A Tool for Real-Time Assessment of IEEE 802.15.4 Networks Through Fault Injection *

Rui Pedro Caldeira, Jeferson L. R. Souza, Ricardo C. Pinto, and José Rufino

LaSIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal
{rcaldeira, jsouza}@lasige.di.fc.ul.pt,
{ricardo.pinto, jmrufino}@ciencias.ulisboa.pt

Abstract. Advances in computer engineering and microelectronics have allowed the use of tiny and powerful computing platforms (i.e., sensors and actuators) everywhere, supporting the monitoring and control of, for example, process for industrial automation and functions within aerospace vehicles. Many of these systems have the ability to host, in the same computing platform, applications with different levels of criticality (or importance), i.e. mixed-critical systems. Wireless sensor and actuator networks (WSANs) become the vivid example of computer networks responsible for the monitoring and control activities of such systems. The dependability and the real-time properties of such networks are crucial. However, one key point is that WSANs are extremely susceptible to communication errors induced by electromagnetic interferences. Furthermore, there is a general lack of knowledge of such error patterns as well as no open tools enabling its capture. This paper presents a state of the art solution for one-hop assessment of WSANs in the presence of errors based on the IEEE 802.15.4 standard. The solution includes devices and functions to monitor the behaviour of the network as well as methods to emulate accidental errors and to perform intentional attacks. All these resources are managed and controlled by a customised version of the well-known open-source Wireshark network protocol analyser. This allows the generation of network error reports fundamental to the evaluation of the real-time capabilities of current wireless network protocols and standards. These error reports contribute to a better knowledge of the error characteristics of WSANs and therefore enable the design of more robust and resilient solutions for WSANs operation.

1 Introduction and Motivation

Wireless Sensor and Actuator Networks are one of the latest revolutions in networking. The absence of cables stem a reduction of Size, Weight and Power Consumption (SWaP) which combined with node mobility, lead to the adoption of these networks as a fundamental communication platform of many different types of systems that may host applications with different levels of criticality (or importance), which are usually

* This work was partially supported by FCT, through project PTDC/EEI-SCR/3200/2012 (READAPT) and through LaSIGE Strategic Project PEst-OE/EEI/UI0408/2014. This work integrates the activities of COST Action IC1402 - Runtime Verification beyond Monitoring (ARVI).

known as mixed-critical systems. Despite some advances in the support of real-time communication in wireless networks, there are still open problems that need the appropriate tools for their study and analysis. One key issue is that wireless networks are subjected to electromagnetic interferences from the surrounding environment that may impair ongoing communications. The presence of interferences can endanger the real-time guarantees of the communications as well as of the overall system. This paper discuses a highly flexible advanced open tool that allows the real-time assessment of one-hop IEEE 802.15.4 networks through the combination of network monitoring (e.g., for the capture of error patterns) with the emulation of accidental faults or even the injection of intentional attacks. The assessment of the network operation, and therefore the contributions of this paper, includes the facets that we detail next.

Network monitoring, which is a technique aimed at analysing the network interactions between its nodes and characteristics of operation such as reliability and timeliness, assumes a passive and non-intrusive role at network assessment. In particular, it allows to assess normal network operation and the disturbances in its behaviour in the presence of errors. Since particular error patterns may be specially relevant for the analysis of the network operation and their natural occurrence may be rare, there is the need to re-enact such error patterns, through fault injection.

Our fault injection methodology is aimed at directly inject interferences in the wireless transmission medium with the objective to stress test the network. Fault injection is split into two types: accidental faults and intentional attacks. The first type focuses in reproducing conditions that can be found in reality. This is very helpful in the sense that allows to test a network at will in a controlled environment, observe its behaviour, and stress the operation of network related protocols. The second is focused in controllably disturb a network, with a malicious objective. Objectives often revolve around data extraction, network unavailability or other actions relevant to the party conducting such disturbances. In this work we focus on accidental faults, and they can be achieved in two ways. Firstly, through the transmission of particular symbol patterns without obeying to the medium access control protocol in use. This translates to the injection of unrecognised noise. Secondly, in situations where it may be useful to also inject symbol patterns that are interpreted by the remaining nodes, we also support the injection of properly formatted traffic. Furthermore, combining network monitoring with fault injection allows to trigger a specific set of fault injection actions upon the occurrence of particular network traffic patterns. This coordinated action allows an active role in network assessment.

Network monitoring activities can be a very data-intensive task, specially when conducted concurrently with fault injection activities. Analysing the harvested data is simply too difficult to do without assistance. Wireshark [7], being a reference network protocol analyser, has the ability to create a graphical representation of network interactions, thus assisting in the network analysis task by being an integrated Graphical User Interface (GUI) for the whole suite.

The remainder of the paper is structured as follows. Section 2 reviews some related work. Section 3 describes the real-time assessment suite and its components. Section 4 describes the implementation details for the suite. Section 5 walks through some simple use cases and finally section 6 concludes the paper and describes the future work.

2 Related Work

Monitoring and fault injection are techniques often used in conjunction to perform dependability evaluation of computer systems and networks.

In a more theoretical approach, KleeNet [16] is a verification framework much similar to a simulation or model-checking. The framework acts as an Hardware Abstraction Layer (HAL) where the software to be deployed can run and be tested against a series of assertions that model the correct operation of the system. That said and reiterating the initial remark, all results remain theoretical and the behaviours of individual nodes as well as the medium are still based on models and to accurately model a complex environment such a Wireless Transmission Medium (WTM) is near to impossible.

