CISTER
Research Center in
Real-Time & Embedded
Computing Systems

Technical Report

Monitoring large scale IEEE 802.15.4/ZigBee based Wireless Sensor Networks

Stefano Tennina

Olfa Gaddour

Fernando Royo

Anis Koubaa

Mario Alves

CISTER-TR-131101

Version:

Date: 11-01-2013

Monitoring large scale IEEE 802.15.4/ZigBee based Wireless Sensor Networks

Stefano Tennina, Olfa Gaddour, Fernando Royo, Anis Koubaa, Mario Alves

CISTER Research Unit

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8340509

E-mail:

http://www.cister.isep.ipp.pt

Abstract

Monitoring of large scale Wireless Sensor Networks is a fundamental task to track the network behavior and measure its perfor- mance in real-world deployments. ZigBee is a standard protocol that operates on top of IEEE 802.15.4 PHY and MAC layers, and represents a very prominent technology for WSNs as it is gaining more and more interest by the research community. Designing a Monitoring tool for IEEE 802.15.4/Zigbee becomes of a paramount importance and many commercial products have been proposed. However, similar commercially available products for monitoring and testing ZigBee based WSNs suffer from being either a proprietary solution not flexible to adapt to different protocol stacks or simply too expensive, typically require special sniff- ing hardware and not suitable for large scale WSNs. In this paper, we present Z-Monitor, a free and extensible monitoring tool and protocol analyzer solution to control and debug IEEE 802.15.4/ZigBee-compliant WSNs. In addition, we detail our experience in deploying Z-Monitor in several testbeds ranging from small to medium to large scale, where the network is monitored by multiple sniffing points, and the architecture of the tool is able to present a global picture of the network as if only one sniffing point is able to collect the whole packets. Finally, we show how Z-Monitor is able to measure network performance in real world deployments.

Monitoring large scale IEEE 802.15.4/ZigBee based Wireless Sensor Networks

Stefano Tennina[1], Olfa Gaddour[2], Fernando Royo[1,4], Anis Koubâa[3], Mario Alves[1]

[1] CISTER/INESC-TEC Research Unit, ISEP/IPP, Portugal.
[2] CES Laboratory, National School of Engineers of Sfax, Sfax, Tunisia
[3] COINS Research Group, Prince Sultan University, Saudi Arabia.
[4] Universidad de Castilla-La Mancha, Spain.
sotna@isep.ipp.pt, olfa.gaddour@coins-lab.org, froyo@dsi.uclm.es, akoubaa@coins-lab.org, mjf@isep.ipp.pt

Abstract. Monitoring of large scale Wireless Sensor Networks is a fundamental task to track the network behavior and measure its performance in real-world deployments. ZigBee is a standard protocol that operates on top of IEEE 802.15.4 PHY and MAC layers, and represents a very prominent technology for WSNs as it is gaining more and more interest by the research community. Designing a Monitoring tool for IEEE 802.15.4/Zigbee becomes of a paramount importance and many commercial products have been proposed. However, similar commercially-available products for monitoring and testing ZigBee based WSNs suffer from being either a proprietary solution not flexible to adapt to different protocol stacks or simply too expensive, typically require special sniffing hardware and not suitable for large scale WSNs. In this paper, we present Z-Monitor, a free and extensible monitoring tool and protocol analyzer solution to control and debug IEEE 802.15.4/ZigBee-compliant WSNs. In addition, we detail our experience in deploying Z-Monitor in several testbeds ranging from small to medium to large scale, where the network is monitored by multiple sniffing points, and the architecture of the tool is able to present a global picture of the network as if only one sniffing point is able to collect the whole packets. Finally, we show how Z-Monitor is able to measure network performance in real world deployments.

Keywords: Wireless sensor networks, protocol analyzer, distributed sniffing

1 Introduction and Motivation

Low Power Wireless Personal Area Networks (LoWPANs) are IEEE 802.15.4 [1] compliant networks, which are becoming increasingly important because of their essential role in Cyber-Physical Systems. LoWPANs are characterized by low-cost, low data rate, infrastructure-less connectivity and suitability to a wide range of applications, making them an essential component of the ubiquitous

computing paradigm. In order to ensure that the LoWPANs are operating at the desired performance level and quantitatively demonstrate to the end-user that their requirements are met, it is essential to provide network and system administrators with tools for monitoring and protocol analysis. Network Monitoring is, in fact, the key to enable: *(i)* executing performance analysis, *(ii)* tracking the network behavior, *(iii)* support to network programmers in debugging their codes, and *(iv)* performing network management operations. Indeed, there exist solutions for LoWPAN monitoring, but they suffer from several limitations, such as their high cost (hundreds of dollars in several cases), their requirement for additional special hardware or their proprietary nature, which makes them not suitable to user customizations.

LoWPANs are typically composed of devices that conform to the IEEE 802.15.4-2003 standard. While the IEEE 802.15.4 standard specifies the Physical and Medium Access Control (MAC) layers and underlying services for LoWPANs, upper layers like Network and Application layers are defined by other standards, like ZigBee and 6LoWPANs.

Despite the fact that ZigBee and 6LoWPAN/RPL are arguably the most important WSN technologies today, very little is available on network monitoring and debugging of these networks. There exist some commercial products for LoWPAN monitoring, but mostly, these are packet sniffers that require special hardware and are quite expensive. To the best of our knowledge, no free and dedicated solution is available to provide network monitoring and debugging that provides full support of IEEE 802.15.4-based protocol family.

