IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 18, 2019, accepted June 16, 2019, date of publication June 26, 2019, date of current version July 17, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2925299

HMP: A Hybrid Monitoring Platform for Wireless Sensor Networks Evaluation

MARLON NAVIA MENDOZA[1,2], (Member, IEEE), JOSÉ CARLOS CAMPELO RIVADULLA[2], ALBERTO MIGUEL BONASTRE PINA[2], JUAN VICENTE CAPELLA HERNÁNDEZ[2], AND RAFAEL ORS CAROT[2]

[1]Grupo Siscom, Escuela Superior Politécnica Agropecuaria de Manabí (ESPAM MFL), Calceta 130601, Ecuador
[2]Instituto ITACA, Universitat Politècnica de València, 46022 Valencia, Spain

Corresponding author: Marlon Navia Mendoza (mnaviam@espam.edu.ec)

This work was supported by the Agencia Estatal de Investigación from the Spanish Ministerio de Economía, Industria y Competitividad, through the project Hacia el hospital inteligente: Investigación en el diseño de una plataforma basada en Internet de las Cosas y su aplicación en la mejora del cumplimiento de higiene de manos, under Grant DPI2016-80303-C2-1-P. The project covers the costs of publishing in open access.

ABSTRACT Wireless sensor networks (WSNs), as an essential part of the deployment of the Internet of Things paradigm, require an adequate debugging and monitoring procedures to avoid errors in their operation. One of the best tools for WSN supervision is the so-called Monitoring Platforms that harvest information about the WSN operation in order to detect errors and evaluate performance. Monitoring platforms for the WSN can be *hardware* or *software implemented*, and, additionally, they can work in *active* or *passive* mode. Each approach has advantages and drawbacks. To benefit from their advantages and compensate their limitations, *hybrid* platforms combine different approaches. However, very few hybrid tools, with many restrictions, have been proposed. Most of them are designed for a specific implementation of WSN nodes; many of them are lack of a real implementation, and none of them provides an accurate solution to synchronization issues. This paper presents a hybrid monitoring platform for WSN, called HMP. This platform combines both hardware and software, active and passive monitoring approaches. This hybridization provides many interesting capabilities; HMP harvests the information both actively (directly from the sensor nodes) and passively (by means of messages captured from the WSN), causing a very low intrusion in the observed network. In addition, HMP is reusable; it may be applied to almost any WSN and includes a suitable trace synchronism procedure. Finally, HMP follows an open architecture that allows interoperability and layered development.

INDEX TERMS Hybrid monitoring, monitoring, synchronization, distributed monitoring platform, WSN.

I. INTRODUCTION

The new advances in the development of the Internet of Things (IoT) make us focus on this approach as one of the technologies that will shape our future. The IoT paradigm includes many successful technologies, whose deployment has also been accelerated [1], [2]. Among them, WSN are one of the most widespread and studied [3]. However, despite of the high degree of development that has been achieved, every WSN implementation may suffer design, implementation and operation errors and failures. Many issues may cause these errors, such as programming errors, failures in wireless communications, malfunction of nodes, environmental effects, among others. Therefore, the operation of a WSN can be affected and failures [4], [5] or security attacks [6] may happen. There are several ways to evaluate and diagnose the functioning of a WSN. These methods may be used during the development, deployment or operation of the network. For example, simulators or emulators [7] are generally used in development phase to test new functionalities, although they have several limitations.

Monitoring equipment, evaluation systems or platforms for WSN –sometimes referred as *WSN Monitors*– facilitate the verification of the correct operation of a WSN and the location of possible errors. These platforms are usually very specific. Some of them are oriented to provide information in real time, which requires many resources and usually generates a non-negligible intrusion in the monitored network.

The associate editor coordinating the review of this manuscript and approving it for publication was Zhangbing Zhou.

Other proposals are based in the generation of a trace of events for further analysis [8].

The monitoring platforms can be classified as *active* or *passive*. Active monitoring platforms interact directly with the elements of WSN and require the nodes to provide its internal state, and thus obtaining high precision data. However, they use resources from the observed WSN, slowing its activity and draining its electrical power. These phenomena are usually called *interference* or *intrusion*, and they affect the operation of the analyzed networks and, therefore, reduce their performance. On the other hand, passive monitoring platforms do not modify neither the operation nor the performance of the WSN, because they are based on spying the communications between the elements of the WSN and/or processing the information that reaches the sink. Nevertheless, the information provided by passive tools may be insufficient for a complete analysis.

The denomination " *hybrid monitoring*" has traditionally been used when referred to a combination of *hardware* and *software* [9]. However, the term *hybrid* may also be used for other monitoring systems that combine two different approaches, such as wired-wireless, local-global, fixed-mobile, or active-passive. The proposals presented in [10] – which combines monitoring approaches by *events* and *time* to detect vulnerabilities in software – or in [11] – a system of active-passive monitoring for WSN– are also presented as examples of hybrid monitoring.

According to the analysis made in [12], new hybrid monitoring systems may combine the advantages of different approaches to compensate for their disadvantages. However, after studying a large number of monitoring proposals, very few of them use hybrid monitoring, and those are very limited.

This paper presents a Hybrid Monitoring Platform (HMP) for WSN that combines both *hardware/ software* and *active/ passive* approaches. One of its main features is the low intrusion or interference obtained thanks to the hybrid approach. This way, software elements on the observed nodes actively provide detailed information about their behavior. In order to discharge the observed node from data processing, storage, time stamping, and transmission procedures as non-hybrid active proposals do, in our hybrid approach additional hardware elements receive these data as events and perform those functions. In addition, other sniffer devices perform passive monitoring of the communications in the observed WSN, providing a new sequence of events. Both sequences of events are recorded in log files, along with time tags, as *traces*. These traces must be synchronized in order to correctly correlate the events of different nodes, as some of them may have a causal relation. This way, HMP integrates a new trace synchronization mechanism called GTSO (Global Trace Synchronization and Ordering Mechanism). HMP includes three different hardware devices: Monitor Node, Sniffer Node, and Monitor Server. A Synchronization Server is also required to support GTSO, although it can be integrated in the Monitor Server.

The rest of this paper is structured as follows: In section 2 an analysis of the state of the art on WSN monitoring is presented. Section 3 details the proposal of the HMP. Section 4 shows the results of the evaluation of the hybrid platform and section 5 compares it with others proposals. Finally, section 6 briefs the conclusions of this proposal.

II. RELATED WORK ABOUT WSN MONITORING

There are several platforms and tools for monitoring or debugging the performance of a WSN. Most of them use an active or a passive approach, but a very few can be considered to adjust into a hybrid approach. This section outlines the most representative monitoring tools proposed in recent years. Additionally, as synchronization is a main issue to guarantee a correct ordering of the events captured in different monitor nodes, a detailed study about synchronization in distributed monitoring platforms is presented.

A. ACTIVE AND PASSIVE MONITORS

SNMS (Sensor Network Management System) [13] is one of the first and best known monitoring systems for WSN. SNMS is basically a very complete network management system based on TinyOS [14] that provides a set of services: health data collection and network status based on queries, and persistent event logging. It is also found with the name *Nucleos* in some references. Due to its capabilities and operating modes, it causes a high intrusion in the nodes [13]. Additionally, it is only suitable for sensor nodes based on TinyOS.

Memento [15] and Lightweight Tracing [16] are examples of active monitors that use simple encoding to record events and related information. The first one adds a piece of data that contains node information in the message to transmit. This way, it is able to detect problems in a node based on the information provided by a group of nodes in the network. Although the intrusion in RAM space is less than SNMS and depends on whether this code can be transmitted, intrusion in code space is high, as well as the time intrusion. The second one (Lightweight) registers events and states in a non-volatile memory, using a very light coding. Afterwards, it carries out a reconstruction and debugging of the behavior of the network. Its intrusion in memory is similar to the previous one and no data about time intrusion is provided. Moreover, a drawback of both tools is that they do not record the time when events are registered.

EnviroLog [17], NodeMD [18], PDA (Passive Distributed Assertions for Sensor Networks) [19] and TARDIS (Trace And Replay Debugging In Sensornets) [20] are proposals based on adding and activating a monitoring code in the sensor nodes. Both EnviroLog and TARDIS record events and save them in a flash memory for later reproduction, although the authors of TARDIS claim to be able to register more types of events than the other proposals. Both tools need a large memory space (up to a 25% of increase). Like many other tools, the authors provide no data about time intrusion. NodeMD creates a trace with the events that produce

a failure in a node for later analysis. Memory requirements are less than previous proposal, but NodeMD generates a time intrusion of near 80 CPU cycles per event registration. PDA emits information about the state of the node based on assertions added to the node code. As in the previous cases, it is not possible to know when the events occur in any of these four monitors. No data of intrusion is provided, only the network overload (as they use the same network to transmit the monitored information) is reported (8%).

Deployment Support Network (DSN) [21] is a platform composed of nodes connected to the WSN's nodes by a serial interface. These attached nodes allow testing, controlling and monitoring applications in real environments. The DSN nodes are connected to each other via Bluetooth. MARWIS [22] is similar to DSN, but it can be applied to heterogeneous networks, using a parallel wireless mesh network to divide the WSN into subnets, according to their characteristics. However, in addition to the intrusion caused by the capabilities of these platforms, the use of Bluetooth for connecting monitoring nodes may cause interferences in the observed WSN, and also suffers from other problems related with this technology. Unfortunately, no intrusion metrics are provided for these tools.