Another interesting development is a hybrid architecture that allows to transfer state from test-bed nodes to simulation platform and vice-versa [14]. However, this approach is much complex because requires physical access to the test-bed hardware as well as access to a simulation platform that simulates both the computing hardware and the wireless transmission medium. Additionally to these requirements, middleware has to be developed in order to support the state transfer operations.

Due to WSANs growing popularity, researchers felt the need to create tools like the ones described earlier to assist them in various research activities. Regarding network monitors, popular use cases for this type of devices are, for instance, those described in [12,15]. However, these works focus in network performance evaluation, paying little to no attention to communication errors and to the stress of network operation via fault injection campaigns. In a theoretical fault injection approach and with network security in mind, strategies for the injection of attacks in wireless networks were devised. Xu et al. [20] raised concerns about how insecure WSANs are. A wide range of attack strategies were defined in [20] taking into account the scarcity of power resources in wireless devices.

O'Flynn [13] developed a dual-transceiver device capable of very precise attack injection techniques, but the control software has focused in approaches that maximize the reaction time in order to jam the highest number of packets possible and not in creating complex fault injection scenarios.

JamLab [6] is an addition to WSAN test-beds that allows to record and reproduce real interference patterns (or signals) as well as emulate some real world devices such as microwave ovens, IEEE 802.11 (Wi-Fi) and IEEE 802.15.1 (Bluetooth) devices. They also recognize that network robustness against interferences and packet loss is crucial to a correct functioning of the network and interferences can cause the timing constraints not to hold. Their study is focused in measuring the Received Signal Strength Indication (RSSI), the Link Quality Indication (LQI) and its impact in the Clear Channel Assessment (CCA). They proved that interferences have a great deal of impact in Packet Reception Rate (PRR) and if packet retransmission is required, power consumption is dramatically increased. JamLab, however, focuses in disturbing physical (PHY) layer operation by re-generating signals in the same frequency and with the same intensity as first detected. One major limitation of JamLab and of other studies that focus on physical layer interference [4,5,8], is that there are no simple methods of mapping such interferences into packet/frame losses. Our tool addresses this limitation by focusing on a packet/frame level approach.

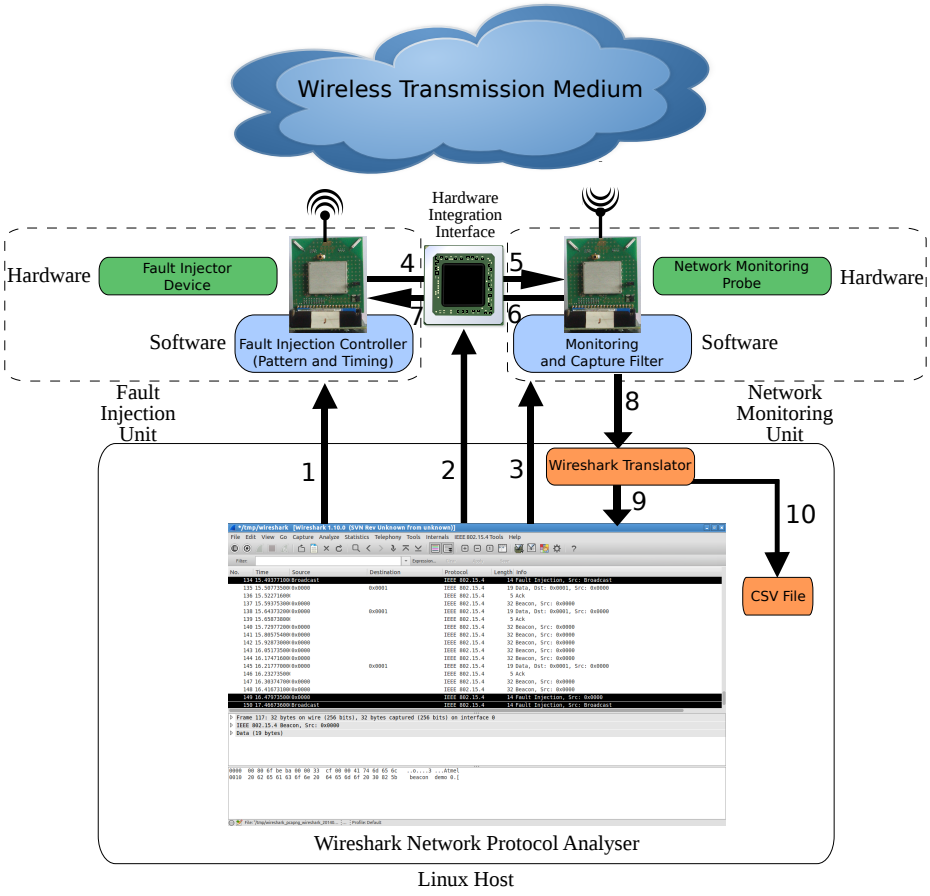


Fig. 1. Tool Architecture Design

3 A suite for Real-Time Assessment of IEEE 802.15.4 Networks

In this section we address in detail each component presented in our architecture and how they all integrate following closely the diagram in Figure 1.

This tool for the real-time assessment of IEEE 802.15.4 networks is composed by four components. Firstly, with the purpose of capturing and report network traffic, a network monitoring unit, which assumes the role of an error-aware packet sniffer was developed. All network monitors provide a very basic function of capturing network traffic without interacting with it. However, such capture is usually restricted to correctly formatted packets. Existing network monitors, to the best of our knowledge, do not capture and signal the occurrence of errors and this is one of the limitations we overcome with our design.