Z-Monitor [2] is the result of our motivation to provide a free application for LoWPAN monitoring and control with the following goals:

- Providing a LoWPAN protocol analysis tool, which can be used without any special sniffer hardware, free to use, modular, and extensible.
- Supporting ZigBee and 6LoWPAN protocols' packets decoding.
- Providing statistical tools for network traffic analysis for enabling network performance evaluation.
- Providing a user-friendly interactive GUI, which does not only display packet information to the end-user, but also can be used to easily control the network and monitor its behavior.
- In new version, providing distributed sniffing point capability, to monitor big networks using several sniffing points.
- Can run on several operating systems i.e., Windows and Linux.

The rest of this paper is as follows. Section 2 overviews the most relevant research efforts and commercial products, and presents the contributions of Z-Monitor as compared to these efforts. In Section 3, we present the software design of Z-Monitor with its standalone and distributed sniffing versions, and describe its main software components. Section 4 demonstrates the effectiveness of Z-Monitor through experiments in several testbeds, including a large scale and dense demonstrator covering three floors of a Scientific Park building. Finally, Section 5 closes the paper with discussions about ongoing work and future extensions.

2 Related Work

Several packet analyzer tools have been proposed to sniff and monitor the IEEE 802.15.4-based networks. In what follows, we present some of the most relevant products that are either free or were proposed for commercial purposes.

Ubiqua protocol analyzer The Ubiqua tool[5] is a real time analyzer that captures and analyses IEEE 802.15.4 over-the-air packets. It has several features including a graphical representation of the network topology with logical network links between nodes, inspecting the network traffic, exploring the available networks and nodes and their attributes. However, it is relatively expensive.

SmartRF Packet Sniffer The CC2420 packet sniffer[6] captures, filters and decodes IEEE 802.15.4 MAC packets, and displays them in a convenient way. Frames may be filtered based on frame type and addressing information. Data may be stored to a binary file format. Data frames, beacon frames, command frames and acknowledgement frames are decoded separately, and each field for each frame is decoded and displayed separately on screen. This tool only considers the Physical, MAC and network layers of the IEEE 802.15.4/ZigBee protocol stack. However, it does not present any analysis of the application layer data and does not provide any means for monitoring and controlling large-scale multi-hop networks.

ZENA The ZENA tool[7] is a free wireless network analyzer that graphically displays wireless network traffic following the IEEE 802.15.4 specification on the 2.4 GHz band. The ZENA analyzer supports the ZigBee, MiWi and MiWi P2P protocols. In conjunction with the hardware packet sniffer, the software can analyze network traffic and graphically display decoded packets. It can also display a graphical representation of the network topology and the messages as they flow through the network. This information can then be saved and/or exported for further analysis. However, it doesn't support the analysis of many LoWPAN protocols such as 6LoWPAN and RPL.

Wireshark Wireshark[8] is a free and open-source packet analyzer tool. It is used largely for network troubleshooting, analysis, software and communications protocol development, and education. It provides a user-friendly interface with storing and filtering features. Wireshark supports capturing packets in both from live network and from a saved capture file. The capture file format is libpcap format like that in "tcpdump". It supports various kinds of operating systems. However, it does not detect automatically the USB interface of some WSN platforms to capture data. In addition, it cannot display the graphical topology representation of the network.

[5] www.ubilogix.com/products/ubiqua
[6] www.ti.com
[7] www.microchip.com
[8] www.wireshark.org

In the following table, we summarize the features of the available tools and we compare them with our tool Z-Monitor. As shown in the figure, Z-Monitor has several advantages. For instance, with the latest version, it is the only tool that supports multiple sniffing points.

Table 1: Comparison of the main features of the available sniffing tools.

	Special hardware needed?	Cost	Supported protocols	Support to distributed sniffing	GUI topology visualization
Ubiqua protocol analyzer	Yes, specifically designed	High cost (999$)	ZigBee, 802.15.4	No	No
SmartRF Packet Sniffer	Yes, specifically designed	Low cost (50$)	ZigBee, 802.15.4	No	No
Zena	Yes, specifically designed	Free	ZigBee, 802.15.4, MiWi	No	Yes
Wireshark	Yes, some devices widespread	Free	ZigBee, 802.15.4, 6LoWPAN, RPL	No	No
Z-Monitor	Yes, several devices widespread	Free	ZigBee, 802.15.4, 6LoWPAN, RPL	Yes	Yes

3 Z-Monitor Design and Features

In this Section, we present the software architecture of Z-Monitor and its main features. We first present the stadalone mode. Then, we present the design of the distributed sniffing version.

3.1 Standalone Mode

The software design objective of Z-Monitor is to provide an open source, extensible, modular and user-friendly solution for LoWPAN monitoring. Z-Monitor allows for monitoring of IEEE 802.15.4-based networks and for analyzing the network behavior through statistical data analysis. Z-Monitor relies on a particular sensor node acting as a passive sniffer that captures network traffic and redirects it to a user-friendly Graphical User Interface (GUI). The fundamental advantage of Z-Monitor as compared to commercially available products reviewed in Section 2 is its independency to any special hardware.

To meet the aforementioned objectives, a component-based approach has been used to design Z-Monitor. The block diagram of the main components is shown in Figure 1.

