A Fault-Injection Prototype for Safety Assessment of V2X Communication

Daniel Skarin, Benjamin Vedder, Rolf Johansson, and Henrik Eriksson
Department of Electronics
SP Technical Research Institute of Sweden
Borås, Sweden
e-mail: {daniel.skarin, benjamin.vedder, rolf.johansson, henrik.eriksson}@sp.se

***Abstract*— This paper describes an approach for injecting faults in ad hoc vehicle networks. A prototype fault injector, which makes it possible to investigate how a cooperative vehicle system behaves in the presence of communication errors, has been developed. The prototype shows a feasible way to use fault injection as technique to produce evidence for a safety case belonging to a cooperative automotive system.**

Keywords-fault injection, safety assessment, IEEE 802.15.4, V2X communication.

I. INTRODUCTION

In the past years, there has been a strong focus on functional safety in the automotive domain. In 2011, the standard ISO 26262 [1] was released, and currently the industry is adopting the development procedure to the standard. At the same time, automotive functions are getting more and more complex; autonomous and cooperative vehicles will soon move from prototypes to products. Safety assessment of cooperative systems will put requirements on evidence which show that communication failures are handled in a safe way. This paper shows a way to inject communication faults in cooperative systems as a technique to produce evidence for a safety case.

Cooperative vehicle systems cover a wide range of interdependence. Willke *et al.* [2] have suggested a taxonomy defining four type levels. On type levels 1 and 2, vehicles and infrastructure are exchanging information without being dependent on it to achieve a safe behavior. On type level 3, the functions rely on communicated information from other vehicles about motion and actuator states to ensure safe and/or efficient operation. On type level 4, applications use inter-vehicle communication to reach a common goal, e.g. driving in a road train (platooning). At least on the type levels 3 and 4, safety requirements will be allocated on the communication between the vehicles (V2V) and between the cars and the infrastructure (V2I).

According to the ISO 26262 standard, safety requirements shall be refined from top-level safety goals to the system components of the physical architecture. For safety-related cooperative functions, this implies that some safety requirements will be put on the V2V and V2I communication, respectively. Furthermore, the standard states what is needed to argue in order to fulfil verification of the safety requirements. For the higher integrity levels (ASIL C and D), it is required to use fault-injection techniques to show that safety mechanisms can handle all safety-relevant faults.

Fault injection in wireless communication used for transfer of safety-critical information in ad hoc vehicle networks needs further research. For computer systems (hardware and software) communicating via wires, there is a fairly long tradition of using fault-injection techniques and tools [3]. Alena *et al.* [4] have investigated how the fault tolerance of wireless sensor networks using IEEE 802.15.4 is affected by interference from other networks and multi-paths. Boano *et al.* [5] present a solution which produce repeatable and precise patterns of interference in wireless sensor networks. Malicious faults (attacks) and some natural faults in ad hoc networks can be assessed using the fault-injection platform developed by de Andrés *et al*. [6].

In this paper, a fault-injection prototype is described. The prototype is based on IEEE 802.15.4 since this standard is used for communication in the automotive and aerospace demonstrators of the KARYON project [7]. However, it is straightforward to adapt the concept to other techniques to be used in the automotive domain (IEEE 802.11p).

Section II introduces relevant fault models originating from functional safety standards. The section also explains how different failure modes can be emulated. Section III describes the fault injection prototype, and Section IV presents initial conclusions and future work.

II. FAULT INJECTION IN COMMUNICATION

A. Fault models

Standards for functional safety, such as ISO 26262 for road vehicles and the generic IEC 61508, list failure modes which are applicable for communication. Part 5 of ISO 26262 [1] lists failure modes for on-chip communication and data transmission. The failure modes for data transmission are applicable for wireless communication. IEC 61508-2 [8] lists identical failure modes for communication. Other important failure modes for communication are blocking access to communication channel [9] and asymmetric information [10]. Table 1 summarizes failure modes applicable for wireless communication.

