Methods and Tools for Assessment of Wireless Networks in Extreme Environments

Rui P. Caldeira, **Jeferson L. R. Souza**, **Ricardo C. Pinto**, and **José Rufino**
LaSIGE, Faculdade de Ciências, Universidade de Lisboa, Lisboa, Portugal
Email(s): {rcaldeira, jsouza}@lasige.di.fc.ul.pt, {ricardo.pinto, jmrufino}@ciencias.ulisboa.pt

***Abstract*—Wireless sensor and actuator networks (WSANs) are now ubiquitous being used in a continuously growing number of application settings. Many of these applications, such as those commonly found in avionics and aerospace, have the ability to host, in the same computing platform, applications with different levels of criticality (or importance), i.e. mixed-critical applications, where real-time and dependability requirements may be a must. WSANs are thus paving their way in these extreme environments and application domains.**

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 an innovative solution for one-hop assessment of WSANs in the presence of errors. The solution includes devices and comprehensive methods to monitor the real-time behaviour of the network, to emulate accidental errors and to perform intentional attacks. This allows to study the error, timing and performance characteristics of WSANs, thus contributing to an accurate characterization of the behaviour of network-based protocols, in several application domains, including extreme environments, opening room for system verification/validation, as well as qualification and certification.

A prototype of the assessment suite based on the IEEE 802.15.4 standard is presented, along with a set of simple, yet representative, use cases.

***Index Terms*—network and protocol assessment; network monitoring; fault injection; real-time; dependability; timeliness; wireless communications; wireless sensor and actuator networks**

I. Introduction and Motivation

Wireless Sensor and Actuator Networks (WSANs) 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 application settings. 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 and errors can endanger the real-time guarantees of the communications as well as of the overall system.

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).

This paper discuses the methods to be used in the design of a highly flexible advanced open tool that allows the real-time assessment of wireless 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.

Network monitoring is a technique aimed at passively analysing the interactions between network nodes. Traditionally, errors are treated as omissions and are not reported by analysis tools. Since the disturbances introduced by error patterns cannot be ignored in critical applications and/or in extreme environments, network monitoring methods able to capture the occurrence of errors are needed.

Once the natural occurrence of errors may be rare in some settings, stress network operation through injection of interfering patterns in the wireless transmission medium may be of utmost importance. On one hand, allows emulating conditions that can be found in reality, namely in extreme environments. This is very helpful since allows to test a network at will in a controlled environment, observe its behaviour, and stress the operation of network related protocols. Another aim is focused in controllably disturb a network, intentionally, to study the susceptibility of network protocols either to accidental faults or to malicious attacks. With this purpose in mind, the combination of fault injection with network monitoring allows to trigger a specific set of fault injection actions upon the occurrence of particular network traffic patterns.

The paper is structured as follows. Section II reviews the related work. Section III discusses the methods needed for network assessment. Section IV describes the assessment suite and its components. Section V will walk through some use cases and section VI presents the conclusions and future work.

II. Related Work

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

Specific monitoring tools and use cases for WSANs, are reported in [1], [2]. 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.

Raising concerns on how insecure WSANs are, a wide range of attack injection strategies were defined in [3], taking into account the scarcity of power resources in wireless devices. A

dual-transceiver device capable of very precise attack injection techniques is described in [4], but the control software 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 [5] 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. Recognizing that robustness against interferences and packet losses is crucial to correct network functioning and that interferences compromise timing guarantees, this study focuses in disturbing physical (PHY) layer operation by recreating 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 [6], is that there are no simple methods of mapping such interferences into packet/frame losses. Our methods and tools address this limitation by focusing on a packet/frame level approach.

III. Methods For Network Operation Assessment in Extreme Environments

The methods required to assess network operation have their roots in network traffic monitoring, enabling the evaluation of network timing and error characteristics. Network traffic monitoring can be combined in runtime with fault injection, which allows: emulation of specific error patterns and environments; stress the operation of network-based protocols with verification/validation purposes.

A. Error-aware network traffic monitoring

All network monitors provide a very basic function of capturing network traffic, restricted to correctly formatted packets. Existing network monitors, to the best of our knowledge, do not capture and signal the occurrence of errors.