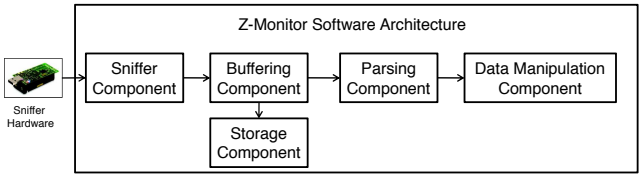

Figure 1: The Block Diagram of Z-Monitor

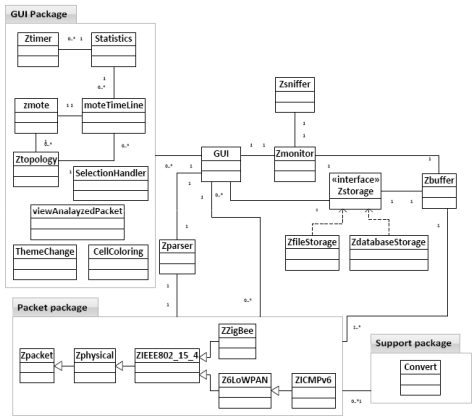

Figure 2: The UML Diagram of Z-Monitor

On the hardware side, the *sniffer hardware* is simply an IEEE 802.15.4-compliant sensor mote, which passively captures the network traffic. Each received packet is redirected to the serial interface through which the sniffer is attached to forward that packet to the software sniffing threads. The sniffer hardware that we have used is a TelosB[9] mote which implements the packetsniffer application available under TinyOS[10]. The packetsniffer application switches the USB port into promiscuous mode and subsequently sniffs all packets that come along. Z-Monitor collects packets arriving from the USB port, stores them in a buffer, performs parsing and packet decoding and finally displays parsed frames and outputs network statistics.

On the software side, five main software components, implemented through various Java classes, have been defined, and the respective simplified UML diagram is presented in Figure 2.

In the following, we present the different software components of Z-Monitor.

– Sniffer Component. It reads data from the source specified by the configuration panel, i.e., the mote, and process it to be interpreted into a logical

[9] www.memsic.com

[10] https://github.com/tinyos/tinyos-main/tree/master/apps/tests/tkn154/packetsniffer

format. The Java class Zsniffer receives the packets from the packetsniffer and redirect them to the buffering component.

- Buffering Component. It stores the incoming bit-stream of data int the volatile memory to avoid packet loss. The Java class Zbuffer handle this operation. For the sake of independence from the network protocols, at this level Zsniffer simply store the raw bit-stream.
- Storage Component. It allows for the permanent storage of the incoming packets for later off-line analysis. The Java class Zstorage is instantiated into two distinct modules ZfileStorage and ZdatabaseStorage which implements both file-storage and database-storage options.
- Parser Component. It is the hearth of Z-Monitor. The Java class Zparser implements the parsing functionalities of the bit-streams, i.e., recognition of the meaning of each field of the packets, and for that it uses several distinct modules, one for each supported protocol.
- Data Manipulation Component. The role of this last component is to display the packets fields and compute useful statistics. Several display options are available, such as a *plain* view, in which the fields of each packet are displayed in a row (as shown in Figure 3) along with a timeline showing the packets sequence, and a *layered* view, similar to WireShark display, in which a packet is displayed layer by layer (i.e., Physical Layer, then MAC Layer, then Network Layer, ...). The statistics that Z-Monitor provides include the total and average number of packets and average packet size.

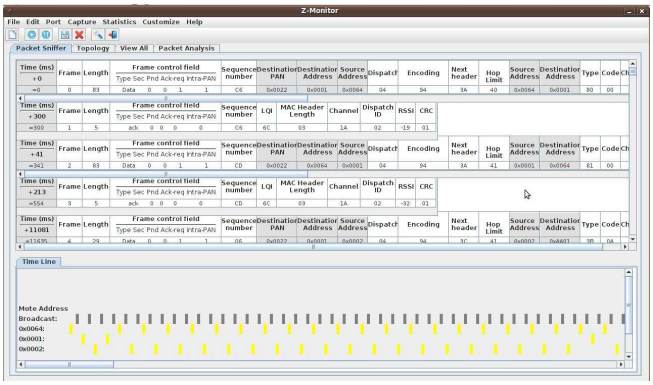

Figure 3: Z-Monitor Frame Decoding

3.2 Distributed Sniffing Mode

In its latest version, Z-Monitor has the feature of distributed sniffing in order to monitor larger networks, i.e., those where only one sniffer is not able to record

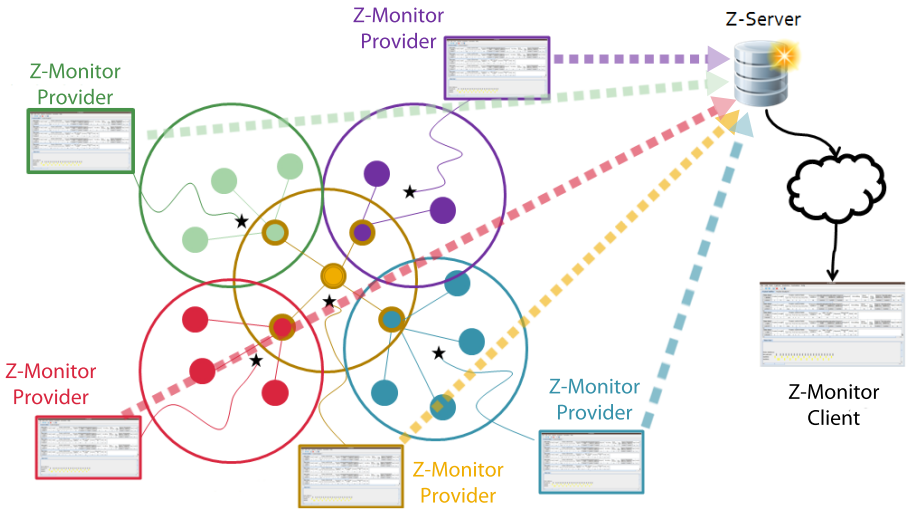

Figure 4: Distributed Z-Monitorarchitecture.