SNIF (Sensor Network Inspection Framework) [21], [22], Pimoto [25], LiveNet [26], SNDS (Sensor Network Distributed Sniffer) [27], NSSN (Network monitoring and packet Sniffing tool for wireless Sensor Networks) [28], EPMOSt (Energy-efficient Passive Monitoring SysTem for WSN) [29], and Z-Monitor [30] are examples of passive monitoring platforms with similar schema. They propose a network of sniffers deployed next to the WSN that captures the transmissions of the nodes. The difference between them is how the captured data is processed. Some of them transmit the data through a TCP/IP (Transport Control Protocol / Internet Protocol) network to another device for processing. In other cases, the sniffer can function as a sink, collecting and analyzing the information. They can also provide a real-time analysis of the operation data of the sensor network. However, these tools cannot obtain information directly from the nodes.

Sympathy [31] and PAD (Passive Diagnosis for WSN) [32] are presented as passive monitoring systems. However, due to their operation, they could be considered as active monitors. Both are based on the aggregation of light information to the messages transmitted by the node, for later deduction of behavior or failures detection. As before, these proposals overload the application messages of the observed nodes.

Minerva [33], FlockLab [34] and TWECIS [35] are examples of Testbeds for WSN. A Testbed does not only monitor the operation of the nodes of a WSN, but it also allows the modification of the configuration in the nodes, and even request them information about their state. However, they are usually limited to laboratory environments and thus are not suitable for a deployed WSN.

Spi-Snooper is a monitoring platform for WSN that integrates hardware and software in a hybrid approach. According to its operation mode it can work actively and passively [36]. In brief, Spi-Snooper is a monitor attached to the sensor node that "spies" the SPI (Serial-Peripheral Interface) bus between the main microcontroller and the radio peripheral controller. However, its application is not possible in nodes based on later microcontrollers with a built-in radio peripheral.

HDF (Hybrid Debugging Framework for Distributed Network Environments) [11] combines both active–the execution of orders and queries about the state of a monitored node– and passive approach –the listening of information by a device linked to the WSN node–, to carry out the debugging of a WSN in real time. However, the authors have not published information about the overload caused in a real WSN, and its operation has been tested only theoretically. From now, just one of its components – the active one– has been developed as a prototype..

As it is clear from the previous paragraphs, passive platforms that only receive messages from the network cannot be aware of what happens internally in the node, whereas active tools can access to the internal node events but produce a non-negligible intrusion in the observed WSN.

Unfortunately, most of active monitoring proposals do not consider the impact of monitoring operation on the observed WSN. The intrusion is admitted as a penalty to be paid to gather all the necessary information, at the expense of perturbing the functioning of the system under test. On the other hand, the very few implemented previously described approaches use different hardware. Finally, the intrusion has proven to be hardly dependent of the monitoring campaign, for instance, related to the number of events to register. For all these reasons, it is very difficult to perform a quantitative comparison analysis between these proposals.

Another main issue to consider is that some of the cited monitoring proposals –both active and passive– do not record the time in which the events occur. Finally, most of monitoring platforms are designed to monitor a specific type of sensor node, so they cannot to be used in WSN that use other hardware.

The main advantage of a hybrid approach is to minimize this intrusion and affect as little as possible the behavior of the system under observation.

So, according to the analysis made in [12], new hybrid monitoring systems could combine the advantages of different approaches to compensate for their disadvantages. However, after studying a large number of monitoring proposals, very few of them use hybrid monitoring, and those are very limited.

B. SYNCHRONIZATION IN WSN DISTRIBUTED MONITORING

Synchronization is one of the most important issues in any distributed system, such as distributed monitoring platforms. The different elements in the platform must coordinate their operation to provide a correct temporal view of the monitored system.

Usually, distributed monitoring platforms for WSN use *online* mechanisms for synchronization. These online synchronization methods adjust the internal clock of the components during the monitoring process. Online synchronization uses several strategies, usually based in messages exchange and calculations on timestamps, for estimating and removing of clock's skew and offset [37], [38]. In a common online mechanism, such as NTP [39], a node could receive one or more messages with information about reference clock and/or local offset, and it can use this information for adjusting its internal clock. This NTP-based approach will be used later to compare our proposal. On the other hand, *offline* synchronization techniques correct the time stamps of the data collected by the platform after performing the monitoring.

SNIF [24] and PDA [19] use a Bluetooth specific synchronization protocol. Many monitor nodes with TCP/IP capability —as Pimoto PC Gateway, sniffers of NSSN [28], components of Minerva [33] or FlockLab [34]— use the Network Time Protocol (NTP) for synchronizing. To achieve higher precision, SNDS [27] uses PTP (Precision Time Synchronization Protocol) and TWECIS uses [35] RTE (Real Time Ethernet); however, both mechanism require additional or special hardware.

Online mechanisms could introduce errors in the correct order of events because they can adjust local clocks backwards. This is due to the different variability of the clock rate, as shown in [40]. Most of the online mechanisms used in WSN monitoring offer accuracies of the order of milliseconds. This may be not suitable to record events in WSN nodes, which can happen within differences of microseconds. Besides, the more precision mechanisms, such PTP or RTE, require additional hardware and cause more overload on monitors [41], [42].

On the other hand, only LiveNet [26] and Pimoto [25] apply offline methods to provide trace synchronization. However, these methods do not consider the variability of the clock rate in different nodes. Therefore, errors in the synchronization could occur.

A synchronization mechanism is required to perform an adequate synchronization for monitoring operation. This way, it is possible to overcome the disadvantages of the aforementioned mechanisms. This proposal must not require specific hardware to be suitable for any monitoring platform, and it must offer a precision according to the observed system.

III. HYBRID MONITORING PLATFORM PROPOSAL

This section details the structure and operation of the Hybrid Monitoring Platform for WSN. First, HMP architecture, the global operation of the platform, the basic functionality of each component and the synchronization mechanism are described. Afterwards, the implementation of each component is individually detailed.

A. PLATFORM ARCHITECTURE

The architecture of HMP is based on a reference model proposed in [43], and presented in Figure 1. Using a reference model aids in the design of monitoring platforms and systematizes the comparative analysis of present and future proposals. Three layers may be identified at this model: Monitoring Layer, Information layer, and Interchange Layer. The Monitoring layer, which is located at the top of the model, deals with the abstraction of the particularities of the observed network, such as the data to acquire, the definition of the capture mechanism and how data will be shown to the user. This abstraction allows that different WSN may be observed with a common systematic approach, and most of the platform hardware may be reused for different monitoring campaigns on different WSN, even they are based on diverse technologies. More on, heterogeneous WSN – where nodes of different technologies collaborate – may be observed in a coordinated way, overcoming the hardware differences among nodes.

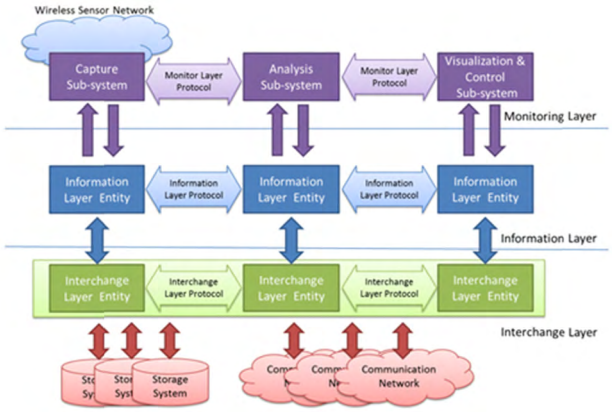


FIGURE 1. Reference model for the monitoring platform [43].

The Information layer –located below the Monitoring layer–performs the encoding of the acquired information in a standardized format, and it provides the time-related services, such as capture triggering and timestamps. The Interchange layer –in the lower part of the model– is in charge of the services related to the storage and/or transmission of the information captured in the different modules of the platform. These services allow retrieving this information for analysis and/or visualization in the Monitoring layer.

For each layer, the architecture defines interfaces to communicate with its adjacent layers. A layered architecture allows changes in one of the layers without affecting the others. This way, any improvement in a platform module should be easier to develop and implement. In addition, this model allows interoperability between components of different monitoring platforms.

HMP consists of three basic components: The Monitor Node (MN), the Sniffer Node (SN), and the Monitor Server (MS). Figure 2 shows the components of the HMP and its location in relation to a monitored WSN.

Each MN records the events of interest reported from its sensor node, adds a timestamp, and stores this information with a structured format in the trace. Later, after the end

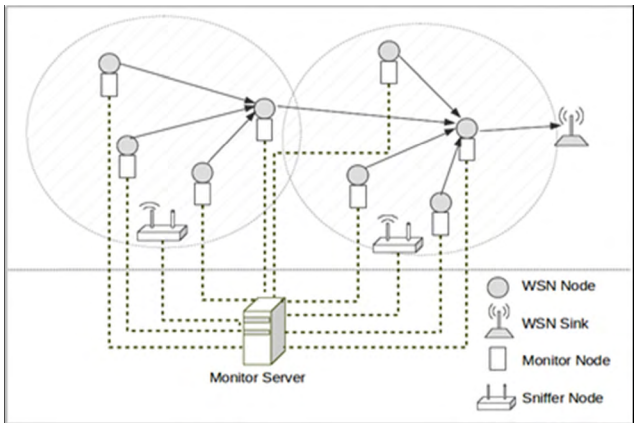


FIGURE 2. Structure and components of HMP. Components of the WSN are included.

of the monitoring campaign, the MN sends the trace to the MS.