Wireless networks operating in extreme environments are particularly susceptible to heavy electromagnetic interference (EMI) and therefore to a high rate of error occurrences. To assess the error characteristics of wireless networks in extreme environments, advanced network monitoring methods must capture both correct and erroneous frames, thus allowing:

- report of network-based protocol operation including its timing characteristics, provided that frames/packets are tagged with a timestamp of their arrival instant;
- capture of the environment/network error characteristics and analysis of its impact on the operation of network-based protocols, provided that frame/packet errors can be detected and identified.

Frame error detection is achieved through a *Frame Check Sequence* (FCS) field included in the general format of a frame. Other fields include the frame header and the data payload, which may have a zero length. Any deviation from the original frame content, e.g., resulting from a corruption in the wireless transmission medium, can be detected through FCS checking mechanisms. The residual probability of undetected errors is negligible [7].

Given the availability of Commercial-Of-The-Shelf (COTS) network interfaces, these may be exploited with the purpose of capturing all traffic transmitted in a specific wireless radio channel. The method for achieving error-aware and time-aware network monitoring includes:

- configuration of the monitoring network interface hardware (a.k.a. sniffer), in the so called *promiscuous mode*, allowing a passive listener behaviour;
- disable of *integrity check procedures*, thus allowing both correct and erroneous frames to be equally reported;
- *extension of the FCS integrity check mechanism* [8], enabling a correct/incorrect integrity indication to be attached to all monitored frames.

B. Runtime traffic analysis

Every monitored frame/packet needs to be analysed with the following purposes:

- detection of an event (e.g., monitoring of a specific frame/packet) triggering traffic capture;
- filtering of the network traffic to be captured;
- detection of specific events triggering and/or controlling a fault-injection campaign.

The contents of both header and payload fields of a frame/packet can be analysed, searching for specific patterns.

C. Fault injection

In the context of a wireless communication environment, any fault injection method aims at interfering with ongoing transmissions in the wireless transmission medium. Several options do exist for the injection of interfering patterns:

- Pure noise patterns, thus emulating plain PHY level interference;
- Selected data patterns, always preceded by the standard preamble[1];
- Selected data patterns encapsulated in a correctly formatted Medium Access Control (MAC) level frame, thus injecting a standard compliant transmission.

The issuing of these interfering patterns implies some sort of data transmission to the wireless transmission medium. To fully control fault injection timings, a direct (constant delay) access to the wireless transmission medium is required. This way, the variable medium access delays that characterize most standard MAC level protocols are avoided. Should the use of COTS wireless network interfaces be exploited, the standard MAC layer protocol needs to be bypassed, selecting a direct access to the wireless transmission medium.

Fault injection methods can then properly control the patterns, symbolic/semantic contents and timings of all the interfering transmissions, in the course of a fault injection campaign. This is achieved through the specification of a set of fault injection parameters – the fault injection profile.

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

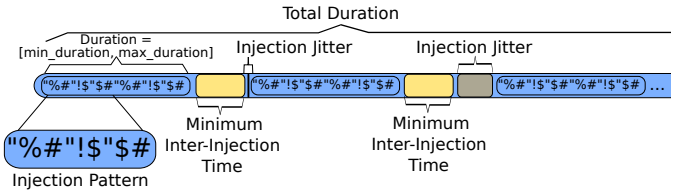

Fig. 1: Representation of some fault injection parameters

1) Fault injection parameters: the definition of a fault injection campaign requires the specification of some parameters in order to build a particular fault injection profile. Some of the parameters are illustrated in Figure 1 and explained further:

- **Injection Mode -** the injection mode parameter determines how the potentially interfering data shall be sent to the wireless transmission medium. Firstly, if the pure noise mode is selected, a given sequence of symbols – the *Injection Pattern* – is simply sent to the wireless transmission medium until the injection event duration is reached. Secondly, in the preamble preceded mode, the injection pattern data is attached to the standard preamble and sent to the wireless transmission medium. Finally, in the encapsulated frame mode the injection pattern includes all the fields of a correctly formatted frame, i.e. the frame header, the (optional) data payload and the FCS trailer. Both good and bad FCS values are allowed;
- **Injection Pattern -** (see Figure 1) the data 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 (pure 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, 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 will always have the same value. Otherwise, it will have a random value between the minimum and maximum durations dictated by a given statistical distribution;
- **Number of Events -** number of total fault injection events in the fault injection campaign, after which the campaign shall terminate;
- **Minimum Inter-Injection Time -** the minimum time interval between consecutive fault injection events, as illustrated in Figure 1;
- **Maximum Injection Jitter -** maximum value of a positive random time, dictated by a given statistical distribution and referred as *Injection Jitter* in Figure 1, to be added to the Minimum Inter-Injection time;
- **Total Duration -** (see Figure 1) the total duration of a fault injection campaign. If the number of events is specified, this parameter is meaningless, since it is confined to a lower as well as to 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 will last until the specified duration is reached. The *Total Duration* is an alternative to the *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 a stop command is explicitly issued.