the whole traffic. Basically, by exploiting the possibility of the standalone version to access a remote database, and by using two extra IDs as keys for the database (an ID for the network and one for the user), a client-server architecture has been implemented as shown in Figure 4. More in detail, multiple standalone versions of Z-Monitor (which are named as *provider*, in this case) collect local pictures of the network and send the data through wireless or wired links (e.g., WLAN) to a global Z-Server, which contains the database. Consequently, a remote user running a Z-Monitor Client can monitor the full network in *real-time* by accessing the Z-Server and continuously refreshing the new data from the database. In other words, from a single Z-Monitor station, the user feels the perception of having a sniffer able to monitor the whole network, regardless of its geographical scale.

Nevertheless, since multiple sniffing points often share common areas of the network, they might sniff the same portions of the network traffic and report duplicated packets to the Z-Server. Therefore, in order to avoid this phenomena, an accurate synchronization among the sniffers is needed. To do so, packets are saved in Z-Server with a timestamp, other than the above mentioned network and user IDs. Consequently, although the tool maintains all the functionality described in the standalone mode, some new characteristics have been implemented, in order to support the distributed sniffing. These new elements are:

- The Z-Server, as a local or remote server machine, hosting the database where packets are stored and reachable from the Z-Monitor Providers over a wired or wireless communication channel (e.g., WLAN).
- The Z-Monitor Client, as a local or remote client machine, which present the full network status and statistics to the final user by accessing the Z-Server.

– Synchronization between Z-Monitor Providers for accurately timestamp the packets in order to avoid duplicates and to keep the correct sequence of packets from the different sniffers. We opted for using the Network Time Protocol (NTP) [3] to achieve the required synchronization.
– Distributed duplicated packets avoidance, to ensure that they don't appear in the database. To do so, every Z-Monitor Provider checks in the database if the incoming packet is already present. If not, it will store it in the database, elsewhere it will simply discard it.

To enable the different operation modes of Z-Monitor, we also implemented a boot up choice where the user can select which mode to run, when launching the execution of Z-Monitor. In particular:

– **Standalone version:** The standalone version is the traditional mode, where a node acts as a sniffer showing the local captured and analyzed packets.
– **Provider version:** Same as standalone, but forced to save packets into a local or remote *Z-Server*.
– **Client version:** without any special hardware attached to the Z-Monitor, the user connects to the Z-Server and analyses the traffic of the interested network in real-time or by looking at the past recent history stored in the database.

4 Experimental Study

In this section, by focusing on the ZigBee protocol, we present an experimental study that shows how to perform monitoring and performance evaluation using Z-Monitor. The objectives of the experimental study are manifold:

– To demonstrate the capabilities of Z-Monitor for network monitoring.
– To show how Z-Monitor is useful in evaluating the performance of IEEE 802.15.4-based WSNs.
– To present the collection of network statistics using Z-Monitor.
– To demonstrate the distributed sniffing feature of Z-Monitor, by collecting medium and large scale networks with multiple sniffing points.

4.1 Standalone Mode

The network topology scenario used in this section is as follows, A WSN contains a total of 12 TelosB motes, which consist in one sniffer mote (running the packetsniffer TinyOS application), and 11 network nodes. As network, we considered a Cluster-Tree topology composed of one ZigBee Coordinator, three ZigBee routers and seven End Devices.

TelosB motes were deployed within a single broadcast domain. Therefore, we will present results from a single-hop network testbed. The transmission power of nodes was set to -25 dBm which is the minimal available transmission power,

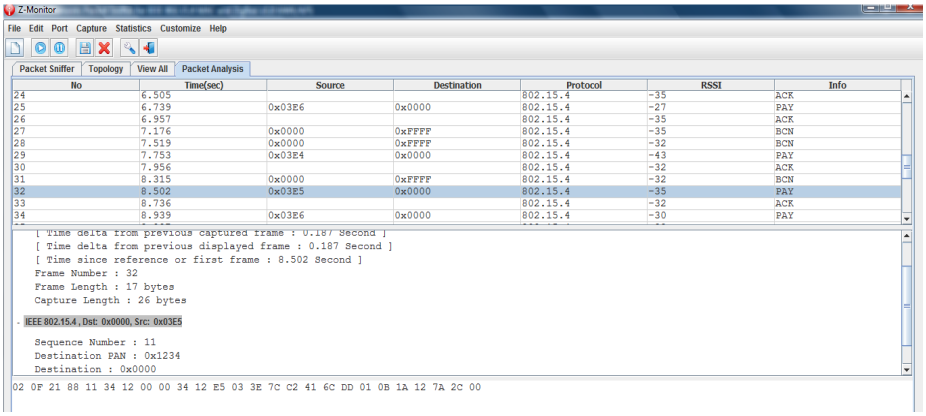

Figure 5: ZigBee Protocol Analysis using Z-Monitor

and the frequency channel was set to 26. We have considered the available open-source implementations of ZigBee protocol namely, the TinyOS Cluster Tree IEEE 802.15.4/ZigBee implementation [4]. The Beacon Order was set to BO=8 leading to a Beacon Interval of BI=3.97 seconds and the Superframe order was set to SO=4. The rest of the network parameters are: (*i*) maximum depth $L_m = 3$ hops; (*ii*) maximum children routers per parent $R_m = 4$ and (*iii*) maximum children nodes per parent $C_m = 6$.