Based on the diagnostic coverage that is claimed for a safety mechanism, ISO 26262-5 Table D.1 [1] lists failure modes that need to be analyzed. Failure modes for on-chip communication are described next.

Stuck-at failures are described as a continuous low or high signal at the pins of an element. They are applicable for elements which have a pin-level interface for data, control, address, and arbitration signals.

 ISBN: 978-1-61208-378-0

TABLE I. FAILURE MODES FOR COMMUNICATION

Failure mode	Interpretation
Message Corruption	The received data of a message is incorrect.
Message delay	A message is received later than expected by all, or some, receivers.
Message loss	A message is lost by all, or some, receivers.
Unintended message repetition	Receivers obtain two or more messages with the same information instead of one message.
Resequencing	Messages are received with incorrect sequence numbering.
Insertion of message	Receivers obtain a message that they did not expect
Masquerading (or incorrect addressing)	A sender transmit messages using an id of a different sender
Asymmetric information	Information from a single sender is received differently by receivers. It can also be that information from a sender is only received by a subset of the receivers
Blocking access to a communication channel	Prevents nodes from accessing the communication channel, similar to a babbling idiot.

The *direct current* fault model extends stuck-at failures with stuck-open, open, or high impedance outputs, and short circuits between signal lines. The analysis of the fault model is applicable for data, control, address and arbitration signals, but is mainly intended for main signals or on highly coupled interconnections.

When several devices are connected to a bus, arbitration is used to determine which device that controls the bus. *No arbitration* and *continuous arbitration* are mentioned as failure modes for on-chip communication in ISO 26262-5 [1]. *Time out* is mentioned in both IEC 61508 and ISO 26262, but neither standard describes the failure mode in more detail.

Soft errors are caused by ionizing particles, supply voltage noise, or cross-coupling between signal lines. The consequence is one or several bit-flips in memories or bus signals.

B. Emulating the Effects of Faults

The failure modes for wireless data communication can be emulated using a combination of jamming, packet injection, and packet sniffing. Jamming [5][11] is used to prevent one or several nodes from receiving or sending packets. Packet injection is used to insert additional, duplicated or corrupted messages in the wireless network. Packet sniffing allows the fault injection module to eavesdrop the wireless traffic in a non-intrusive manner. This is useful for logging and for triggering the injection of different failure modes.

Table 2 shows how different failure modes can be implemented by combining jamming and packet injection. For example, the effects of a message delay can be emulated by jamming to prevent nodes from receiving the original message, and then resending the original message with a delay. This assumes that we have a priori knowledge of the content of the message. Message losses are emulated by activating jamming when specific messages are being transmitted by a node.

Figure 1 and Figure 2 illustrate how failure modes for on-chip communication are emulated. The signal between two elements passes through a fault injection module which has the capability to modify the transmitted signal value. For most failure modes, such as soft errors, a faulty signal only relies on the value of the non-faulty signal as shown in Figure 1. For short-circuits between signals, however, the values of two or more signals are needed, as shown in Figure 2.

TABLE II. EMULATING FAILURE MODES USING JAMMING AND PACKET INJECTION

Failure mode	Jamming	Packet Injection
Message Corruption	x	x
Message delay	x	x
Message loss	x	
Unintended message repetition		x
Resequencing		x
Insertion of message		x
Masquerading (or incorrect addressing)		x
Asymmetric information	x	x
Blocking access to a communication channel	x	

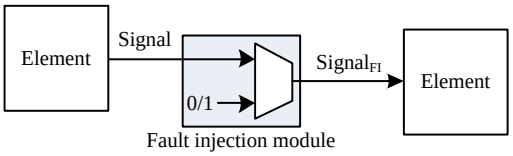

Figure 1. Injection of stuck-at faults in a signal.