Other parameters, strictly related with PHY layer operation, such as the transmission power and the radio channel frequency, complete the definition of the fault injection profile.

2) Fault injection profiles - reference use cases: to illustrate the flexibility, pertinence and usefulness of the fault injection profile approach specified in section III-C1, a relevant set of well-known fault injection profiles, extensively described in the literature [3], [4], is summarized in Table I.

The *constant profile* continuously injects a regular noise pattern and is meant for the injection of jamming attacks [3]; the *random profile* injects randomly generated noise, being suited for the casual corruption of frame transmissions [3]; the *adaptive* and *frame-type adaptive* profiles are aimed to synchronize the fault-injection timing with the network traffic, as specified by the methods described in section III-B, being the latter a specialisation of the former. For example, the last profile is specially useful to corrupt beacon frames, thus causing one-hop wide communication blackouts [9].

In Table I, both adaptive profiles were parametrised in compliance with the IEEE 802.15.4 standard [10]; different fault injection event durations may be required for compliance with other network standards.

D. Post-capture traffic analysis

The following methods are envisaged for post-capture traffic processing actions:

- conversion from a native frame format representation into *de-facto* standard representations, such as PCAP [11] and Comma-Separated Values (CSV);
- presentation of the captured traffic to the user, possibly through an easy to use Graphical User Interface (GUI), such as the one available in Wireshark [12];
- storage of the captured traffic in statistics files, using the *de-facto* standard representations;
- tools for processing of the statistical files, aiming traffic timing/error analysis and characterization.

Fault Injection Profile	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 (IEEE 802.15.4)	133 bytes (IEEE 802.15.4)	-	1	-	0	Cyclical
Frame-Type Adaptive	Noise	19 bytes (IEEE 802.15.4)	133 bytes (IEEE 802.15.4)	-	1	-	0	Cyclical

TABLE I: Reference fault injection profiles expressed in function of fault injection parameters

IV. A SUITE FOR REAL-TIME ASSESSMENT OF WIRELESS NETWORKS

This section addresses the generic architecture of a tool (Figure 2), following the methods specified in section III. The resulting prototype targeting IEEE 802.15.4 networks is mentioned, when relevant. A thorough description of the IEEE 802.15.4 tool prototype is advanced in [13].

A. Network Monitoring Unit

The Network Monitoring Unit aims to support the error-aware network traffic monitoring methods of section III-A. In essence, this unit is composed by a hardware network interface, which includes the radio technology (PHY layer) and, sometimes, some residual MAC layer functions, configured for promiscuous listening with integrity check disabled.

Existing COTS devices can provide an effective support to error-aware network traffic monitoring: the prototyping of the IEEE 802.15.4 tool, uses the Atmel AT86RF232 device [14].

To enable evaluation of precise timing characteristics, a timestamp is attached upon the arrival of each frame, including erroneous ones, as represented by interaction 5 of Figure 2.

B. 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 (Figure 2), which holds one or more fault injection profiles. This allows a precise timing in the issuing of wireless medium interfering patterns (interaction 2 of Figure 2). Timestamping of a fault injection campaign begin and end events is performed upon its signalling by interaction B (Figure 2).

The fault injector device is a perfectly common network interface, being configured to bypass any MAC layer function, thus allowing a direct access to the PHY layer and to the wireless transmission medium. This device can be effectively supported on COTS technology: the prototyping of the IEEE 802.15.4 tool, uses the Atmel AT86RF232 device [14].

C. Hardware Integration Unit

The Hardware Integration Unit enables the interaction and cooperation between the network monitoring unit and the fault injection unit, as shown in Figure 2. The software components of both units may be hosted at processor cores, instantiated in a single Field Programmable Gate Array (FPGA) [15]. The prototyping of the IEEE 802.15.4 tool, uses a Atmel microcontroler for each unit [16].