In contrast, fault injection devices can be described as devices that are able to interact with the underlying network but are not obliged to follow the medium access proto-

cols used in that particular network. In short, fault injection devices have the ability to interact with the network without any restrictions in regards of timing and/or content. And so, in order to provide the ability to emulate accidental faults and to perform intentional attacks, a fault injection unit has been built to flexibly support the injection of faults directly in the wireless transmission medium.

Thirdly, to provide interaction and cooperation between the network monitoring unit and the fault injection unit, as well to support advanced network analysis and filtering functions, a special purpose hardware integration interface was introduced.

Finally, Wireshark, a reference network protocol analyser with IEEE 802.15.4 packet visualization capabilities, was extended to manage all aspects of the network assessment, including control and data collection from the previously mentioned components.

3.1 Network Monitoring Unit

The Network Monitoring Unit includes a Commercial-Of-The-Shelf (COTS) hardware network interface device, acting as network monitoring probe (a.k.a. sniffer) with the purpose of capturing all traffic transmitted through a specific wireless radio channel. This is useful for testing networks and protocol implementations. The network monitoring probe is not an active member of the network, and is instead a passive listener. This is compliant with the IEEE 802.15.4 standard when using the so called *Promiscuous Mode* with the *integrity check disabled*: the network monitoring probe is allowed to capture and report all traffic following the IEEE 802.15.4 frame format specification. However, this is insufficient to assess the network error pattern since frames sensed with errors would not be specifically signalled and therefore would be handled as any other frame. Both correct and erroneous frames would be equally reported.

Wireless networks, due to the open nature of the transmission medium are extremely susceptible to electromagnetic interference (EMI) and therefore to frame errors. In order to capture and signal frames corrupted by errors, the operation of the network monitoring probe needs to be enhanced.

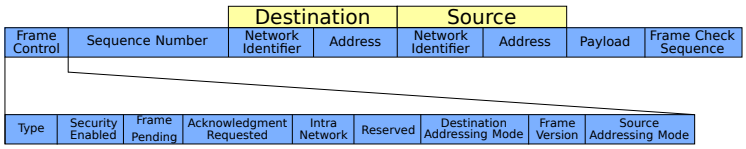


Fig. 2. IEEE 802.15.4 General Frame Format

The IEEE 802.15.4 traffic is constituted by frames following the general format represented in Figure 2. The Frame Check Sequence (FCS) field is a 16-bit frame integrity number used to evaluate the correctness of each captured frame. Deviations from the original content, e.g. resulting from corruption, can be detected through FCS checking mechanisms. The probability of undetected frame errors is negligible [9].

A simple, yet fundamental, *extension of this FCS integrity check mechanism* [18], enabled by modern network interface controllers such as the Atmel AT86RF232 [1],

allows to detect and signal erroneous frames. By taking advantage of the *FCS extension*, that allows us to signal corrupted traffic, and of the previously mentioned *promiscuous mode*, that allows us to capture all traffic, we are able to capture **all** network traffic that flows though the wireless transmission medium and to distinguish correct frames from those affected by errors.

This ability is **fundamental** to evaluate the error characteristics of a given network and can be used to estimate the impact of such error characteristics in the normal network operation.

To enable a precise evaluation of network operation timing characteristics, a timestamp is generated at the network monitoring unit at the arrival of each frame, including erroneous frames. This timestamp is attached to the frame and both are delivered at the Hardware Integration Interface, via the interaction 6 of Figure 1, and at the Wireshark translator, via the interaction 8 of Figure 1.

Concluding, the network monitoring unit offers to the surrounding components an extended promiscuous network traffic capture service that includes both correct and erroneous frames as well as the specific signalling of the latter.

3.2 Fault Injection Unit

The Fault Injection Unit is constituted by a hardware network interface, the fault injection device, controlled by a software component that runs on the fault injection controller, as illustrated in the left uppermost part of the Figure 1.

The fault injector device is a perfectly common Commercial-Of-The-Shelf (COTS) network interface with the exception that the network interface is configured to bypass the medium access control protocol thus allowing a direct access to the wireless transmission medium. Traditional IEEE 802.15.4 nodes use the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) mechanism to sense the medium before transmitting, which is therefore disabled.

Disabling the medium access protocol allows effective fault injection actions. This way, the fault injector device supports the injection of user-defined:

- Pure noise patterns;
- Selected data patterns, always preceded by the standard preamble[1];
- Selected data patterns encapsulated in a correctly formatted frame, thus injecting a standard compliant transmission.

The Fault Injection Controller is a software component executing on top of the hardware fault injection device with the responsibility to control the patterns and timings to be utilized in a fault injection campaign.

The fault injection unit has two modes of operation: user-defined configurations or pre-configured fault injection profiles.

Both modes originate the fault injection sequence illustrated in Figure 3 accordingly with the parameters that are now explained further.

[1] The preamble is a pre-defined sequence of symbols for synchronization of the receiver's circuitry with the incoming sequence of symbols.

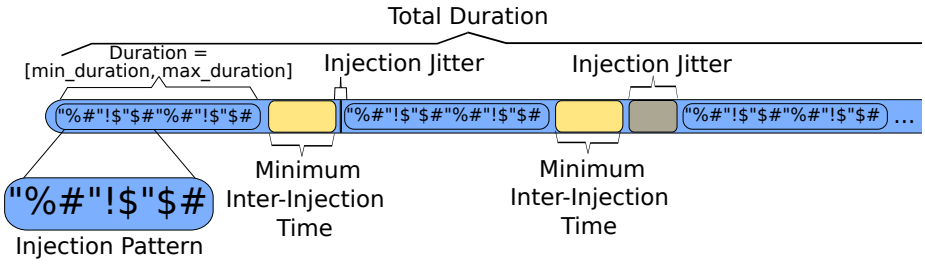


Fig. 3. Graphical representation of some fault injection parameters

Fault injection parameters User-defined configuration requires the specification of some parameters in order to build a particular fault injection profile. Some of the parameters are illustrated in Figure 3 and explained further. For future reference, and for the particular case of the IEEE 802.15.4, all time units are to be considered as being in microseconds.