Each SN covers a specific area of the WSN, being able to capture the messages transmitted by the sensor nodes in this area. In this sense, its function does not differ from the sniffers that use other proposals, although the SN is integrated with the rest of the HMP according with the predefined architecture. The Monitor Server collects the information obtained by the different platform elements for processing. In addition, the MS could be in charge of other functions, such as the synchronization of the obtained data as described below.

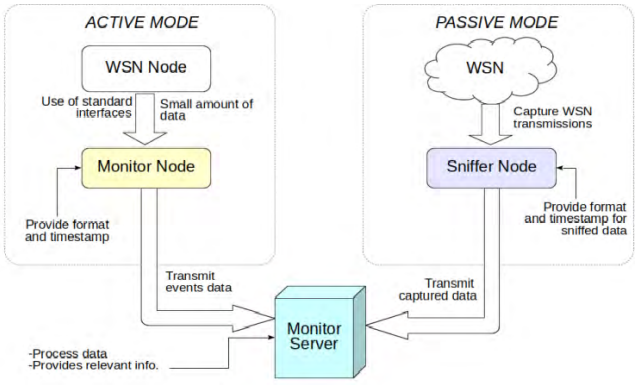


FIGURE 3. Scheme of the operation of HMP.

As told, HMP combines both active and passive monitoring mechanisms. Figure 3 illustrates the platform components that work in each of these modes, its basic operations and the data flow between them. The active component of the platform is mainly located in the sensor nodes of the observed WSN and their attached Monitor Node. Each MN connects with a very small piece of software added to each monitored sensor node, called software " *traps*". This code will be executed when an event of interest is detected. It transmits its associated code plus additional data (if required) to the MN, through one of the standard interfaces available in the WSN node. The MN receives and adds a timestamp to the received data; it formats them in a structured way and finally stores or transmits all the gathered information to the Monitor Server for processing.

The Sniffer Nodes are the passive components of the platform. Each SN captures the application messages on the air and saves them with a predefined format, which includes a timestamp. The captured messages will be transmitted later to the Monitor Server, like the MN messages.

The traces obtained in both passive and active monitors (SN and MN) must be processed by the Monitor Server (MS). Traces are merged in a single log file that reflects the relevant events in all the observed elements of the WSN, and thus its behavior. This information is stored for later processing, and may be displayed and analyzed in the MS, or be used as synthetic load for simulation campaigns.

Finally, it is interesting to highlight the possibility of using HMP in any distributed system, regardless of its specific characteristics (hardware, communication protocol, etc.). Only the SN has to be adapted with the same communication protocol used in the distributed system, to sniff the messages.

B. PLATFORM COMPONENTS DESIGN

Figure 4 shows the structure and operation of the MN. The Monitor is based on both hardware and software elements, following a hybrid approach that, as will be demonstrated later, is essential to achieve low intrusion in the WSN operation.

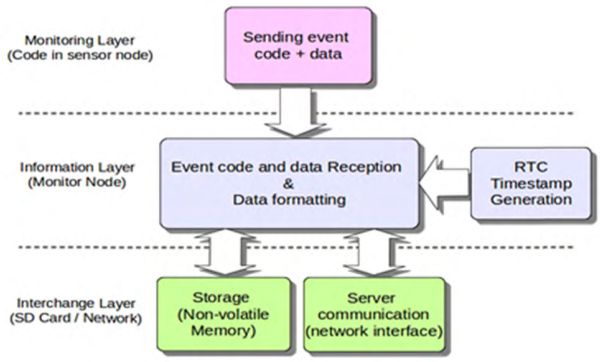


FIGURE 4. Structure and composition of the Monitor Node.

The software component is located in the Monitoring layer. The aforementioned software traps capture and send information about the events to be recorded, as defined by the user.

A trap is activated when its corresponding event is detected in the observed sensor node. Then, it sends a specific event code through a physical interface of the sensor node. This is called a software *probe*.

The hardware component of the MN mainly covers the information level. It consists of the attached node that receives the information sent from the sensor node and processes it. The Mon-Inf interface can be implemented using any transmission equipment available in the sensor node, preferably a standard interface (serial or parallel).

The Interchange layer can provide two different services. Data may be stored on a non-volatile memory – for example a Secure Digital (SD) memory–, or may be sent through a communication interface (preferably through a secondary monitoring network to avoid overload on the application network) to the Monitor Server. Both methods may be used simultaneously if required by the application.

The Information layer must register the timing of the captured events. Thus, the MN uses a Real Time Clock (RTC) for the generation of timestamps.

Following the aforementioned architecture, Figure 5 shows the components of the Sniffer Node. The Monitoring level is implemented by means of a hardware probe, which consists of a wireless transducer compatible with the WSN. It operates in promiscuous mode, with the same configuration as the monitored network (frequency, channels, signal modulation, etc.). The packets received through this probe are converted into events that are processed by the Information layer and the Interchange layer, as described in the MN.

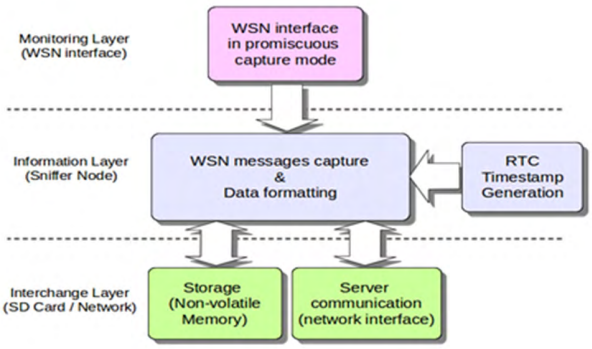


FIGURE 5. Structure and composition of the Sniffer Node.

Finally, the Monitor Server must collect all the traces generated for each element of the platform, process them and generate a single trace that reflects all the gathered information. This process involves the resynchronization of the traces that have been generated by the other elements of the HMP, and the unification of the same ones. As discussed later, the traces merging process needs a resynchronization mechanism to avoid errors in the global trace (change of order of events due to inaccuracies of the nodes clocks).

Optionally, the MS may display the results. The elements of the MS are shown in Figure 6.

C. PLATFORM SYNCHRONIZATION

HMP integrates a new trace synchronization mechanism called GTSO (Global Trace Synchronization and Ordering Mechanism). Its objective is the synchronization of the individual traces obtained by each one of the HMP elements, avoiding the drawbacks of the mechanisms cited in section 2.B.

In [45], the authors performed a study of problems and alternatives for distributed monitoring platforms, and finally proposed a new trace synchronization mechanism

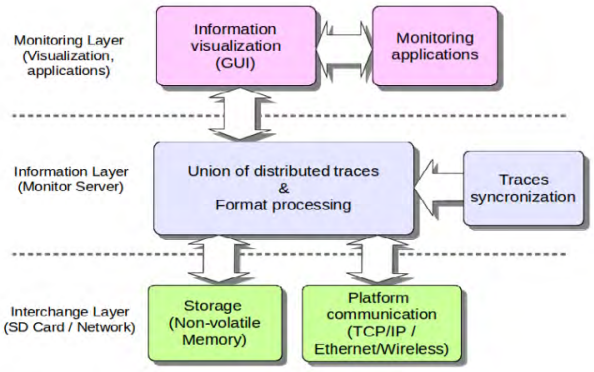


FIGURE 6. Structure and composition of the Monitor Server.

called GTSO. This mechanism is a simple offline mechanism based on the inclusion of common synchronization points in the traces generated by the components of the platform.

GTSO requires a Synchronization Server called *SyncRoot*. The function of the SyncRoot is to provide synchronism between the elements of the HMP, to achieve coherence in the sequence of events in the generated traces. The SyncRoot generates periodic events, which are received by each element of the platform and added as *synchronization points* in their trace. The SyncRoot also stores the timing of these synchronization events. The traces will be corrected (synchronized) based on these synchronization points. Although the SyncRoot can be implemented as a standalone element, in our demonstrator it has been integrated into the MS. The Figure 7 shows the operation of the mechanism.

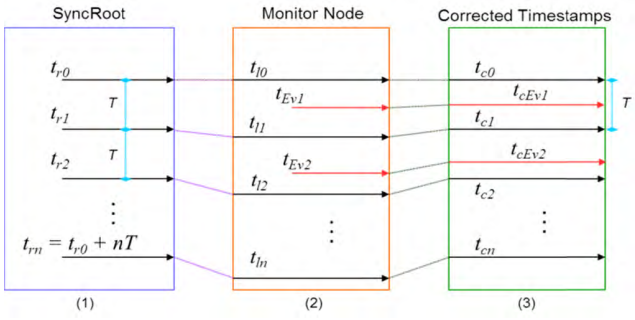


FIGURE 7. Schematic of global trace synchronization method.

The SyncRoot transmits periodically a synchronization point (step 1). The HMP element (monitor or sniffer node) inserts in its trace an event which includes the local arrival time of this point (step 2).

Using these synchronization points, the timestamp of events can be corrected (step 3). The local clock (and so the generated timestamps) can be delayed (as in Figure 7) or advanced compared with the reference time in SyncRoot.

This synchronization is performed by means of the following procedure: Let us t_{Evm} denote the timestamp of an event m, that has been recorded in a HMP node between two

consecutive local synchronization points with timestamps t_{ln-1} and t_{ln}, that is it, $t_{ln-1} \leq t_{Evm} \leq t_{ln}$, with $t_{ln-1} < t_{ln}$. The corrected timestamp of the event m, denoted as t_{CEvm}, may be calculated with the formula (1), where t_{Cn-1} is the timestamp in SyncRoot of the synchronization point recorded in t_{ln-1}, and T is the transmission period of synchronization points.

$$t_{CEvm} = t_{Cn-1} + \frac{t_{Evm} - t_{ln-1}}{t_{ln} - t_{ln-1}} T \quad (1)$$

As the drift rate may be unstable over time [40], the transmission period (T) for the synchronization points must be adjusted to the characteristics of the local clocks.