In this study, we measure the *network convergence time* metric of each node, which is the duration a node spends to join the LoWPAN network. To do so, said that all 10 motes (routers and end-devices) wake up at the same time, as soon as a node associates with a parent node, it sends a specific data packet. With Z-Monitor the time when such data packets are sent is recorded and is considered as the joining time of that node.

Figure 5 shows a Z-Monitor screenshot while such a network were running. It is clear from the screenshot that the ZigBee packets are correctly parsed and all the packet types (Beacon, Acknowledgement, Data frames) are supported.

Figure 6 shows the joining time of each node captured by Z-Monitor.

It is clear from the figure that the first ZigBee router joins the network after three beacons (12 seconds) which is expected. In addition, the joining time grows linearly with the ZigBee network size. This is because ZigBee end-devices cannot join the network before the joining of their parents (the routers).

4.2 Distributed Sniffing Mode

In order to validate the new version and measure its performance, we have deployed different WSN testbeds. We have deployed large scale networks with multiple sniffing points placed in different geographical locations (of the same WSN). In what follows, we present the experiment that we conducted in order to test the distributed sniffing version of Z-Monitor.

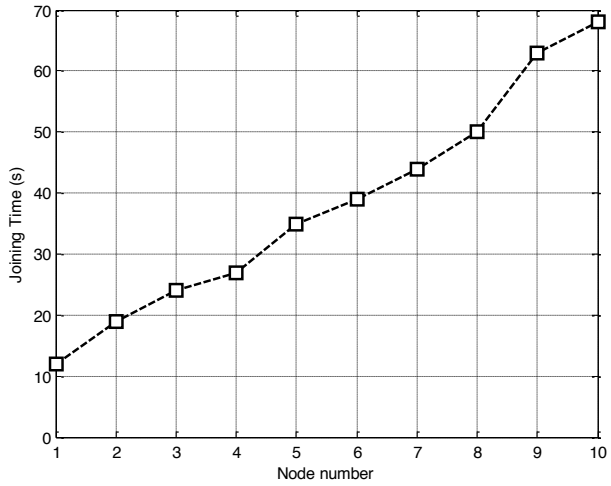

Figure 6: Convergence time of ZigBee WPAN (10 nodes)

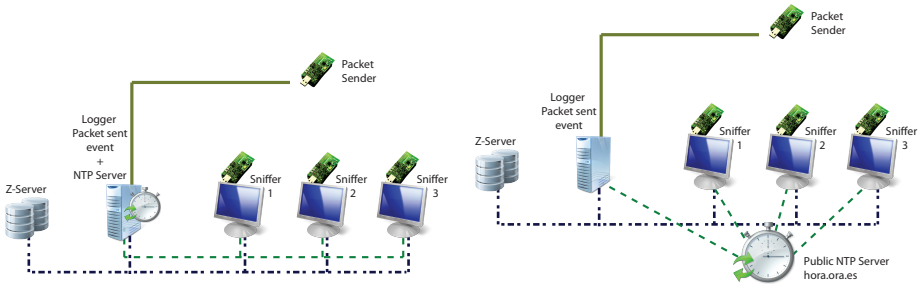

Figure 7: Topology used for the NTP accuracy synchronization experiment. NTP server is located (a) on the same LAN; (b) over Internet.

4.2.1 Evaluation synchronization of multiple sniffing point The synchronization has been obtained thanks to NTP, this decision has been taken according to the results of the following experiments, where the main objective is to shown the accuracy time synchronization, obtained using this protocol when the server is located in local network or when this server is located remotely. For these tests we use three different sniffers, each one of them connected to different PCs (see Figure 7a). These PCs are Z-Monitor Providers, and each one of them is configured to use as time reference a NTP server, connected to the same LAN.

Once the packet sender starts to send beacons packets, these beacons are received by the different sniffers and sent to Z-Server, but also the NTP Server is recording the event *Beacon.sendDone* in order to get a reference time between the sniffers and the packet sender. Each package in the database is accompanied by the sniffer_id and the timestamp set by the sniffer according to the NTP protocol.

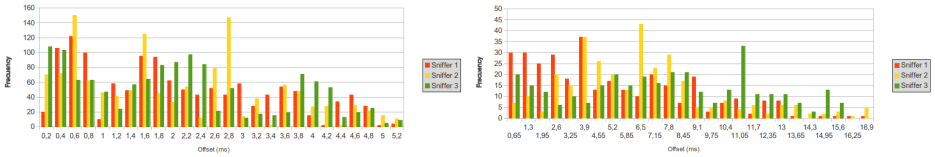

Figure 8: Results, in terms of offset, for the same beacon in the three sniffers, when NTP server is located (a) on the same LAN; (b) over Internet.