- **Injection Mode -** The injection mode parameter determines how the potentially interfering data is sent to the wireless transmission medium. Firstly, when the noise mode is selected, the device simply sends a given sequence of symbols (injection pattern) to the wireless transmission medium until the injection event duration finishes. Secondly, in the preamble preceded mode, the user selected data is attached to the standard preamble and sent to the wireless transmission medium. Finally, in the encapsulated frame mode the user can configure all the fields of a correctly formatted frame;
- **Injection Pattern -** (see Figure 3) the format of the user-defined data, in hexadecimal, to be injected to the wireless transmission medium. Should a preamble be used (preamble preceded and encapsulated frame modes), it is attached before the injection pattern; otherwise (noise mode), only the injection pattern is sent. If the defined duration is lower than the time necessary to transmit the pattern this should be trimmed and periodically re-transmitted otherwise;
- **Minimum Duration -** the minimum duration of a single fault injection event. It can be expressed in time units or in bytes;
- **Maximum Duration -** the maximum duration of a single fault injection event. It can be expressed in time units or in bytes. If the minimum and maximum durations are equal, the duration of the fault injection event is equal to the previously stated values. Otherwise, it is equal to a value dictated by a given statistical distribution between the minimum and maximum durations;
- **Number of Events -** number of total fault injection events in the fault injection campaign, after which the campaign shall terminate;
- **Minimum Inter-Injection Time -** (see Figure 3) the minimum time interval between consecutive fault injection events;
- **Maximum Injection Jitter -** (see Figure 3) maximum value of a positive random time to be added to the Minimum Inter-Injection time, dictated by a given statistical distribution;

- **Total Duration -** (see Figure 3) the total duration of a fault injection campaign. If the number of events is specified, the user no longer is able to set its value, since it is confined to a lower as well as an upper bound dependent on the specific number of events, its (real) durations, its minimum inter-injection time and its (real) injection jitter durations. Conversely, if the total duration is specified, the fault injection campaign lasts until the specified duration is reached. The Total Duration is an alternative to Number of Events parameter described earlier, and for that reason, both parameters cannot be specified for the same fault injection campaign;
- **Trigger -** the condition to start the fault injection campaign using the previously described parameters. The trigger can be defined to operate according two modes: one-shot mode, where only one instantiation of the fault injection campaign is performed; and the cyclical mode, where the fault injection campaign is repeated cyclically until the stop command is explicitly issued.

Pre-configured fault injection profiles Some combinations of these parameters are so relevant that we have decided to include the capability to chose between a set of specific instantiations of the previously described parameters, thus defining pre-configured fault injection profiles. These profiles are useful both for the emulation of accidental faults and for the injection of intentional attacks.

- **Constant -** this profile continuously injects constant noise in the wireless transmission medium. This profile is meant for inducing communication blackouts such as jamming attacks [20];
- **Random -** this profile injects random noise on the wireless transmission medium. This profile is better suited for campaigns where it is only required the occasional corruption of transmissions [20];
- **Adaptive -** This profile tries to synchronize the fault injection timing with the traffic pattern of the underlying network. Thus the fault injection campaign is triggered by network monitoring unit after a specific type of traffic pattern has been detected by the monitoring and capture filter (see Figure 1);
- **Frame-type Adaptive -** specialisation of the **Adaptive** fault injection profile, where faults are injected to destroy the frames matching the specification(s) indicated to the network monitoring unit. This profile is specially useful when, for example, aiming to corrupt only beacon frames causing one-hop wide network blackouts [17].

A summary of the relevant fault injection parameters for each of the profiles is represented in Table 1.

One of the key features of this Fault Injection Unit is its flexibility when defining fault injection scenarios. The Fault Injection Unit is already equipped with several easy-to-use pre-defined scenarios while at the same time allowing a user to use configuration parameters if customisation is required. The presented fault injection profiles are suitable in the sense that they are practical implementations of well-known wireless network attacks which are also useful to emulate some accidental fault scenarios [20].

	Injection Mode	Minimum Duration	Maximum Duration	Total Duration	Number of Events	Minimum Inter-Injection Time	Injection Jitter	Trigger
Constant	Noise	∞	∞	∞	-	0	0	One-Shot
Random	Noise	Random		-	-	Random	0	One-Shot
Adaptive	Noise	19 bytes	133 bytes	-	1	-	0	Cyclical
Frame-Type Adaptive	Noise	19 bytes	133 bytes	-	1	-	0	Cyclical

Table 1. Fault Injection Profiles expressed as Fault Injection Parameters

3.3 Hardware Integration Interface

The hardware integration interface is a component that enables interaction and cooperation between the network monitoring unit and the fault injection unit, as shown in Figure 1.

The first aspect of this interaction concerns the analysis and filtering of captured frames. In essence any frame field, such as frame-type, addresses and some contents of the payload, can be subjected to analysis. Selected configurations of the Fault Injector Unit such as pre-configured adaptive and the frame-type adaptive fault injection profiles requires analysis of the captured traffic and in these occasions cooperation between both units is required to trigger a fault injection campaign, as shown by interaction 6 of Figure 1, where the network monitoring tool hands frames for analysis to the hardware integration interface, and interaction 7 of Figure 1 where the hardware integration interface decides to start a fault injection campaign.