A wrong value of T may produce timing errors, such as an inversion (change) in the correct order of observed events, when these have been captured from different sources in a distributed platform, as the authors have tested previously in [40]. Therefore, T can be increased when the HMP nodes are provided with a high quality local clock, minimizing the overhead of this process, as far as the correct ordering is granted. On the other hand, when using low quality clock sources, T must be decreased to achieve the required accuracy.

D. IMPLEMENTATION OF PLATFORM COMPONENTS

1) IMPLEMENTATION OF MN

As stated above, a sensor node of the observed WSN is monitored by a small piece of code inserted in the sensor node (software part) and a Monitoring Node (MN) that receives the information transmitted by this code (hardware part). The code added to the monitored node performs two main functions: i) the initialization of the interface to communicate with the MN and ii) the software traps that capture the events (probe). A library that contains both functions and a header file with the necessary definitions and prototypes is included in the software of the sensor node. Source files required to be compiled in the monitored WSN node, are also included.

Algorithm 1 shows an example of the probe implementation added to the sensor node. A function –named *WriteLog*– sends the information byte by byte to the MN through the selected hardware interface (Mon-Inf interface). As can be seen in the *main* function of Algorithm 1, a call to the *WriteLog* function is added for each event to be captured. The information sent for each call includes the appropriate event code and the additional data related to the event of interest. In the example shown in Algorithm 1, the registered events are the start of the sensor node execution and the transmission of a wireless message.

Each event requires a single invocation of the trap routine. An interrupt routine manages the transmission of the different bytes that identify the event and its parameters.

Regarding the implementation of the Mon-Inf interface, this MN may use either a parallel interface, through GPIO (General Purpose Input/Output) pins, or a serial interface, such as UART (Universal Asynchronous Receiver / Transmitter) or SPI bus. In [44] it has been evaluated the high influence of the chosen communication interface in the intrusion caused.

Algorithm 1 Monitoring Code in the Sensor Node

```
WriteLog( DataLong, code, msg[] )
if DataLong >0) then
    msgTx = 0 × 80 + code
    Serial_send (msgTx)
    Serial_send (DataLong)
    for ind = 0...DataLong do
      Serial0_send (msg[ind])
    end for
else
    Serial_send (msgTx)
end if
```

```
main (void) /*Sensor Node main application*/
    ...
    WriteLog(0,Log_Reset,message); //Event Reset (start)
    ...
    temperature_frame_send (message);
    WriteLog(12,Log_TxData,message); //Event TxData
    ...
```

When the MN receives an event data from the probe, the Information Level applies a pre-established format to the stored information. Each monitoring campaign may choose and define the more suitable storage format, and provide it to the Information Layer in the MN by means of a template file stored in the SD memory. When this file exists in the SD, the template is loaded and applied to each event. In other case, a generic format is applied. This format must include a *Timestamp* generated by the MN.

Then, the Interchange Level receives each event and its related data. This information must be delivered to the monitor server (MS), even being transferred (through a monitoring communication network) and/or stored (i.e. in a SD memory) for later manual handling. In our demonstrator, an XML (Extensible Markup Language) type structure has been defined and Ethernet has been used as monitoring network. However, any other wired or wireless communication interfaces would also be applicable.

To verify the right operation of the HMP, a prototype of NM was developed and incorporated into the definitive platform (Figure 8). It is based on an STM32F407 microcontroller –ARM Cortex M4– incorporated in the evaluation board STM32F4Discovery [46], alongside with an expansion board STM32F4DisBB [47]. This expansion board incorporates a micro-SD memory storage and many communications features, such as Ethernet Interface.

2) IMPLEMENTATION OF SN

The Sniffer Node was based on the same hardware that the NM prototype, plus a hardware probe. This probe is implemented by means of a wireless interface similar to the sensor nodes in observed WSN, working in promiscuous mode.

FIGURE 8. The MN incorporated into the platform.

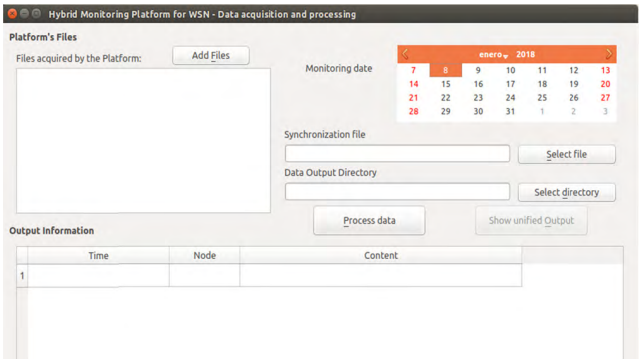


FIGURE 9. Main window of the graphic interface of the MS.

As in the MN, the Interchange level of this SN has been implemented using both communication and storage options.

3) IMPLEMENTATION OF MS

The Monitor Server may be implemented in any computer that supports the developed applications (currently running under Linux OS). The main functions of the MS are focused on the Monitoring layer, especially in the processing of the obtained traces, such as storage, synchronization and unification of the traces, automatic analysis of the obtained unified trace and visualization of the results (in both graphic and textual modes).

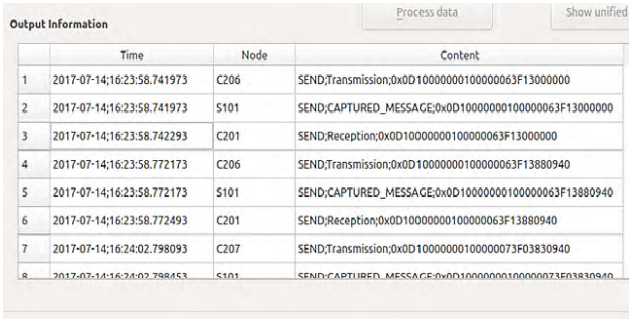
Beyond this specific application, developed for this purpose, other existing applications or systems could also be used –e.g. Wireshark [48], NAM for NS2 [49], or NetAnim for NS3 [50]–. Just a suitable plug-in must be provided to transcribe into native files the XML structures defined for the generated information.

The GUI –located in the Monitoring level in the Server– allows the control of the main MS options of the monitoring platform. This interface, like the data's processing procedures explained above, was implemented in Python programming language in an Ubuntu GNU/Linux environment. Both the synchronism messages and trace reception were performed by applications created in the C language using the GCC compiler. They are invoked from the MS main application.

Figure 9 shows the main window of the MS application, with the monitoring settings, and Figure 10 shows an example of the output with the unified trace.

IV. EVALUATION OF THE PLATFORM

Intrusion is the most important feature of monitoring tools, as far as it affects the performance of the observed system, and therefore modify the obtained measures. This section evaluates the intrusion generated by the monitoring operation of HMP in the observed WSN. First of all, the WSN under monitoring and how HMP is applied to this system is described. In second place, its operation and some results are shown. From these results, the intrusion generated by HMP is studied.



	Time	Node	Content
1	2017-07-14;16:23:58.741973	C206	SEND;Transmission;0x0D10000000100000063F13000000
2	2017-07-14;16:23:58.741973	S101	SEND;CAPTURED_MESSAGE;0x0D10000000100000063F13000000
3	2017-07-14;16:23:58.742293	C201	SEND;Reception;0x0D10000000100000063F13000000
4	2017-07-14;16:23:58.772173	C206	SEND;Transmission;0x0D10000000100000063F13880940
5	2017-07-14;16:23:58.772173	S101	SEND;CAPTURED_MESSAGE;0x0D10000000100000063F13880940
6	2017-07-14;16:23:58.772493	C201	SEND;Reception;0x0D10000000100000063F13880940
7	2017-07-14;16:24:02.798093	C207	SEND;Transmission;0x0D10000000100000073F03830940
8	2017-07-14;16:24:02.798453	S101	SEND;CAPTURED_MESSAGE;0x0D10000000100000073F03830940

FIGURE 10. Unified trace processed by the platform.

A. MONITORED WSN

A habitat monitoring WSN application has been used to evaluate HMP. In this application, WSN nodes periodically measure the ambient temperature and transmit it to a sink node. The wireless sensor nodes are based on the CC1110F32 microcontroller from Texas Instruments, a low power sub-1Ghz system-on-chip designed for low power wireless applications [51]. WSN designers decided to use the SimpliciTI RF Protocol. This protocol is provided by Texas Instruments as on open library [52] and was designed to provide an easy implementation and deployment out-of-the-box on several Texas Instruments RF platforms [51]. SimpliciTI defines three types of nodes: *End device* (ED), *Repeater or range extender* (RE), and *Access point* or *Collector* (AP). The nodes implemented as ED are provided with a temperature sensor. Other nodes adopt the role of RE, and forward the measurements to a sink node, configured as an AP.

This way, the application consists of sensor nodes (ED) that periodically (every 10 seconds) wake up from low power mode, measures the ambient temperature, reading its value from a sensor and transmits it towards a collector node (AP). After that, returns to low power mode. To reach the collector, another repeater node (RE) must forward the frame. This RE also acts as a sensor node (ED), but it does not switch to low power mode, because it has to provide the forwarding services continuously. The forwarding operation is transparent to the ED device, which is not aware of this reemission required to access the collector. The described scheme is shown in Figure 11.