We run the test for about 30 minutes, getting a total of 1500 beacons and analyze the offset of the same beacon in each of the three sniffers, comparing the reference time stored by the NTP Server, when the event *Beacon.sendDone* is truly dispatched, and the timestamp stored by each of the sniffers, when they received such a packet. Figure 8a shows the time offset for each sniffer. The range of such offset is around [0.2–5.2] ms.

For the second test, the same assumptions holds, but now a free remote NTP server has been used[11] (see Figure 7b to clarify) Running the test as before we got an offset ranging in [0.6–17] ms (Figure 8b).

Since an offset less than 20 ms is often acceptable and enable to distinguish duplicated packets in low data rate WSNs, these results confirm that our solution is working. Next sections will further demonstrate how the distributed sniffers approach effectively allowed to monitor several networks and derive key performance.

4.2.2 Evaluation over a cluster-tree network The objective of this section is to show the performance of Z-Monitor obtained over a cluster-tree network deployed in the I3ASensorBed [5]. The network topology is shown in Figure 9, where the position of each node and sniffer is evidenced.

The network is configured according to the following parameters: *(i)* maximum number of children for a parent (C_m) set to 9; *(ii)* maximum depth (L_m) set to 3 hops; *(iii)* maximum number of child routers for a parent (R_m) set to 4; *(iv)* beacon order (BO) set to 10; and *(v)* superframe order (SO) set to 5. Note that the beacon order is computed by the Time Division Cluster Scheduling algorithm to avoid intra-clusters collisions [6].

The goal of this section is to show the experiment scenario and the performance obtained by the distributed sniffers. The network works as described next. *(i)* The coordinator starts sending beacons immediately after its switch on. It receives association requests from the nodes and negotiate the beacon start requests from Routers. It is installed on node 30 of Figure 9. *(ii)* Each Router waits until the reception of beacons from its pre-assigned parent to start the association process, then it initiates the negotiation with the Coordinator to get a windows for its superframe. When it succeeds with the negotiation, it starts transmitting beacons and at run time it forwards data from the children to the

[11] *hora.roa.es*, available over Internet

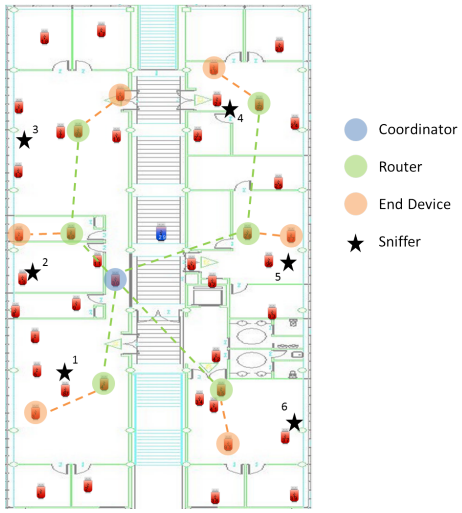

Figure 9: Topology used for the experiment and sniffer location. The six sniffers are marked as stars, the coordinator is the light blue circles, the six routers are green circles and the six end devices are brown circles.

parent. *(iii)* Each End Device waits to receive beacons from its pre-assigned parent and starts the association with it. When it joins the network, it starts sending periodic data packets to the parent to be forwarded to the Coordinator. *(iv)* In six sniffing points the distinct instances of Z-Monitor Provider are configured to feed data packets into the common remote database. The experiment lasts for about 15 minutes.

Analyzing the amount of packets saved in Z-Server, we can conclude some points related to the basic functionality expected from the tool.

- No duplicated packets are present at the end of the experiment in the data base. This confirms that the duplicated packet avoidance technique is working fine.
- There are packets stored from all the sniffers, i.e., there are beacons saved from different sniffers. This denotes that the synchronization NTP-based is accurate enough and the system is fair, i.e., there isn't any sniffer working as a *privileged* one, due to synchronization issues.

Figure 10a shows the total amount of packets stored by each sniffer.

The number of beacons saved for every node acting as Router or Coordinator in the network is shown in Figure 10b. Obviously, it depends on the chronological order of association to the parent node.

Finally, the synchronization and the time accuracy can be evaluated checking the inter-arrival time for the beacons and the offsets of the Routers' beacons with respect to each parent, i.e., the respect to the scheduling fixed by the TDCS algorithm. Figure 11a shows that more than 99% of the beacons are sent within

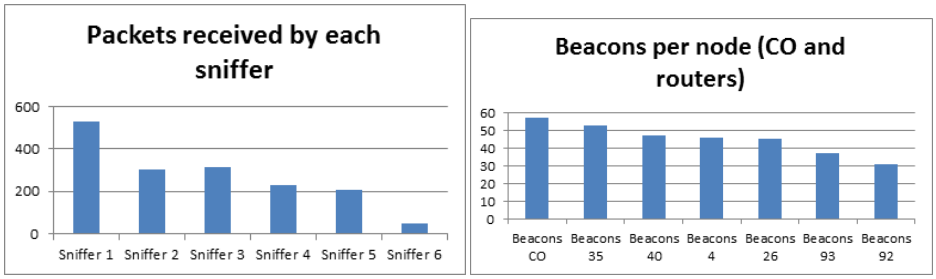

Figure 10: (a) Packets received by each sniffer; (b) Beacons per node.