The analysis of the captured traffic allows to detect relevant pre-configured traffic patterns to be used to trigger fault injection campaigns. An effective and highly flexible analysis and filtering of network traffic can be directly mapped into special-purpose hardware components, such as Content Addressable Memories (CAMs), integrated into Field Programmable Gate Arrays (FPGAs) [19]. The possibility of integrating these types of special-purpose hardware components with the tool greatly enhance the reaction capabilities of the tool in the sense that these components enables parallel comparison and matching of the captured traffic, against several specified frame patterns.

On the other hand, the start and the end of each fault injection event needs to be issued to the network monitoring unit to be timestamped, ordered, and inserted into the traffic flow to be delivered at the Wireshark Translator, represented by interaction 4 of Figure 1 where the notification is sent to the hardware integration interface and interaction 5 of Figure 1 where the notification is forwarded to the network monitoring unit.

3.4 Integration with Wireshark

Wireshark is a very flexible reference network analyser. It is extremely general in the sense that it can be extended by the means of plugins to analyse networks and protocols

that were not initially considered in its design and engineering. Wireshark can monitor traffic both from real and virtual devices such as networking hardware, regular files and even inter-process communication channels. Beyond that, the captured data can be saved for later analysis. Based on these reasons, Wireshark was chosen to be extended in order to incorporate the control functionalities of the network monitoring and fault injection units.

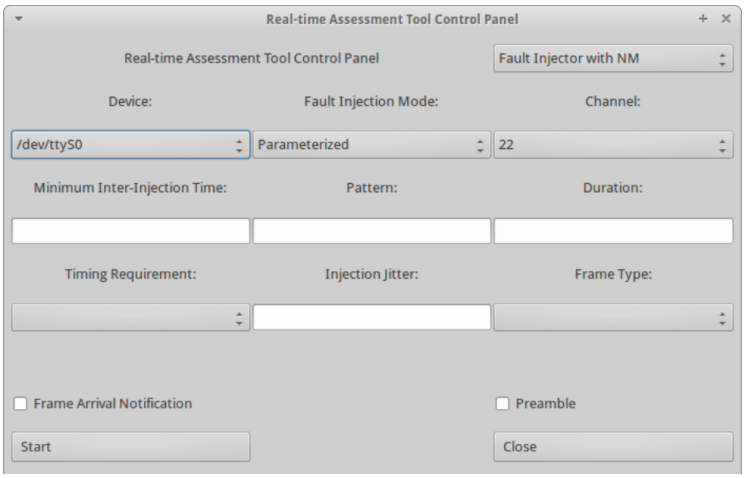


Fig. 4. Integrated Control Panel

Firstly, Wireshark communicates directly with the units in order to perform initialisation and command functions. The Fault Injection and the Network Monitoring Units are configured with the channel to be used in their respective activities, represented by the interactions 1 and 3 of Figure 1. In addition, the hardware integration interface is configured with all network monitoring and fault injection variables, represented by the interaction 2 of Figure 1, in order to correctly coordinate the interaction between the Network Monitoring and Fault Injection Units. In this sense, an extension to the Graphical User Interface was introduced to create a user-frendly interface to ensure that users can configure the units in an easy way. The result of this extension is presented in Figure 4. The figure presents the most complex case of the configuration of a fault injection campaign, accompanied with a network monitoring activity (Fault Injector with NM, in the top-right corner of Figure 4) with manual parameter specification (see section 3.2), including the path to the Network Monitoring Unit (below the *Device* label in Figure 4), and the path to the Fault Injection Unit which is derived as the device immediately following the Network Monitoring Unit. This establishes the communication paths between Linux/Wireshark Software Components and the underlying hardware units, as required for interactions 2, 3 and 8 (Network Monitoring Unit) and interaction 1 (Fault Injection Unit), depicted in the diagram of Figure 1.

After initialization, the network monitoring unit delivers captured frames to Wireshark indirectly. First, the captured frame arrives the Wireshark Translator (represented by interaction 8 in Figure 1) which is then adapted to the Wireshark format and promptly forwarded to Wireshark, as shown by interaction 9 of Figure 1.

The Wireshark Translator is a piece of software which has the responsibility to translate the raw frames collected from the network monitoring unit into data understandable by Wireshark. This process requires that a PCAP header [11], the format used by Wireshark, is attached to the raw frame so that Wireshark is aware of, among other aspects, the network type of the frame and its length.

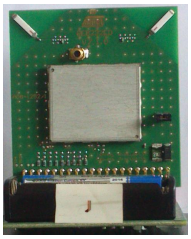


Fig. 5. Atmel REB232ED-EK

3.5 Statistics File

The Wireshark Translator application also creates a statistics file in the form of Comma-Separated Values (CSV), represented by interaction 10 of Figure 1, which is recognized by most statistical data applications. The file includes a timestamp, signalling when the frame starts and its duration, as well as the length of the frame, if the frame passed the checksum verification, and the frame itself. To simplify statistics file post-processing (e.g., inter-frame time and jitter calculation), a second timestamp signalling when the frame ends is also included. In this way, all relevant information regarding the frame is included in the same place. This file is useful in providing a base for additional processing that may not be achieved using Wireshark.

4 Implementation

Built using Commercial Off-The-Shelf (COTS) hardware, our implementation currently supports the Atmel REB232ED-EK (Figure 5) Evaluation Kit [3]. The hardware provides serial interface in order to connect to a computer host and exchange data.