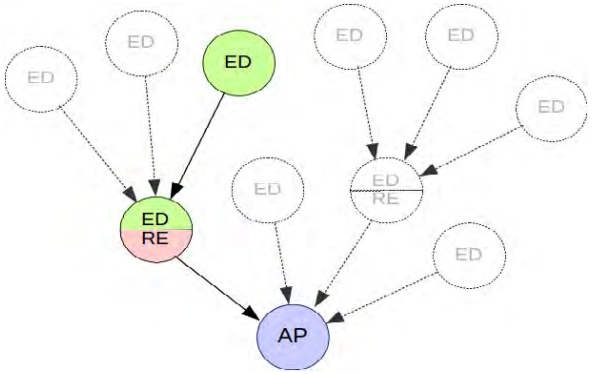


FIGURE 11. Communication scheme of the monitored WSN.

The MNs were easily connected to the observed WSN nodes by means of a standard interface as commented in next section, and the monitoring code was introduced into the SimpliciTI library in the sensor nodes; so, a slight software modification of the observed nodes is necessary. The three types of nodes involved – sensor node (ED), repeater (RE) and collector (AP) – have been monitored with the aforementioned MN.

The events observed in each one of these nodes depend on its role. In the sensor nodes (ED), four events were considered significant: data transmission (*Transmission*), data reception (*Reception*), transmission error (*Log_Error*) and wake up (*WakeUp*). The RE node remained active continuously, so *Wakeup* events were not considered; only the events of data transmission, transmission error and data reception were registered. Finally, the collector node (AP) only considered the events of reception and start of operation (*Reset*).

Moreover, to sniff wireless messages, several SN nodes were added. In this case, the SN has been adapted in order to include the appropriate *probe:* a Texas Instrument Sub-Ghz RF module with SimpliciTI routines working in promiscuous mode to capture all messages. These messages are passed to the main controller of the SN in order to fulfill the Interchange level tasks. Finally, events from SN nodes are identified in the trace by the *Captured_Message* event.

In order to obtain reliable results, experiments showed in next sections were repeated $\boldsymbol{n}$ times until an estimation of $\boldsymbol{\mu}$ (that is, the value of the measured parameter) was obtained with a 95% of confidence interval, according to Student's t-distribution with $\boldsymbol{n-1}$ degrees of freedom.

The next figures enlighten the analysis of application of HMP to observe the operation of the previously described WSN for diagnostic purposes.

Figure 12 and Figure 13 show parts of the unified trace registered by the platform. In Figure 12, a message was sent from the ED (node C207), and it was retransmitted by the ED/RE (node C206) to the sink or AP (node C201). In the highlighted lines, the transmission of the ED, the reception and retransmission by the ED/RE, as well as the captures in the SN (node S101) for both transmissions are observed. The first hop is marked in yellow. The second hop, in pink. In this second hop, it can be noticed that the reception in the sink is not observed (the next C206 node transmission two seconds later corresponds to another message). From this trace it may be deduced that the problem must be related to the reception in the sink, as it can be assured that the message was sent by the ED/RE.

An active monitoring platform would detect the transmission command in ED/RE node and lack of reception in the AP node; but it cannot be ensured that the message was really sent to the air. The source of the problem could be both the transmission in ED/RE and the reception in AP. On the other hand, a passive monitoring platform, such as a distributed sniffer platform, would detect the message in the air, but it could not ensure if AP received it. The hybrid monitoring allows pinpointing the source of the problem with more accuracy.

HMP may be also useful to detect some programming errors. Figure 13 shows an example where a possible problem in the WSN nodes is detected. The application stablishes that each node must transmit the measured temperature each 10 seconds. However, the operation cycles of both the repeater node and the sensor node were slightly different from the 10 seconds. In Figure 13, *Transmission* events in the sensor node ED (node C207) and in the repeater node ED/RE (node C206) have been highlighted. It can be noticed that the cycle in the ED/RE node (pink lines) was shorter than the cycle in the ED (yellow lines). This phenomenon could produce unwanted effects depending on the application requirements.

Many causes could be responsible of this behavior. May be that difference in the clock speed of the nodes has not been taken into account [40], or simply denotes nonexistence –or malfunction– of a synchronization mechanism in the WSN. Other causes could be programming errors or hardware defects.

These delays could have consequences. For example, they could disserve the use of low consumption modes in the WSN between sampling periods. The detection of these problems justifies taking measures to ensure the synchronization among the WSN. The HMP platform, therefore, may be useful to detect also this kind of anomalies.

Finally, the HMP is able to record both internal events and communication events that happen in the nodes of a WSN. Thus, the traces generated by the platform also may be used to reproduce the operation of the WSN in a simulator. Besides, others mechanisms for diagnosis, as for example the proposed in [57] (a belief rule base model for fault diagnosis of WSN) or [58] (a directional diagnosis approach for determining WSN faults), could be used combined with HMP, based on the obtained traces.

These are only a few examples of the usefulness of the HMP to monitor and evaluate the operation of a WSN.

B. INTRUSION EVALUATION OF THE MONITOR NODE

The intrusion –also called interference or overload– is the variation caused by the monitoring functions on the

Timestamp	HMP node	Event	Data length	Frame dest.	Frame source	Frame Code	Message
17:14:53,685924	C207	WakeUp	0x0000				
17:14:53,688590	C207	Transmission	0x0D	0x10000000	0x10000007	0x3F03BA	0x0947
17:14:53,688889	S101	CAPTURED_MESSAG	0x0D	0x10000000	0x10000007	0x3F03BA	0x0947
17:14:53,688899	C206	Reception	0x0D	0x10000000	0x10000007	0x3F03BA	0x0947
17:14:53,692593	**C206**	Transmission	0x0D	0x10000000	0x10000007	**0x3F12BA**	**0x0947**
17:14:53,692743	**S101**	CAPTURED_MESSAG	0x0D	0x10000000	0x10000007	**0x3F12BA**	**0x0947**
17:14:55,332607	C206	Transmission	0x0D	0x10000000	0x10000006	0x3F13C2	0x0A4D
17:14:55,332743	S101	CAPTURED_MESSAG	0x0D	0x10000000	0x10000006	0x3F13C2	0x0A4D
17:14:55,332770	C201	Reception	0x0D	0x10000000	0x10000006	0x3F13C2	0x0A4D
17:15:01,785189	C207	WakeUp	0x0000				

FIGURE 12. A sample of message lost in the AP node.

Timestamp	HMP node	Event	Data length	Frame dest.	Frame source	Frame Code	Message
16:57:19,900170	C207	WakeUp	0x3000343300				
16:57:19,903450	**C207**	Transmission	0x0D	0x10000000	0x10000007	**0x3F0352**	0x0947
16:57:19,903817	S101	CAPTURED_MESSAGE	0x0D	0x10000000	0x10000007	0x3F0352	0x0947
16:57:19,903857	C206	Reception	0x0D	0x10000000	0x10000007	0x3F0352	0x0947
16:57:19,908496	C206	Transmission	0x0D	0x10000000	0x10000007	0x3F1252	0x0947
16:57:19,908656	S101	CAPTURED_MESSAGE	0x0D	0x10000000	0x10000007	0x3F1252	0x0947
16:57:19,908720	C201	Reception	0x0D	0x10000000	0x10000007	0x3F1252	0x0947
16:57:19,949334	**C206**	Transmission	0x0D	0x10000000	0x10000006	**0x3F1359**	0x0A41
16:57:19,949493	S101	CAPTURED_MESSAGE	0x0D	0x10000000	0x10000006	0x3F1359	0x0A41
16:57:19,949517	C201	Reception	0x0D	0x10000000	0x10000006	0x3F1359	0x0A41
16:57:29,949140	**C206**	Transmission	0x0D	0x10000000	0x10000006	**0x3F135A**	0x0A41
16:57:29,949340	S101	CAPTURED_MESSAGE	0x0D	0x10000000	0x10000006	0x3F135A	0x0A41
16:57:29,949371	C201	Reception	0x0D	0x10000000	0x10000006	0x3F135A	0x0A41
16:57:29,981198	C207	WakeUp	0x3000000000				
16:57:29,984517	**C207**	Transmission	0x0D	0x10000000	0x10000007	**0x3F0353**	0x094D
16:57:29,984858	S101	CAPTURED_MESSAGE	0x0D	0x10000000	0x10000007	0x3F0353	0x094D
16:57:29,984898	C206	Reception	0x0D	0x10000000	0x10000007	0x3F0353	0x094D

10,081060s

9,9998s

FIGURE 13. A sample of cycle time differences.

performance of the monitored system [9]. In HMP, the software added to sensor nodes may be considered the main cause of intrusion. Three parameters of performance are affected by this code: used memory, duty cycle (working time), and energy consumption. Energy intrusion has proven [44] to be directly related to the working time and the characteristics of the observed node. Therefore, this parameter has not been directly studied in this evaluation.

1) TIME INTRUSION

The addition of program lines in a sensor node also affects its performance, insofar as it implies that the sensor node has to execute new instructions. Usually, the sensor nodes in WSN remain in the low-power state most of the time, and only switch to active mode for short time intervals to perform their functions, and then return to low-power state. Therefore, this active time is increased due the addition of trap routines.

To evaluate the intrusion in working time, this increment of execution time has been measured experimentally. The application running on an ED was modified to include the activation and deactivation of a digital output line at the beginning and the end of the execution of the trap function. The pulse width of this signal was measured with a digital Keysight oscilloscope, model DSO6014A. According to its specifications, their time measurements must assume an error of $\pm$ 0.005% in the reading plus $\pm$ 0.1% of the width of the oscilloscope screen [53].