Specified in Figure 2: interaction A conveys the triggers for fault injection events/campaigns; the occurrence of injection events is signalled on interaction B; interactions 1-2-3 are mainly intended for initialisation purposes; interaction 4 conveys the monitored traffic.

The runtime traffic analysis of section III-B, optimally resorts to special-purpose hardware, such as FPGA-based instantiation of Content Addressable Memories (CAMs) [17], allowing extremely low latencies in the traffic analysis/fault injection loop. Non-optimal approaches resort to software implementations (e.g., prototyping of the IEEE 802.15.4 tool).

D. Integration with Wireshark and other tools

This section addresses how the post-capture traffic analysis of section III-D are supported. Wireshark communicates directly with the units for initialisation and command purposes: interaction 1, and indirectly, interactions 2-3 of Figure 2.

After initialization, the network monitoring unit delivers captured frames, to the Wireshark Translator (interaction 5) first, then to the Wireshark tool (interaction 6).

The Wireshark Translator is a piece of software that translates the raw frames collected from the network monitoring unit into the *de-facto* standard PCAP format [11], used by Wireshark. The Wireshark Translator also creates a CSV format statistics file, interaction 7 of Figure 2, which is recognized by most statistical data application and is useful in providing a base for additional processing that may not be achieved using Wireshark. The Wireshark internal filtering and data processing facilities are still available [12].

V. USE CASES

This section discusses some simple, yet representative use cases of wireless network evaluation.

A. Standalone Network Monitoring

The first use case, illustrates the functionality of our tool in the traffic monitoring of a IEEE 802.15.4 network, operating in a ”dirty environment” where an heavy electromagnetic interference is expected from ”alien” IEEE 802.11 nodes. The IEEE 802.15.4 nodes operate in 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 likelihood of interference is high [18]. The IEEE 802.15.4 network operates