Both units were implemented so they could be easily configured by Wireshark. In Figure 4, it is shown the integrated control panel. To configure a network monitoring session, it is required to select the path of the hardware, as well as the wireless radio channel to be monitored and the function in the top right corner of Figure 4 (e.g., standalone network monitoring or fault injection with network monitoring).

The monitoring starts by signalling the Network Monitoring Unit (via interaction 3 of Figure 1) and starting up the Wireshark translator. After a frame capture is successful, elements including the frame length and timestamps are attached to the frame and sent out via the serial connection (via interaction 8 of Figure 1).

In this work we choose to output the raw data and not commit to any specific data output format (e.g., the PCAP format) in order to save the scarce input/output resources of the Atmel hardware supporting both the Network Monitoring and the Fault Injection units and to allow any application to read the traffic and perform any kind of processing and analysis. The Wireshark translator has the function to conform the Network Monitoring Unit output to the PCAP format and forward it to Wireshark.

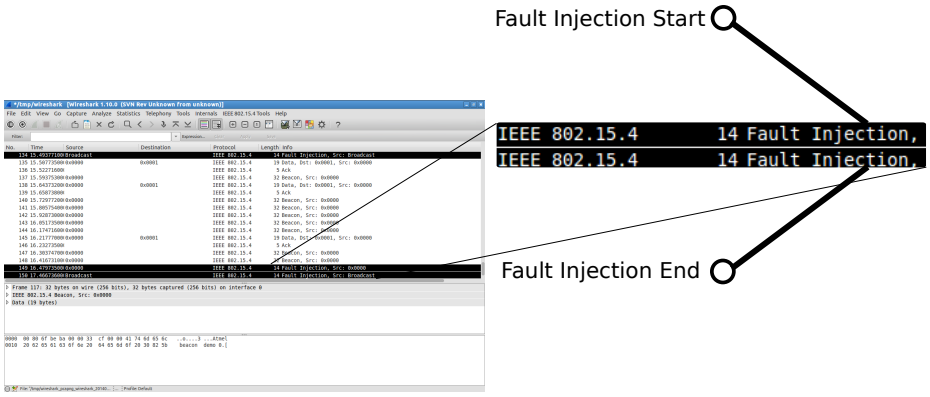


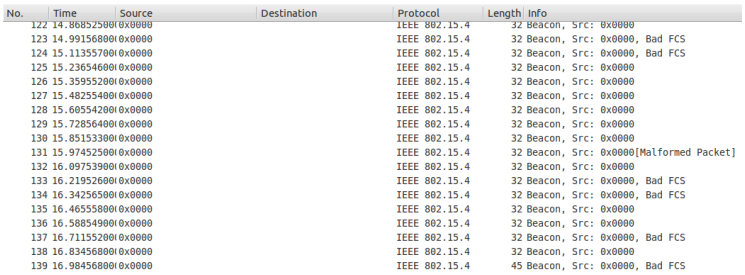
Fig. 6. Wireshark capture screen

With this control panel is also possible to define fault injection campaigns making use of the parameters or modes described in section 3.2. A fault injection campaign is constituted by a series of fault injection events. In the Wireshark screen (see Figure 6), additional elements (represented as black shaded highlighted frames) were introduced to Wireshark to delimit fault injection events. These elements, represented in Figure 6, notify the users when a fault injection event begins and when it ends, representing between them the frames that arrived during the fault injection. In the particular case of the example in Figure 6, the Constant fault injection profile (Section 3.2/Table 1) was selected. Since pure noise is constantly injected in the wireless transmission medium; no frame is received by any node, and therefore no frame is captured during the fault injection campaign. It is worth noting that the IEEE 802.15.4 interpretation capabilities of Wireshark were extended in order to make sure that these new elements were recognized and all functionalities of Wireshark, such as filtering and colouring, also apply to these new elements.

5 Use Cases

This section discusses two simple, yet illustrative use cases: firstly, an example emphasising the network monitoring functions while the second focuses in a fault injection campaign.

5.1 Standalone Network Monitoring



No.	Time	Source	Destination	Protocol	Length	Info
122	14.86852500	0x0000		IEEE 802.15.4	32	Beacon, Src: 0x0000
123	14.99156800	0x0000		IEEE 802.15.4	32	Beacon, Src: 0x0000, Bad FCS
124	15.11355700	0x0000		IEEE 802.15.4	32	Beacon, Src: 0x0000, Bad FCS
125	15.23654600	0x0000		IEEE 802.15.4	32	Beacon, Src: 0x0000
126	15.35955200	0x0000		IEEE 802.15.4	32	Beacon, Src: 0x0000
127	15.48255400	0x0000		IEEE 802.15.4	32	Beacon, Src: 0x0000
128	15.60554200	0x0000		IEEE 802.15.4	32	Beacon, Src: 0x0000
129	15.72856400	0x0000		IEEE 802.15.4	32	Beacon, Src: 0x0000
130	15.85153300	0x0000		IEEE 802.15.4	32	Beacon, Src: 0x0000
131	15.97452500	0x0000		IEEE 802.15.4	32	Beacon, Src: 0x0000[Malformed Packet]
132	16.09753900	0x0000		IEEE 802.15.4	32	Beacon, Src: 0x0000
133	16.21952600	0x0000		IEEE 802.15.4	32	Beacon, Src: 0x0000, Bad FCS
134	16.34256500	0x0000		IEEE 802.15.4	32	Beacon, Src: 0x0000, Bad FCS
135	16.46555800	0x0000		IEEE 802.15.4	32	Beacon, Src: 0x0000
136	16.58854900	0x0000		IEEE 802.15.4	32	Beacon, Src: 0x0000
137	16.71155200	0x0000		IEEE 802.15.4	32	Beacon, Src: 0x0000, Bad FCS
138	16.83456800	0x0000		IEEE 802.15.4	32	Beacon, Src: 0x0000
139	16.98456800	0x0000		IEEE 802.15.4	45	Beacon, Src: 0x0000, Bad FCS