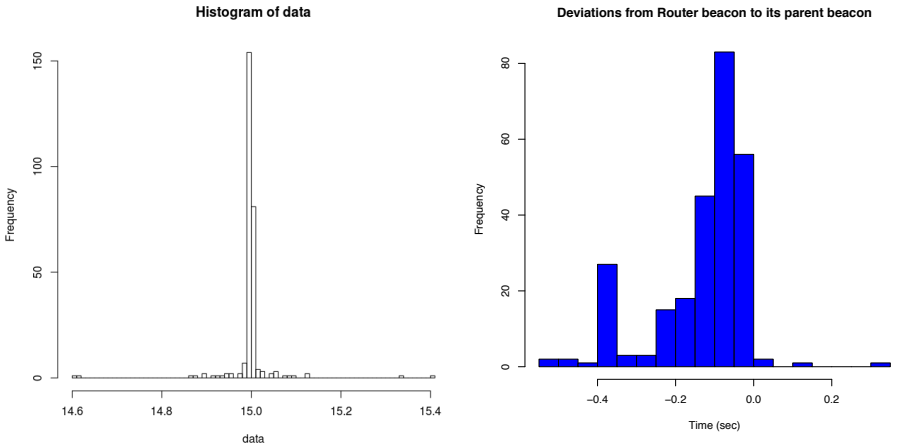

Figure 11: (a) Deviation from the programmed Beacon Interval. $BO = 10$ implies a beacon interval of around 15 seconds; (b) Deviation from the programmed start time of all the routers.

an error of ± 0.1 seconds with respect to the intended time. Figure 11b shows that there is a slight deviation of the start time with respect to the scheduling in the implementation, mainly due to the fact that the resolution of the timers in the TelosB platform is not fully compliant with the IEEE 802.15.4 standard.

Overall, these results validate the implementation of the distributed sniffers, in the sense that the system allows system designers and network administrators to have a global picture of the network behaviour, without being a single sniffer able to capture all the packets. Since all packets are correctly stored in the data base, i.e., in the right sequence and without duplicated packets, users can easily access the data base to have a full picture of the network and add more statistical tools to further show the network insights.

A Large Scale Test Case Similar results of the controlled deployment described above have been obtained in a larger scale deployment in the frame of the EM-

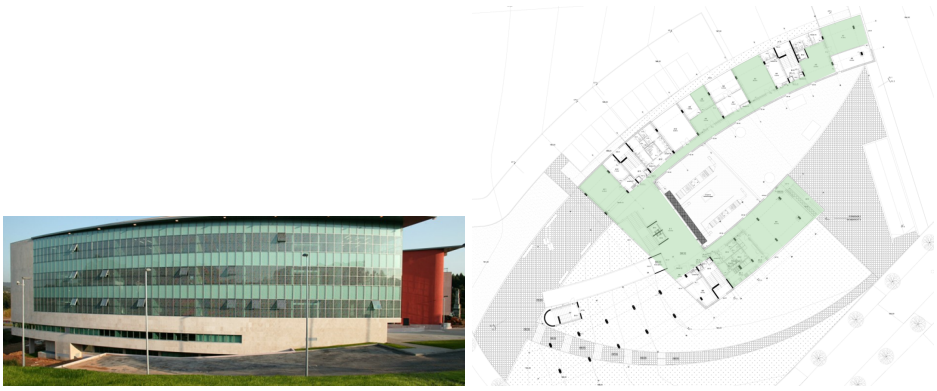

Figure 12: Location of EMMON final review at SANJOTEC Business and Technology Park in Sao Joao da Madeira (PT) [9].

MON project [7, 8]. EMMON is an FP7 Artemis European project, whose consortium was composed by 9 institutions (both from academia and industry) from 6 EU Countries. The aim of the project, which ended in 2012, was to demonstrate tools and methodologies for large scale and dense real-time networked embedded systems.

For the final review, the consortium has deployed 400 TelosB nodes over 3 floors of an office building to show the performance of the EMMON network architecture (Figure 12) to monitor environmental parameters, such as humidity and temperature.

The network was divided into several patches, each one running on a different IEEE 802.15.4 frequency channel to keep the collision probability low. Each patch includes a gateway, i.e., a notebook equipped with a TelosB to communicate with the WSN and connected to the WLAN through the access points available in the building premises. Monitoring data were reported from the sensor nodes to a central server station, placed in a room on the second floor of the building, through such gateways. The server was reachable by external clients through a public IP address. Other than the multi-hop and multi-tiers hierarchical networking architecture, the project encompassed several aspects, such as the design and implementation of an efficient and distributed middleware for data handling and service providing, and the implementation of a user friendly interface on the server to allow clients to interact with the network on a query-based mechanism. Overall, the EMMON project included all aspects related to the full system design and implementation.