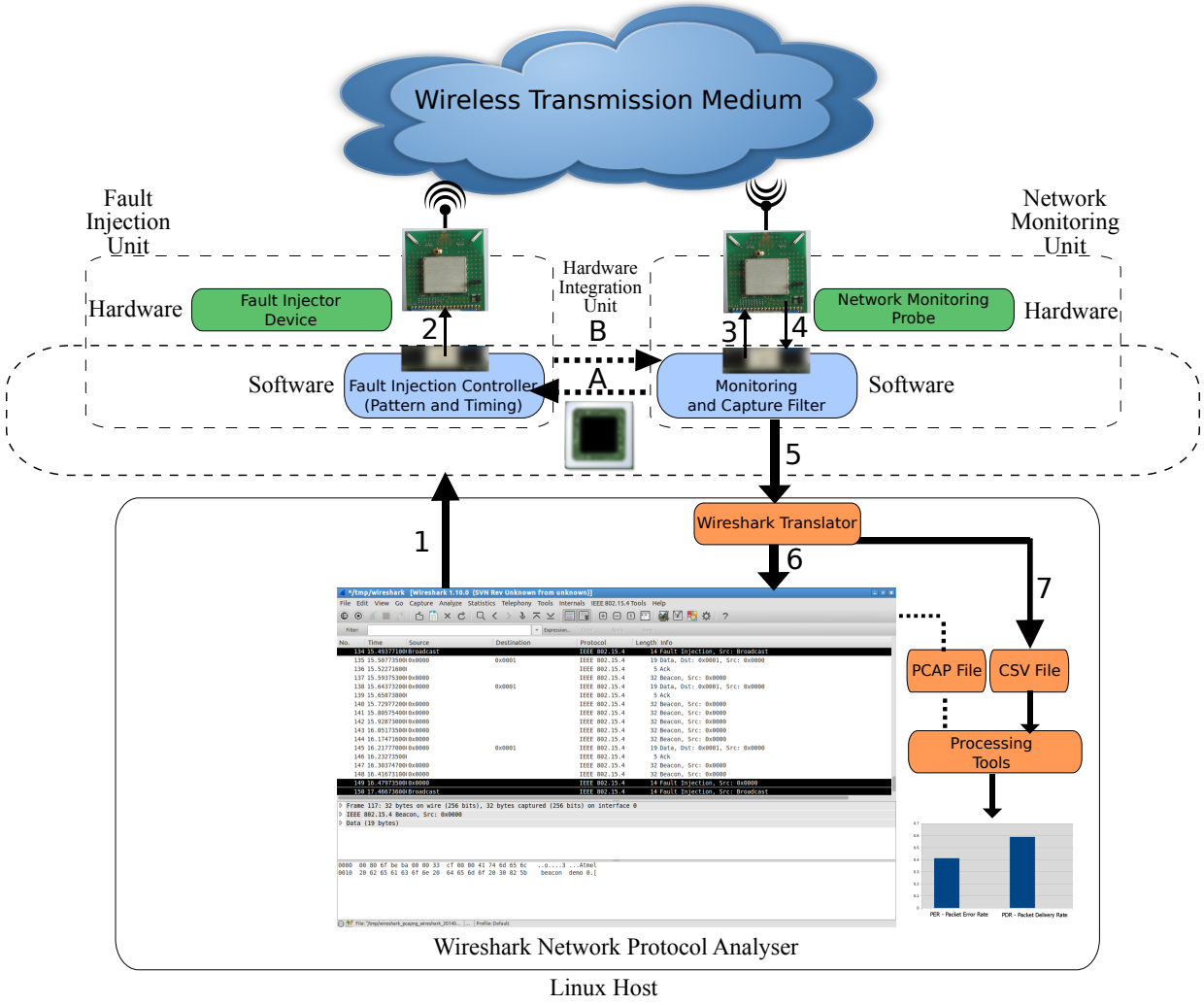

Fig. 2: Tool Architecture Design

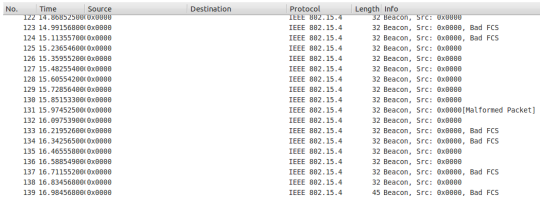

Fig. 3: Standalone network monitoring

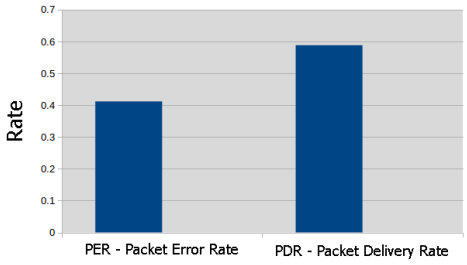

Fig. 4: IEEE 802.15.4 packet rates monitored in a ”highly dirty environment”

under very light-weighted load conditions, with the network coordinator transmitting mostly beacon frames.

The results inscribed in Figure 3, 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 [10]. The observed packet rates, automatically extracted from the statistics file by post-capture processing tools, is represented in the chart of Figure 4.

B. Network monitoring with constant jamming

This use case injects faults in the wireless medium using the reference *constant profile* defined in Table I. The results are shown in Figure 5: the black shaded highlighted frames are special events, signalling to Wireshark users when a fault injection campaign begins and when it ends; 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.

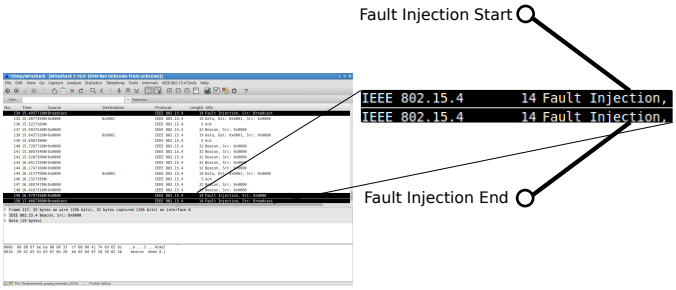

Fig. 5: Making use of the *constant* fault injection profile

C. Network monitoring with parametrised fault injection

This experiment illustrates the use of parametrised fault injection: the same IEEE 802.15.4 network operates in a "clean environment"; only occasional frame errors are expected. To stress the operation of network-based protocols, a fault injection campaign was conducted, with the following parameters:

- **Injection Mode:** Preamble Preceded
- **Injection Pattern:** AABBCC
- **Minimum Duration:** 28000 μs
- **Maximum Duration:** 28000 μs
- **Number of Events:** 100
- **Minimum Inter-Injection Time:** 1000000 μs
- **Maximum Injection Jitter:** 0 μs
- **Total Duration:** not defined
- **Trigger:** Cyclic
- **Transmission Power:** 3 dBm
- **Radio Channel Frequency:** 2.435 Ghz (channel 17)