Fig. 7. Standalone Network Monitoring

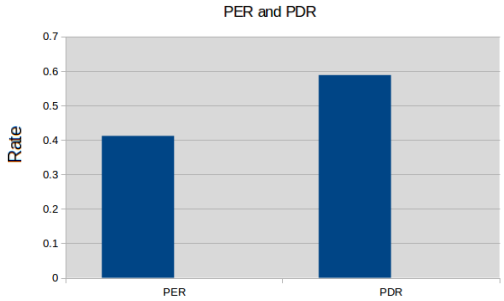


Fig. 8. Chart representing the Packet Error Rate (PER) and the Packet Delivery Rate (PDR) of a IEEE 802.15.4 network under heavy interference, as generated by the interpretation of the statistics file

The first use case, illustrates the functionality of our tool in the traffic monitoring of a IEEE 802.15.4 network. However, this network is operating in a ”dirty environment” where a heavy electromagnetic interference is expected from ”alien” wireless nodes.

One of the main sources of interference on the IEEE 802.15.4 channels from wireless ”alien” nodes results from the coexistence with IEEE 802.11 nodes. Since both standards operate in the same 2.4 Ghz Industrial, Scientific and Medical (ISM) bands the probability, in some cases, of both protocols interfere is high [10].

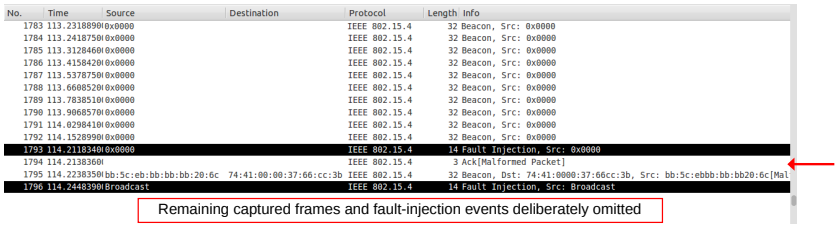


Fig. 9. Network monitoring with Fault Injection

To increase the likelihood of interference, we have set the IEEE 802.15.4 coordinator to channel 17 (2.435 Ghz) which is the closest channel to an ”alien” IEEE 802.11 access point operating in channel 6 (2.437 Ghz). The IEEE 802.15.4 network operates under very lightweighted load conditions, with the network coordinator transmitting mostly beacon frames.

The results from our analysis, inscribed in Figure 7, show a high number of corrupted frames, some of them grouped in bursts of interference. However, these error bursts do not violate (at least in this experiment) the upper bound of three consecutive frame errors defined in the IEEE 802.15.4 standard specification [2].

Furthermore, the statistics file generated was used to be interpreted in a LibreOffice Calc Spreadsheet equipped with functions to automatically parse the data and generate charts, like the one represented in Figure 8. This chart represents the Packet Error Rate (PER), calculated by dividing the number of corrupt frames by the total of captured frames, and the Packet Delivery Rate (PDR), calculated by dividing the number of correct frames by the total of captured frames, based on data shown in Figure 7.

5.2 Network monitoring with Fault Injection

In this experiment, intended to demonstrate the functionality and effectiveness of the fault injection methodology, the same IEEE 802.15.4 network operates in a ”clean environment” where only some occasional frame errors, due to the natural electromagnetic interference from the environment, are expected. However, a fault injection campaign was conducted, with the following parameters:

- **Injection Mode:** Preamble Preceded Symbols
- **Injection Pattern:** AABBCC
- **Minimum Duration:** 28000 microseconds
- **Maximum Duration:** 28000 microseconds
- **Number of Events:** 100
- **Minimum Inter-Injection Time:** 1000000 microseconds
- **Maximum Injection Jitter:** 0 microseconds
- **Total Duration:** Not Defined
- **Trigger:** Cyclic

After this parametrisation, the Fault Injection Unit transmitted the pattern *AABBCC* preceded by the IEEE 802.15.4 preamble in a loop during 28000 microseconds, then

waited 1000000 microseconds. This sequence happened 99 more times as stated in the Number of Events parameter. Simultaneously, the network monitoring unit listened to all these interactions then were passed on to Wireshark following the interactions 8 and 9 of Figure 1. An excerpt of the results of this fault injection campaign is illustrated in Figure 9, where the black backgrounded rows signal the begin and the end of a fault injection event. The beacon frame transmitted by the network coordinator and signalled by the red arrow at the right of Figure 9 has been corrupted by the fault injection event. Other captured frames and fault injection events have been deliberated suppressed from the screen of Figure 9, using the wireshark own filtering facilities. It is worth noting that in this context the frames designated as "Ack" in the Wireshark packet list window actually represents the fault injection. The AABBCC pattern when analysed by the Wireshark internal components (in particular, by the Wireshark dissector) is considered an Acknowledgement frame (see Figure 2).

6 Conclusion and Future Work

In this work we have presented a real-time assessment suite for IEEE 802.15.4 networks. This suite provides two services, namely network monitoring and fault injection.

Network monitoring allows the capture and analysis of network traffic under normal operating conditions and in the presence of errors. Each captured frame is timestamped with the arrival instant thus supporting the analysis of the real-time characteristics of the network and with an indication of correctness, thus supporting the analysis of the network error characteristics.