It was found in previous experiments [44] that the trap execution time depends on the amount of data that the MN must register for this event. The designer of the monitoring campaign decides the number of bytes to be registered in each event in order to study the system behavior. This size may vary from just one byte to identify an event without additional information, to many bytes to record the values of some variables for later analysis, and even the complete content of a WSN message.

Keeping this in mind, three cases were considered: a simple trap that only sends one byte (1-byte code to identify the captured event), an extended trap with 2 bytes (event code + additional byte) and finally a trap with 4 bytes of parameters (trap code and three additional bytes). It was found that, as expected, the results depend on the implementation of the Mon-Inf interface, as time intrusion is related with the transmission time of data from sensor node to the NM.

TABLE 1. Intrusion in time in the monitored node (in microseconds).

Interface	1 byte	2 bytes	4 bytes
Parallel	3.7	3.8	7.30
SPI/UART	3.2	3.3	3.4

Three different available interfaces of the microcontrollers (GPIO, SPI and UART) have been considered.

Table 1 shows the results obtained. As shown, parallel transmission using GPIO obtains the worst results. This is caused by the lack of specific hardware for flow control over parallel transmission through digital I/O ports. Therefore, the implementation of an acknowledgment protocol (*hand-shake*) in the monitoring library was required to avoid control flow errors. On the other hand, when using standard serial interfaces –such as SPI or UART– the transmission time does not increase significantly with the number of bytes transmitted, probably due to the availability of dedicated devices for data transmission that use direct memory access (DMA) in parallel with the main processor of the sensor node. These devices also take care of flow control, so no additional management is required.

The impact of monitoring operation in the working times of the observed sensor node has been estimated, increasing its awaking time.

Based on the results of Table 1, and considering eight registered events with 8 bytes of data each one (the trap code, a data length byte and six of additional data), the intrusion was calculated. Sensor node working times from 10 to 70 milliseconds were considered, according with many common WSN applications. For instance, [54] presents a wireless node to measure performance on a track bike, with a node working time of 30ms. Other cases such as that presented in [55] –monitoring charge of environmental cooling and heating– or in [56] –a wireless sensor node activated by wireless power–, the time in which the node is active can reach 100ms.

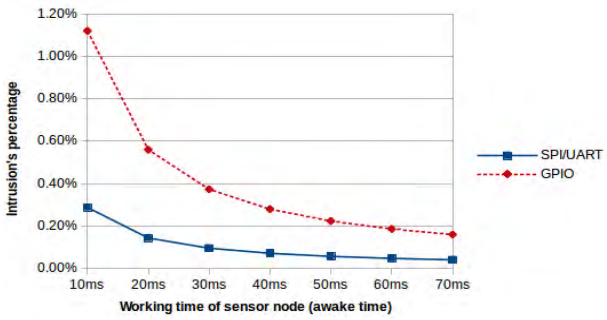


FIGURE 14. Intrusion estimation in working time.

Figure 14 shows the results of the time intrusion. For the worst case –10ms of working time, using parallel communication with GPIO– the intrusion barely exceeds 1%. When the working time is greater than 30ms, it does not reach 0.4%. On the other hand, using a serial interface and DMA it does not exceed 0.3% in the analyzed cases.

2) CODE INTRUSION

The addition of program lines in a sensor node usually increases the required memory usage. The code intrusion has been evaluated by comparing the size in bytes of the compiled code before and after the addition of the monitoring functions. As expected, the increase in size depends on the number of recorded events, that is, the number of traps added into the original code.

Table 2 shows the values of the intrusion in the nodes with the SimpliciTI protocol registering four (ED), three (RE) and two (AP) events. The evaluation was made using the SPI lines as Mon-Inf interface. An deeper study about the influence of Mon-Inf interface implementation in the intrusion can be found in [44]. That study was performed with a previous version of the MN.

TABLE 2. Increment of code and memory using caused by MN (in bytes).

Interface	Original code size		Increment	
	Code	xdata	Code	xdata
Sensor	14697	1423	948 (6.33%)	66 (4.64%)
Repeater	14996	1421	931 (6.21%)	66 (4.64%)
Access point	16602	1644	629 (3.79%)	66 (4.01%)

The code memory required for monitoring purposes is less than 1KB. Notice that the code increment in the Access point is smaller than in the other two. It was found that original Access Point implementation already included some SimpliciTI routines that were not used in Sensor and Repeater implementation, but are required for monitoring interface. On the other hand, the increase in data memory using –shown in the *xdata* column– is constant for all cases (memory space for monitor routines variables), and it does not reach 5% in relation to the space required by the original application of the node.

V. COMPARISON WITH OTHER PROPOSALS

There are a very few proposals found in the bibliography that may be compared with HMP. Table 3 shows a comparison of the characteristics of several platforms, mentioned above.

In some cases, there is not information available about the intrusion caused by the monitoring platform over the observed WSN. In others, it is not clear if the tool can be applied to other nodes different from the used by their authors. For instance, many implementations are based on TinyOS options, being very difficult its portability to other environments. Anyway, for the platforms which provide this information, it can be seen that the declared intrusion is not negligible.

Table 4 shows a comparison of features and capabilities between active monitors, passive monitors, and HMP. The very low intrusion caused by HMP is one of its best advantages when compared with other monitoring platforms.

TABLE 3. Comparison of WSN monitoring platforms.

Platform or Tool	Code (ROM) in bytes	Memory (RAM) in bytes	Time intrusion or overhead	WSN traffic overhead (%)	WSN node used in tests	Used interface	Available for other nodes/interfaces
SNMS (Nucleos) [13]	n/a	1281 (SNMS and 4 queries)	n/a	n/a	Mica2	Messages through WSN	No
Sympathy [31]	1558 (1900 with ping-about)	47 (97 with ping-about)	n/a	<=31%	Simulation with TinyOS	Messages through WSN	Yes?
EnviroLog [17], [59]	15160	809	0.4ms	0%	Mica2	*not apply*	No
Memento[15]	n/a	<400	n/a	n/a	Mica2	Messages through WSN	Yes?
NodeMD [18][59]	3556	302	43-79 CPU's cycles per record	0%	Mantis	*not apply*	No
DSN [21]	n/a	n/a	n/a	0%	BTnode, Moteiv, Tmote	UART	Yes?
Lightweight Tracing [16]	4000-5000	315	n/a	0%	Mica2	*not apply*	No
PAD [32]	n/a	n/a	n/a	<8%	TelosB	Messages through WSN	Yes?
TARDIS [20]	10K-12K (23%-25%)	2.6K	0.01-0.2 duty cycles (1%-20%)	0%	TelosB	*not apply*	No
HDF [11]	n/a	n/a	n/a	0%	n/a	UART, JTAG	Yes?
HMP	**639-950 (4%-8%)**	**66**	**< 1.2% of duty cycle (<8µs per record)**	**0%**	**CC1110 + SimpliciTI nodes**	**Parallel GPIO, UART, SPI**	**Yes**

(n/a) not available

TABLE 4. Capabilities of active platforms, passive platforms, and HMP.

Feature / Capability	Active monitors	Passive Monitors	HMP
Internals' node events recording	Yes	No	Yes
WSN messages sniffing (in air)	No	Yes	Yes
Synchronization of traces	Sometimes	Usually	Yes
Customizable trace format	Usually	Sometimes	Yes
Availability of interfaces to the node	Very few	*not required*	Many options
Caused intrusion or overhead	Low to Medium	None	Very Low
Reusability of components	Rarely (most of them only for the same kind of WSN)	Usually	Yes
Applicable to different types of nodes / WSN	Rarely	sometimes	Yes

The solution applied in HMP for active monitoring is also a hybrid approach.

On one hand, active software monitoring approaches usually require the observed node to process and store the events, thus hugely increasing the memory intrusion. In HMP this effect is avoided by the temporal addition of an external node (MN) which is dedicated to these functions.

A very few tools study its time interference, that's it, they are not aware of the influence of measurement process in the performance of the observed WSN. However, for those that analyze it, the intrusion tends to be higher than the exposed in Table 2 and Figure 14 for HMP. Only NodeMD is comparable in time intrusion with HMP. By reducing the intrusion in code, memory and time, the lower disturbance caused by monitoring in the WSN provides more accurate monitoring results.

It also must be noticed that some of these tools use the WSN to report the captured information. For instance, Sympathy may increase WSN traffic in 31% and PAD in 8%. Others, such as SNMS and Memento, also use WSN messages for their purposes. HMP does not modify nor uses the WSN frames, causing no intrusion for this motive.

Attending its characteristics, there are a very few proposals found in the bibliography that may be compared with HMP. The closer ones, as hybrid platforms, could be the aforementioned Spi-snooper [36] and HFD [11].

Spi-Snooper passively monitors the communication between the microcontroller and the external radio processor, which are connected by a SPI bus. This is suitable for a very few architectures, being unusable when other interfaces are used, or both elements are in the same chip. Moreover, only communication-related events may be observed, keeping the internal state of the processor unobservable.

HDF is also a hybrid monitoring environment, where the monitor nodes may query the observed node about its state. Passive observation is also possible. The authors have not developed the whole platform, and only a single component – the active one – has been designed as a prototype. This design is highly hardware-dependent and the overload of this monitoring has not been evaluated, but it may be supposed that the intrusion caused by queries is not negligible.

VI. CONCLUSION

In this paper, a Hybrid Monitoring Platform (HMP) for WSN has been presented. HMP is designed to be used to aid in development, debugging and deployment of WSN in both laboratory and on-the-field WSN applications.

HMP is based on an open architecture, which provides both interoperability and flexibility in the development of its components. Moreover, the use of a standard format to store the collected information allows other applications to use this information.