The pattern *AABBCC* preceded by the IEEE 802.15.4 preamble is transmitted during 28000 μs. After a wait of 1000000 μs, the sequence is repeated 99 times. The results of this campaign are shown in Figure 6: the beacon frame transmitted by the IEEE 802.15.4 network coordinator and signalled by the red arrow at the right has been corrupted by a fault injection event; other captured frames and fault injection events have been deliberated suppressed from the screen, using the Wireshark own filtering facilities; the AABBCC frame pattern when analysed by the Wireshark dissector [12] is labelled as an IEEE 802.15.4 acknowledgement frame.

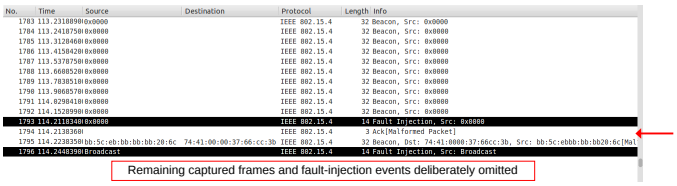

Fig. 6: Exploiting parametrised fault injection

VI. Conclusion and Future Work

This work presented the methods required for real-time assessment of networks and protocols and how they can be integrated in a easy to use tool.

Network monitoring allows the capture and analysis of network traffic under normal operating conditions and in the presence of errors. Each captured frame (correct or erroneous) is timestamped with the arrival instant thus supporting the analysis of the timing characteristics of the network.

Supporting effective fault injection campaigns, the tool enables both the emulation of real error patterns and the stress test of network-related protocols, such as [19], opening room for their verification/validation.

Future work includes improvements in design and engineering of the tool along with its extension to IEEE 802.11p networks, targeting vehicular applications.

References

[1] W.-B. Pöttner and L. Wolf, "IEEE 802.15. 4 packet analysis with Wireshark and off-the-shelf hardware," in *Proceedings of the Seventh Int. Conference on Networked Sensing Systems (INSS2010)*, 2010.

[2] 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)*, Oct 2011, pp. 939–947.

[3] W. Xu, K. Ma, W. Trappe, and Y. Zhang, "Jamming sensor networks: attack and defense strategies," *IEEE Network*, vol. 20, no. 3, pp. 41–47, 2006.

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

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

[6] 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.

[7] T. Fujiwara, T. Kasami, A. Kitai, and S. Lin, "On the undetected error probability for shortened hamming codes," *IEEE Transactions on Communications*, vol. 33, no. 6, Jun. 1985.

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

[9] ——, "Characterization of inaccessibility in wireless networks - a case study on IEEE 802.15.4 standard," in *Proceedings of the Third IFIP TC 10 International Embedded Systems Symposium (IESS 2009)*, Langenargen, Germany, Sep. 2009.

[10] *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)*, IEEE, Sep. 2011.

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

[12] G. Combs, "The Wireshark network protocol analyzer," Available at: http://www.wireshark.org/, accessed in June 5, 2013.

[13] R. P. Caldeira, J. L. R. Souza, R. C. Pinto, and J. Rufino, "A tool for real-time assessment of IEEE 802.15.4 networks through fault injection," in *Atas do 7º Simpósio de Informática (INForum 2015)*, Covilhã, Portugal, Sep. 2015, (accepted for publication).

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

[15] "Spartan-3E FPGA family data sheet," Xilinx Inc., Aug. 2009.

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

[17] K. Locke, *Parameterizable Content-Addressable Memory - Application Note XAPP1151*, Xilinx, Mar. 2011.

[18] I. Howitt and J. Gutierrez, "IEEE 802.15.4 low rate - wireless personal area network coexistence issues," in *Proceedings of Wireless Communications and Networking (WCNC)*. IEEE, Mar. 2003, pp. 1481–1486.

[19] J. L. R. Souza, R. C. Pinto, and J. Rufino, "Mechanisms to enforce dependability and timeliness in wireless communications," in *Proceedings of 2nd Int. IEEE Conf. on Wireless for Space Applications and Extreme Environments (WiSEE 2014)*, Noordwijk, The Netherlands, Jul. 2014.