With this suite effective fault injection campaigns can be easily defined and instantiated while providing means to visualize and record its effects. Such fault injection campaigns are relevant, to test and validate models describing the operation of network related protocols and strategies to verify and/or improve it.

In particular, both services provided by the suite are useful to study and validate innovative research approaches aiming to bring hard real-time guarantees to WSANs. This has been one of the main motivations for designing and developing this tool.

Future work will include the sophistication of the hardware integration interface. The sophistication of the fault injection controller will allow more and better ways to specify a fault injection campaign. Finally, future work will also include improvements in the tools reading the statistics file, processing it, and outputting fault injection parameters that, when inserted in the fault injection tool, will emulate a scenario approximated to the one represented in the statistics file.

References

[1] *AT86RF232 - Low Power, Transceiver for Zigbee, IEEE 802.15.4, 6LoWPAN, RF4CE and ISM Applications*, 2011.

[2] IEEE standard for local and metropolitan area networks–part 15.4: Low-rate wireless personal area networks (LR-WPANs). *IEEE Std 802.15.4-2011 (Revision of IEEE Std 802.15.4-2006)*, pages 1–314, Sept 2011.

[3] *REB232ED-EK - Low Power, Evaluation Kit for Zigbee, IEEE 802.15.4, 6LoWPAN, RF4CE and ISM Applications*, 2011.

[4] P. Bartolomeu and J. Fonseca. An assessment of the IEEE 802.15.4 PHY immunity to WiFi interference. In *Emerging Technologies and Factory Automation (ETFA), 2010*, Sept 2010.

[5] C.A. Boano, Zhitao He, Yafei Li, T. Voigt, M. Zuniga, and A. Willig. Controllable radio interference for experimental and testing purposes in wireless sensor networks. In *Local Computer Networks, 2009.*, Oct 2009.

[6] C.A. Boano, T. Voigt, C. Noda, K. Romer, and M. Zúñiga. JamLab: Augmenting sensornet testbeds with realistic and controlled interference generation. In *10th Int. Conference on Information Processing in Sensor Networks (IPSN)*, pages 175–186, April 2011.

[7] Gerald Combs. The Wireshark network protocol analyzer. Available at: `http://www.wireshark.org/`. Accessed in June 5, 2013.

[8] D. Eckhardt and P. Steenkiste. Measurement and analysis of the error characteristics of an in-building wireless network. In *Annual Conference of Special Interest Group on Data Communication (SIGCOMM)*, 1996.

[9] T. Fujiwara, T. Kasami, A. Kitai, and S. Lin. On the undetected error probability for shortened hamming codes. *IEEE Transactions on Communications*, 33(6), June 1985.

[10] I Howitt and J.A Gutierrez. IEEE 802.15.4 low rate - wireless personal area network coexistence issues. In *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, volume 3, pages 1481–1486 vol.3, March 2003.

[11] Van Jacobson and S McCanne. libpcap: Packet capture library. *Lawrence Berkeley Laboratory, Berkeley, CA*, 2009.

[12] A Koubaa, S. Chaudhry, O. Gaddour, R. Chaari, N. Al-Elaiwi, H. Al-Soli, and H. Boujelben. Z-monitor: Monitoring and analyzing IEEE 802.15.4-based wireless sensor networks. In *IEEE 36th Conf. on Local Computer Networks (LCN)*, pages 939–947, Oct 2011.

[13] C.P. O'Flynn. Message denial and alteration on IEEE 802.15.4 low-power radio networks. In *New Technologies, Mobility and Security (NTMS), 2011 4th IFIP International Conference on*, pages 1–5, Feb. 2011.

[14] F. Österlind, A. Dunkels, T. Voigt, N. Tsiftes, J. Eriksson, and N. Finne. Sensornet checkpointing: Enabling repeatability in testbeds and realism in simulations. In Utz Roedig and CormacJ. Sreenan, editors, *Wireless Sensor Networks*, volume 5432 of *Lecture Notes in Computer Science*, pages 343–357. Springer Berlin Heidelberg, 2009.

[15] W.-B. Pöttner and L. Wolf. IEEE 802.15. 4 packet analysis with wireshark and off-the-shelf hardware. In *Proceedings of the Seventh International Conference on Networked Sensing Systems (INSS2010). Kassel, Germany*. Citeseer, 2010.

[16] R. Sasnauskas, O. Landsiedel, M. H. Alizai, C. Weise, S. Kowalewski, and K. Wehrle. KleeNet: Discovering Insidious Interaction Bugs in Wireless Sensor Networks Before Deployment. In *Proceedings of the 9th ACM/IEEE Int. Conference on Information Processing in Sensor Networks*, IPSN '10, pages 186–196, New York, NY, USA, 2010. ACM.

[17] J. L. R. Souza and J. Rufino. Characterization of inaccessibility in wireless networks - a case study on IEEE 802.15.4 standard. In *Analysis, Architectures and Modelling of Embedded Systems. Proceedings of the Third IFIP TC 10 International Embedded Systems Symposium (IESS 2009), Langenargen, Germany, September, 2009.*, September 2009.

[18] J. L. R. Souza and J. Rufino. Analysing and reducing network inaccessibility in IEEE 802.15.4 wireless communications. In *38th IEEE Conference on Local Computer Networks (LCN)*, Sydney, Australia, October 2013.

[19] Spartan-3E FPGA family data sheet, August 2009.

[20] W. Xu, Ke Ma, W. Trappe, and Y. Zhang. Jamming sensor networks: attack and defense strategies. *Network, IEEE*, 20(3):41–47, 2006.