On the networking side, Z-Monitor was used to collect statistics and assess the performance of such a large scale and very dense WSN-based system. In particular, 6 sniffers were running several days on each floor to report on the activities of the deployed nodes. These nodes were organized into 4 patches of 100 nodes each. The communication protocol implemented was the ZigBee Cluster Tree, rooted at the gateway and composed by 12 to 15 clusters. By referring to

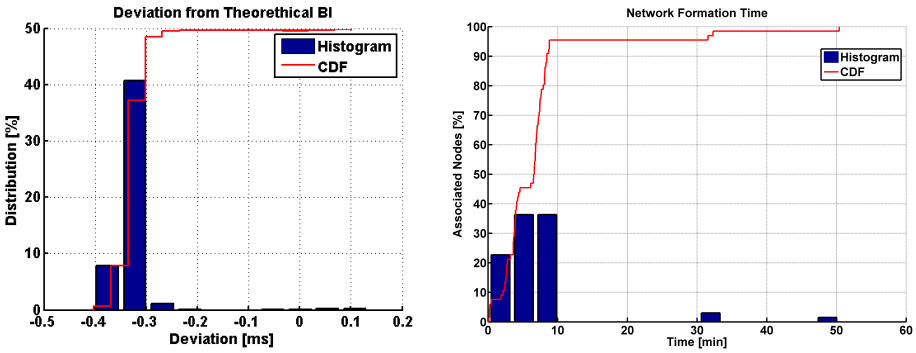

Figure 13: (a) Beacon Deviation measured in the real world and large scale deployment; (b) Network formation time in the large scale deployment.

the same symbols defined above, the network was configured according to the following parameters: *(i)* $L_m = 3$ hops; *(ii)* $R_m = 4$; *(iii)* $C_m = 15$; *(iv)* $BO = 10$, which means a beacon interval BI of roughly 15 seconds; *(v)* $SO = 5$.

Figure 13a and Figure 13b show the performance of this deployment and basically confirm the results described above in the smaller scale scenario. More in details, Figure 13a demonstrated that the beacons sent by each local coordinator (i.e., the gateways or the routers) show a deviation from the theoretical value by less than 0.5 ms. Figure 13b reports on the network formation time, i.e., when a node associates with the parent in the ZigBee Cluster Tree mode. This figure demonstrates that for allowing such large network deployment to run smoothly the network formation time was quite high, but in line with the expectations. In particular, it is worth to note that over 90% of the nodes for each patch were able to join the network in less than 10 minutes.

5 Conclusions and Future Works

In this paper, we have presented Z-Monitor, a WPANs monitoring and protocol analysis framework. We have demonstrated the capabilities of Z-Monitor to monitor the behavior and evaluate the performance of COTS WPANs technologies, namely IEEE 802.15.4 and ZigBee protocols. We have successfully monitored distributed networks in two real scenarios: a laboratory environment at I3ASensorBed [5] and a multi floor building at San Jotec [9] in the frame of the EMMON project [7]. Thus, we have proved the new distributed features of Z-Monitor.

We are currently working towards extending Z-Monitor to support more advanced features including *(i)* extending parsing component to support new COTS protocols implementations, such as TinyRPL [10], *(ii)* integrating advanced filtering and statistical analysis features, and *(iii)* improving the topology view interface.

Overall, we believe that Z-Monitor provides a convenient solution for researchers and students for developing, debugging and deploying wireless sensor network applications based on LoWPANs. This is also confirmed by the fact that Z-Monitor is being already used by several Universities and research centers around the world, as we have witnessed more than 500 downloads since its release.

Acknowledgments

This work was funded by (*i*) Al-Imam Mohamed bin Saud Islamic University (IMAMU), (Riyadh, Saudi Arabia) within the R-Track project under the grant 8-INF-2008 of the National Plan for Sciences and Technology (NPST); (*ii*) partially supported by Portugal National Funds through FCT (Portuguese Foundation for Science and Technology) and by ERDF (European Regional Development Fund) through COMPETE (Operational Programme "Thematic Factors of Competitiveness"), within the MASQOTS project, ref. FCOMP-01-0124-FEDER-014922; and (*iii*) jointly supported by the MINECO and European Commission (FEDER funds) under the project TIN2012-38341-C04-04. We also thank Nada Al-Elaiwi, Hanan Al-Soli, Rihab Chaari, Ahmed Dkhil and Mossab Alsania for programming earlier versions of Z-Monitor.

References

1. IEEE 802.15.4 Standard, Part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs), IEEE-SA Standards Board.
2. Z-Monitor: A Monitoring Tool for IEEE 802.15.4 WPANs (2011). URL http://www.z-monitor.org
3. D. Mills et al., Network time protocol version 4: Protocol and algorithms specification. URL https://tools.ietf.org/html/rfc5905
4. Open-ZB open-source toolset for the IEEE 802.15.4/ZigBee protocols. URL http://www.open-zb.net
5. M. O. Antonio et al., I3asensorbed: a testbed for wireless sensor networks, in: Technical Report DIAB-11-12-1, Departamento de Sistemas Informticos. UCLM, 2011.
6. S. Tennina et al., The dark side of demmon: what is behind the scene in engineering large-scale wireless sensor networks, MSWiM '11, ACM, New York, NY, USA, 2011, pp. 41–50.
7. Emmon - embedded monitoring, www.artemis-emmon.eu (May 2012). URL www.artemis-emmon.eu/
8. S. Tennina et al., Emmon: A wsn system architecture for large scale and dense real-time embedded monitoring, IFIP EUC2011 pp. 150 –157. doi:10.1109/EUC.2011.32.
9. D. Almeida, EMMON open day 2012 at sanjotec, EMMON Open Day 2012 event in SANJOTEC Business and Technology Park in Sao Joao da Madeira, Portugal (June 2012). URL www.youtube.com/watch?v=OZE7o8CVIdA
10. J. G. Ko et al., Evaluating the Performance of RPL and 6LoWPAN in TinyOS, IPSN.