The hybrid approach combines both passive and active operation to provide greater observability of the monitored WSN with low intrusion. The active part, also, uses a hybrid hardware/software approach to reduce intrusion. Software traps offer high flexibility for the designer to define the most suitable monitoring campaign, and the MN provides additional hardware to process the events without increasing the workload of the observed sensor node.

From the point of view of distributed monitoring, two of the HMP proposed elements are remarkably new. The first is the Monitoring Node (MN), a hybrid hardware-software component. The MN implements the three layers of the architecture: Monitoring layer covers the reception of events from the sensor node; Information layer provides a mechanism of format assignment and time stamp; Interchange layer performs the data storage. This frees the monitored node from these tasks and therefore achieves a very low intrusion to the observed sensor node. On the other hand, the mechanism of trace synchronization is also a novelty. It provides a simple but effective procedure, susceptible to be applied in many other monitoring systems that require offline trace synchronization.

When comparing with other studied proposals, the intrusion caused by the MN is lower than other active approaches, that usually introduce intrusion, but provides very valuable information regarding the operation of the node. In addition, the MN has proven to be suitable for most of the sensor node implementations, as it can use many standard interfaces. The MNs may be reused in many new monitoring campaigns.

HMP gathers information from both internal events of the sensor nodes and their wireless transmissions. Combining both sources of information allows a more complete analysis of the behavior of the monitored WSN. In addition, the final trace reproduces the true operation of the WSN, so it may be used as synthetic workload for many simulation tools. This opens many new possibilities for these tools, such as the evaluation of the performance of a new WSN implementation.

The applicability of HMP on real WSN nodes has been evaluated, being a success. The modular architecture allows a highly configurable monitoring, and the functions implemented in the MS provide an agile processing of the obtained data. The benefits achieved by the platform have been demonstrated by means of the detection of two anomalies in the operation of the nodes.

Scalability is one of the best characteristics of HMP. Increasing the number of observed nodes in a WSN would not harm the performance of the monitoring system, as each MN is independent of the rest. Even more, real-time reports – which are not required for WSN monitoring – may be achieved if the monitoring communication network is properly sized.

As a future work, the HMP capabilities will be improved by the evolution of MN and SN. A wireless GTSO synchronization mechanism based on a secondary radio channel for synchronization signals is being verified. In this new approach, the Interchange layer is implemented by a SD card, and all the HMP elements are provided with an autonomous power source, avoiding any wire requirement. In this new implementation, the applicability of HMP in deployed networks will be greatly improved.

Finally, a parser is being developed for translating any trace log to a synthetic workload for several network simulators.

REFERENCES

[1] P. Tuwanut and S. Kraijak, "A survey on IoT architectures, protocols, applications, security, privacy, real-world implementation and future trends," in *Proc. 11th Int. Conf. Wireless Commun., Netw. Mobile Comput.*, Sep. 2015, pp. 1–6.

[2] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2347–2376, 4th Quart., 2015.

[3] A. Tripathi, H. P. Gupta, T. Dutta, R. Mishra, K. K. Shukla, and S. Jit, "Coverage and connectivity in WSNs: A survey, research issues and challenges," *IEEE Access*, vol. 6, pp. 26971–26992, 2018.

[4] A. Mahapatro and P. M. Khilar, "Fault diagnosis in wireless sensor networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 2000–2026, 4th Quart., 2013.

[5] D. Raposo, A. Rodrigues, J. S. Silva, and F. Boavida, "A taxonomy of faults for wireless sensor networks," *J. Netw. Syst. Manage.*, vol. 25, no. 3, pp. 591–611, Jul. 2017.

[6] M. M. Patel and A. Aggarwal, "Security attacks in wireless sensor networks: A survey," in *Proc. Int. Conf. Intell. Syst. Signal Process. (ISSP)*, Mar. 2013, pp. 329–333.

[7] A. K. Dwivedi and O. P. Vyas, "An exploratory study of experimental tools for wireless sensor networks," *Wireless Sensor Netw.*, vol. 3, no. 7, pp. 215–240, 2011.

[8] P. Eugster, V. Sundaram, and X. Zhang, "Debugging the Internet of Things: The case of wireless sensor networks," *IEEE Softw.*, vol. 32, no. 1, pp. 38–49, Jan./Feb. 2015.

[9] D. Ferrari, G. Serazzi, and A. Zeigner, *Measurement and Tuning of Computer Systems*. Upper Saddle River, NJ, USA: Prentice-Hall, 1983.

[10] M. U. Khan and M. Zulkernine, "A hybrid monitoring of software design-level security specifications," in *Proc. 14th Int. Conf. Qual. Softw.*, Oct. 2014, pp. 111–116.

[11] Y.-J. Kim, S. Song, and D. Kim, "HDF: Hybrid debugging framework for distributed network environments," *ETRI J.*, vol. 39, no. 2, pp. 222–233, Apr. 2017.

[12] A. Schoofs and G. M. P. O'Hare, and A. G. Ruzzelli, "Debugging low-power and lossy wireless networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 2, pp. 311–321, 2nd Quart., 2012.

[13] G. Tolle and D. Culler, "Design of an application-cooperative management system for wireless sensor networks," in *Proc. 2nd Eur. Workshop Wireless Sensor Netw.*, Feb. 2005, pp. 121–132.

[14] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, "TinyOS: An operating system for sensor networks," in *Ambient Intelligence*. Berlin, Germany: Springer-Verlag, 2005, pp. 115–148.

[15] S. Rost and H. Balakrishnan, "Memento: A health monitoring system for wireless sensor networks," in *Proc. 3rd Annu. IEEE Commun. Soc. Sensor Ad Hoc Commun. Netw.*, vol. 2, Sep. 2006, pp. 575–584.

[16] V. Sundaram, P. Eugster, and X. Zhang, "Lightweight tracing for wireless sensor networks debugging," in *Proc. 4th Int. Workshop Middleware Tools, Services Run-Time Support Sensor Netw.*, Dec. 2009, pp. 13–18.

