Val, A. Arriola, C. Cruces, R. Torrego, E. Gomez, and X. Arizkorreta, “Time-synchronized Wireless Sensor Network for structural health monitoring applications in railway environments,” *2015 IEEE World Conf. Fact. Commun. Syst.*, pp. 1–9, 2015, doi: 10.1109/WFCS.2015.7160550.
 - [100] A. Leva and F. Terraneo, “Low power synchronisation in wireless sensor networks via simple feedback controllers: The FLOPSYNC scheme,” ... *Control Conf. (ACC)*, 2013, pp. 5017–5022, 2013.
 - [101] “Miosix embedded OS.” [Online]. Available: <http://miosix.org/>. [Accessed: 16-Jan-2020].
 - [102] “Embedded Parallel Operating System | HomePage.” [Online]. Available: <https://epos.lisha.ufsc.br/HomePage>. [Accessed: 16-Jan-2020].
 - [103] Q. M. Chaudhari and E. Serpedin, “Clock Offset and Skew Estimation in Wireless Sensor Networks with Known Deterministic Delays and Exponential Nondeterministic Delays,” in *2008 The Third International Conference on Digital Telecommunications (icdt 2008)*, 2008, pp. 37–40, doi: 10.1109/ICDT.2008.8.
 - [104] H. Wang, H. Zeng, and P. Wang, “Clock Skew Estimation of Listening Nodes with Clock Correction upon Every Synchronization in Wireless Sensor Networks,” *IEEE Signal Process. Lett.*, vol. 22, no. 12, pp. 2440–2444, 2015, doi: 10.1109/LSP.2015.2491320.
 - [105] “JavaScript - Wikipedia.” [Online]. Available: <https://en.wikipedia.org/wiki/JavaScript>. [Accessed: 23-Jan-2020].
 - [106] “HTML - Wikipedia.” [Online]. Available: <https://en.wikipedia.org/wiki/HTML>. [Accessed: 23-Jan-2020].
 - [107] “Node.js.” [Online]. Available: <https://nodejs.org/es/>. [Accessed: 23-Jan-2020].

-
- [108] “CockroachDB: Distributed SQL.” [Online]. Available: <https://www.cockroachlabs.com>. [Accessed: 23-Jan-2020].
 - [109] “jQuery: jQuery, Javascript and HTML5 UI framework for building web and mobile apps.” [Online]. Available: <https://www.jquery.com/>. [Accessed: 23-Jan-2020].
 - [110] “Highcharts, Interactive JavaScript charts for your webpage.” [Online]. Available: <https://www.highcharts.com/>. [Accessed: 24-Jan-2020].
 - [111] V. Krishnamurthy, K. Fowler, and E. Sazonov, “The effect of time synchronization of wireless sensors on the modal analysis of structures,” *Smart Mater. Struct.*, vol. 17, no. 5, 2008.
 - [112] S. Kim *et al.*, “Health Monitoring of Civil Infrastructures Using Wireless Sensor Networks,” in *Proceedings of the 6th International Symposium on Information Processing in Sensor Networks*, 2007, pp. 254–263, doi: 10.1109/IPSNS.2007.4379685.
 - [113] J. Li, K. A. Mechitov, R. E. Kim, and B. F. Spencer, “Efficient time synchronization for structural health monitoring using wireless smart sensor networks,” *Struct. Control Heal. Monit.*, vol. 23, no. 3, pp. 470–486, 2016, doi: 10.1002/stc.1782.
 - [114] M. J. Whelan, M. V. Gangone, K. D. Janoyan, and R. Jha, “Real-time wireless vibration monitoring for operational modal analysis of an integral abutment highway bridge,” *Eng. Struct.*, vol. 31, no. 10, pp. 2224–2235, 2009.
 - [115] E. Sazonov, V. Krishnamurthy, and R. Schilling, “Wireless Intelligent Sensor and Actuator Network - A Scalable Platform for Time-synchronous Applications of Structural Health Monitoring,” *Struct. Heal. Monit.*, vol. 9, no. 5, pp. 465–476, 2010, doi: 10.1177/1475921710370003.
 - [116] R. W. Clough and J. Penzien, *Dynamics of structures*. 1993.
 - [117] “AN3371 Application note Using the hardware real-time clock (RTC) in STM32 F0, F2, F3, F4 and L1 series of MCUs Introduction,” 2012.
 - [118] “RM0383 Reference manual STM32F411xC/E advanced Arm ®-based 32-bit MCUs,” 2018.
 - [119] Maxim Integrated, “DS3231 Extremely Accurate RTC/TCXO/Crystal.” 2015.
 - [120] ST Microelectronics, “AN2971 Application note. Typical temperature characteristics of 32 KHz crystal,” 2009.

List of Acronyms

ADC	Analog to Digital Converters
AMKF	Aided Multi-model Kalman Filter
APP	Application Layer
aTSST	administration TSST
BAN	Body Area Network
CS-MNS	Clock Sampling Mutual Network Synchronization
DOI	Digital Object Identifier
EACS	Environment-Aware Clock Synchronization
EPOS	Embedded Parallel Operating System
FEM	Finite Element Model
FM	Frecuency Modulation
FTSP	Flooding Time Synchronization
GPS	Global Positioning System
GPS-PPS	GPS Pulse Per Second
GTSP	Gradient Time Synchronization Protocol
HSE	High Speed External
IoT	Internet of Things
ISM	Industrial, Scientific and Medical
ISO	International Standards Organization
MAC	Medium Access Control Layer
MAC (depend on the context)	Modal Assurance Criteria
MCU	MicroController Unit
MCXO	Microcomputer Compensated Crystal Oscillator
MTS	Maximum Time Synchronization
NET	Network Layer
NI	National Instruments
NIST	National Office for Standardization
NTP	Network Time Protocol
OCMO	Oven Controlled Microelectromechanical Oscillators
OMA	Operational Modal Analysis
ORM	Object Relational Mapping
OSI	Open Systems Interconnection
PHY	Physical Layer
PLL	Phase-Locked Loop
ppb	parts per billion

ppm	parts per million
PTP	Precision Time Protocol
QoS	Quality of Service
RATS	Rate-Adaptative Time Synchronization
RBS	Reference Broadcast Synchronization
RPO	Recovery Point Objective
RTC	Real-Time Clock
RTSP	Recursive Time Synchronization Protocol
RTSP (depend on the context)	Robust Time-Synchronization Protocol
SHM	Structural Health Monitoring
SMTS	Secured Maximum consensus-based Time Synchronization
SQL	Structured Query Language
TCTS	Temperature Compensated Time Synchronization
TCXO	Temperature Compensated Crystal Oscillators
TDM	Time Division Multiplexing
TDMA	Time Division Multiple Access
TPSN	Timing-sync Protocol for Sensor Networks
TS	Time Synchronization
TSMP	Time Synchronized Mesh Protocol
TSN	Time-Sensitive Networking
TSST	Time-Synchronization Search Tool
TSTP	Trustful Space-Time Protocol
URL	Uniform Resource Locator
UTC	Universal Time Coordinated
uTSST	user TSST
VCXO	Voltage-Controlled Crystal Oscillator
VHT	Virtual High-resolution Time
WLAN	Wireless Local Area Network
WMTS	Weighted Maximum Time Synchronization
WN	Wireless Network
WPAN	Wireless Personal Area Networks
WSN	Wireless Sensor Networks
XO	Crystal Oscillator