[17] L. Luo, T. He, G. Zhou, L. Gu, T. F. Abdelzaher, and J. A. Stankovic, "Achieving repeatability of asynchronous events in wireless sensor networks with EnviroLog," in *Proc. 25TH IEEE Int. Conf. Comput. Commun.*, Apr. 2006, pp. 1–14.
[18] V. Krunic, E. Trumpler, and R. Han, "NodeMD: Diagnosing node-level faults in remote wireless sensor systems," in *Proc. 5th Int. Conf. Mobile Syst., Appl. Services*, Jan. 2007, pp. 43–56.
[19] K. Romer and J. Ma, "PDA: Passive distributed assertions for sensor networks," in *Proc. Int. Conf. Inf. Process. Sensor Netw.*, Apr. 2009, pp. 337–348.
[20] M. Tancreti, V. Sundaram, S. Bagchi, and P. Eugster, "TARDIS: Software-only system-level record and replay in wireless sensor networks," in *Proc. 14th Int. Conf. Inf. Process. Sensor Netw.*, Apr. 2015, pp. 286–297.
[21] M. Dyer, J. Beutel, T. Kalt, P. Oehen, L. Thiele, K. Martin, and P. Blum, "Deployment support network," in *Wireless Sensor Networks*, K. Langendoen and T. Voigt, Eds. Berlin, Germany: Springer, 2007, pp. 195–211.
[22] G. Wagenknecht, M. Anwander, T. Braun, T. Staub, J. Matheka, and S. Morgenthaler, "MARWIS: A management architecture for heterogeneous wireless sensor networks," in *Proc. Int. Conf. Wired/Wireless Internet Commun.*, 2008, pp. 177–188.
[23] M. Ringwald, K. Römer, and A. Vitaletti, "SNIF: Sensor Network Inspection Framework," Dept. Comput. Sci., ETH, Zürich, Switzerland, Tech. Rep., 2006.
[24] M. Ringwald and K. Römer, "Snif: A comprehensive tool for passive inspection of sensor networks," in *Proc. Fachgespräch Sensornetzwerke*, Jul. 2007, p. 59.
[25] A. Awad, R. Nebel, R. German, and F. Dressler, "On the need for passive monitoring in sensor networks," in *Proc. 11th EUROMICRO Conf. Digit. Syst. Des. Archit., Methods Tools*, Sep. 2008, pp. 693–699.
[26] B.-R. Chen, G. Peterson, G. Mainland, and M. Welsh, "LiveNet: Using passive monitoring to reconstruct sensor network dynamics," in *Proc. Int. Conf. Distrib. Comput. Sensor Syst.*, 2008, pp. 79–98.
[27] X. Kuang and J. Shen, "SNDS: A distributed monitoring and protocol analysis system for wireless sensor network," in *Proc. 2nd Int. Conf. Netw. Secur., Wireless Commun. Trusted Comput.*, vol. 2, Apr. 2010, pp. 422–425.
[28] Z. Zhao, W. Huangfu, and L. Sun, "NSSN: A network monitoring and packet sniffing tool for wireless sensor networks," in *Proc. 8th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Aug. 2012, pp. 537–542.
[29] F. P. Garcia, R. M. C. Andrade, C. T. Oliveira, and J. N. De Souza, "EPMOSt: An energy-efficient passive monitoring system for wireless sensor networks," *Sensors*, vol. 14, no. 6, pp. 10804–10828, Jun. 2014.
[30] S. Tennina, O. Gaddour, A. Koubâa, F. Royo, M. Alves, and M. Abid, "Z-Monitor: A protocol analyzer for IEEE 802.15.4-based low-power wireless networks," *Comput. Netw.*, vol. 95, pp. 77–96, Feb. 2016.
[31] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin, "Sympathy for the sensor network debugger," in *Proc. 3rd Int. Conf. Embedded Netw. Sensor Syst.*, Nov. 2005, pp. 255–267.
[32] Y. Liu, K. Liu, and M. Li, "Passive diagnosis for wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 18, no. 4, pp. 1132–1144, Aug. 2010.
[33] P. Sommer and B. Kusy, "Minerva: Distributed tracing and debugging in wireless sensor networks," in *Proc. 11th ACM Conf. Embedded Netw. Sensor Syst.*, Nov. 2013, p. 12.
[34] R. Lim, F. Ferrari, M. Zimmerling, C. Walser, P. Sommer, and J. Beutel, "FlockLab: A testbed for distributed, synchronized tracing and profiling of wireless embedded systems," in *Proc. ACM/IEEE Int. Conf. Inf. Process. Sensor Netw. (IPSN)*, Apr. 2013, pp. 153–165.
[35] A. Pötsch, A. Berger, G. Möstl, and A. Springer, "TWECIS: A testbed for wireless energy constrained industrial sensor actuator networks," in *Proc. IEEE Emerg. Technol. Factory Automat. (ETFA)*, Sep. 2014, pp. 1–4.
[36] M. S. Hossain, W. S. Lee, and V. Raghunathan, "SPI-SNOOPER: A hardware-software approach for transparent network monitoring in wireless sensor networks," in *Proc. 8th IEEE/ACM/IFIP Int. Conf. Hardw./Softw. Codesign Syst. Synth.*, Oct. 2012, pp. 53–62.
[37] M. Jabbarifar, "On line trace synchronization for large scale distributed systems," Ph.D. dissertation, Département génie informatique génie logiciel, École Polytechnique de Montréal, Montreal, PQ, Canada, 2013.
[38] A. Sampath and C. Tripti, "Synchronization in distributed systems," in *Advances in Computing and Information Technology*, 176th ed., N. Meghanathan, D. Nagamalai, and N. Chaki, Eds. Berlin, Germany: Springer, 2012, pp. 417–424.
[39] D. L. Mills, "Internet time synchronization: The network time protocol," *IEEE Trans. Commun.*, vol. 39, no. 10, pp. 1482–1493, Oct. 1991.
[40] M. Navia, J. C. Campelo, A. Bonastre, R. Ors, and J. V. Capella, "Drift clock analysis on distributed embedded systems for IoT applications," in *Proc. Workshop Innov. Inf. Commun. Technol.*, Jun. 2016, pp. 42–49.
[41] J. Edison, "IEEE-1588 standard for a precision clock synchronization protocol for networked measurement and control systems—A tutorial," Agilent Technol., White Paper, Oct. 2005, pp. 1–94.
[42] A. Depari, P. Ferrari, A. Flammini, D. Marioli, and A. Taroni, "A new instrument for real-time Ethernet performance measurement," *IEEE Trans. Instrum. Meas.*, vol. 57, no. 1, pp. 121–127, Jan. 2008.
[43] J. V. Capella, J. C. Campelo, A. Bonastre, and R. Ors, "A reference model for monitoring IoT WSN-based applications," *Sensors*, vol. 16, no. 11, p. 1816, Oct. 2016.
[44] M. Navia, J. C. Campelo, A. Bonastre, R. Ors, J. V. Capella, and J. J. Serrano, "Active low intrusion hybrid monitor for wireless sensor networks," *Sensors*, vol. 15, no. 9, pp. 23927–23952, Sep. 2015.
[45] M. Navia, J. C. Campelo, A. Bonastre, and R. Ors, "GTSO: Global trace synchronization and ordering mechanism for wireless sensor network monitoring platforms," *Sensors*, vol. 18, no. 1, p. 28, Dec. 2018.
[46] STMicroelectronics. *STM32F4DISCOVERY Discovery kit with STM32F407VG MCU*. Accessed: May 16, 2016. [Online]. Available: http://www.st.com/content/st_com/en/products/evaluation-tools/product-evaluation-tools/mcu-eval-tools/stm32-mcu-eval-tools/stm32-mcu-discovery-kits/stm32f4discovery.html
[47] Embest Technology Company. *STM32F4DIS-BB User Manual*. Accessed: Aug. 5, 2017. [Online]. Available: http://www.element14.com/
[48] Wireshark Foundation. (2012). *Wireshark—About*. Accessed: Jun. 1, 2018. [Online]. Available: https://www.wireshark.org/about.html
[49] T. Buchheim. (2002). *Nam: Network Animator*. Accessed: Jun. 1, 2018. [Online]. Available: https://www.isi.edu/nsnam/nam/
[50] Nsnam. (2017). *NetAnim*. Accessed: Jun. 1, 2018. [Online]. Available: https://www.nsnam.org/wiki/NetAnim
[51] Texas Instruments. *CC1110-CC1111 Sub-1 GHz Wireless MCU with up to 32 kB Flash Memory*. Accessed: Aug. 4, 2017. [Online]. Available: http://www.ti.com/product/CC1110-CC1111
[52] Texas Instruments. *SIMPLICITI SimpliciTI Compliant Protocol Stack*. Accessed: Aug. 4, 2017. [Online]. Available: http://www.ti.com/tool/SimpliciTI
[53] *6000 Series Oscilloscopes Data Sheet*, K-Technologies, Santa Rosa, CA, USA, 2017.
[54] S. Gharghan, R. Nordin, and M. Ismail, "Energy-efficient ZigBee-based wireless sensor network for track bicycle performance monitoring," *Sensors*, vol. 14, no. 8, pp. 15573–15592, Aug. 2014.
[55] A. Molina-Garcia, J. A. Fuentes, E. Gomez-Lazaro, A. Bonastre, J. C. Campelo, and J. J. Serrano, "Development and assessment of a wireless sensor and actuator network for heating and cooling loads," *IEEE Trans. Smart Grid*, vol. 3, no. 3, pp. 1192–1202, Sep. 2012.
[56] D.-S. Lee, Y.-H. Liu, and C.-R. Lin, "A wireless sensor enabled by wireless power," *Sensors*, vol. 12, no. 12, pp. 16116–16143, Dec. 2012.
[57] W. Gong, K. Liu, and Y. Liu, "Directional diagnosis for wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 5, pp. 1290–1300, May 2015.
[58] W. He, P.-L. Qiao, Z.-J. Zhou, G.-Y. Hu, Z.-C. Feng, and H. Wei, "A new belief-rule-based method for fault diagnosis of wireless sensor network," *IEEE Access*, vol. 6, pp. 9404–9419, 2018.
[59] Q. Cao, T. Abdelzaher, J. Stankovic, K. Whitehouse, and L. Luo, "Declarative tracepoints: A programmable and application independent debugging system for wireless sensor networks," in *Proc. 6th ACM Conf. Embedded Netw. Sensor Syst.*, Nov. 2008, vol. 71, no. 3, pp. 85–98.

MARLON NAVIA MENDOZA (M'15) received the B.Eng. degree in informatics systems from the Universidad Técnica de Manabí, Portoviejo, and the M.S. degree in computer engineering and the Ph.D. degree in informatics from the Universitat Politècnica de València, Spain. He has been a Professor with the Escuela Superior Politécnica Agropecuaria de Manabí (ESPAM MFL), since 2008. He currently teaches teleinformatics and operating systems programming. He also manages a research group of computer systems. His research interests include wireless sensor networks, computer networks, and the Internet of Things.

JOSÉ CARLOS CAMPELO RIVADULLA received the M.Sc. degree in computer engineering and the Ph.D. degree from the Universitat Politècnica de València, Spain, in 1993 and 1999, respectively, where he is currently an Associate Professor with the Department of Computer Engineering. He is also a member of the Smart Electronic and Networks Research Group (SENS), ITACA Institute. He is also teaching computer networks and industrial local area networks. His research interests include wireless sensor networks, the Internet of Things, embedded systems design, and fault tolerant systems' design.

ALBERTO MIGUEL BONASTRE PINA received the M.Sc. degree in computer engineering and the Ph.D. degree from the Universitat Politècnica de València (UPV), Spain, in 1991 and 2001, respectively, where he is currently an Associate Professor. He is also a member of the Smart Electronic and Networks Research Group (SENS), ITACA Institute. He is also teaching computer networks and wireless sensor networks. His research interests include the Internet of Things and wireless sensor networks, being especially devoted to fault tolerance in chemical analysis systems.

JUAN VICENTE CAPELLA HERNÁNDEZ received the Engineering degree in computer science and the M.S. degree in computer engineering from the Universitat Politècnica de València (UPV), in 1998 and 2002, respectively, where he is currently pursuing the Ph.D. degree. He was the Head of the HP Institute, UPV, from 2013 to 2015. He is currently an Associate Professor with the Department of Computer Engineering (DISCA), UPV, and a Deputy Director of Research. He is also a member of the Smart Electronic and Networks Research Group (SENS), ITACA Institute. His main research interests include distributed systems design, wireless and underwater sensor networks, and the Internet of Things (IoT).

RAFAEL ORS CAROT is currently a Professor with the Department of Computer Engineering (DISCA), Universitat Politècnica de València (UPV). He also teaches computer engineering, computer networks, and biomedical engineering. He is also a Co-Director of the Smart Electronic and Networks Research Group (SENS), ITACA Institute. His research interest includes the development of real-time reliable distributed systems and its applications in the control of several processes. He has authored more than 90 research papers on these subjects, and he has participated in several projects to translate these investigations to the industry.