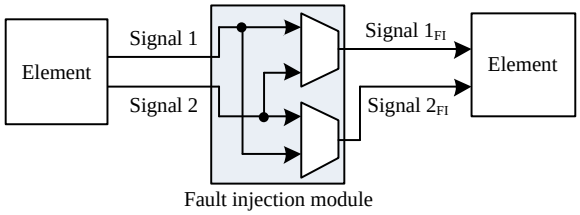

Figure 2. Injection of short-circuit failures between two signals.

C. Controlling When to Inject Faults

Figure 1 shows a state machine for controlling the fault injection. The idle state has an internal counter to keep track of the currently evaluated trigger. When all triggers have been evaluated to true in the correct order, fault injection is activated in the state “Start FI”. Following that, the “FI” state is immediately entered.

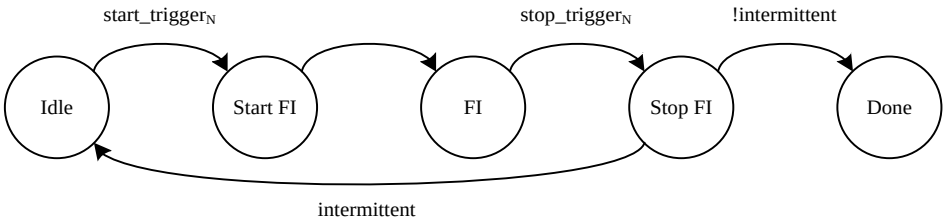

Figure 3. State machine to control the fault injection.

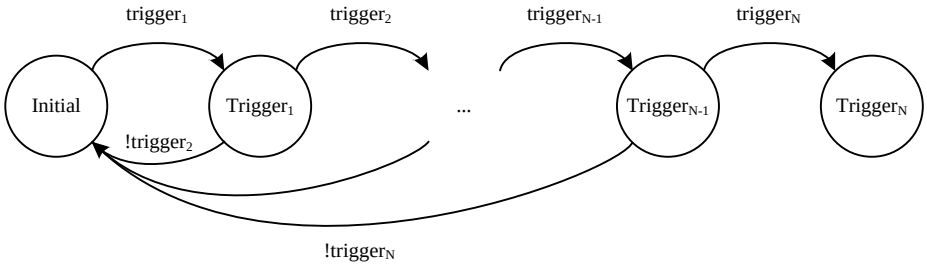

Figure 4. State machine to handle start and stop triggers for fault injection.

The "FI" state is exited when all stop triggers have been fulfilled. Unless an intermittent fault is emulated, the fault injection is stopped. For intermittent faults, there is a return to the idle state and another wait for start trigger fulfillment. Fault injection is activated using triggers which can be based on: elapsed time, probability per received packet, sender or receiver address of a packet, or data in the payload of a packet. Several triggers can be combined so that fault injection is started or stopped by a chain of events, as shown in Figure 2. Using this approach, well-known packet loss models such as Bernoulli and Gilbert-Elliot [12] can be supported, as well as simple triggers based on, e.g., elapsed time.

III. FAULT INJECTION PROTOTYPE

The fault injection concept described in the previous section has been implemented for vehicle demonstrators in the KARYON project [7]. The fault injection prototype can be used for injecting failures in IEEE 802.15.4 data communication, and in the on-chip communication. Figure 5 shows a picture of the fault injection node, which uses the STM32F4 microcontroller from ST and the CC2520 communication chip from Texas Instruments. The node is based on layout and hardware schematics which are freely available from [13].

The fault injector uses ChibiOS/RT [14] as its operating system, and implements the state machine described in Section II.C. The following fault injection triggers are supported:

- Time – Enabled after a specified time has elapsed.
- Packet probability – Enabled with a specified probability for each received packet.
- Packet destination address – Enabled when a packet with a matching source address is received.
- Packet source address – Enabled when a packet with a matching destination address is received
- Packet data – Enabled when the specified data matches the received data
- The fault injector is configured using USB commands, or by sending configuration packets via IEEE 802.15.4.

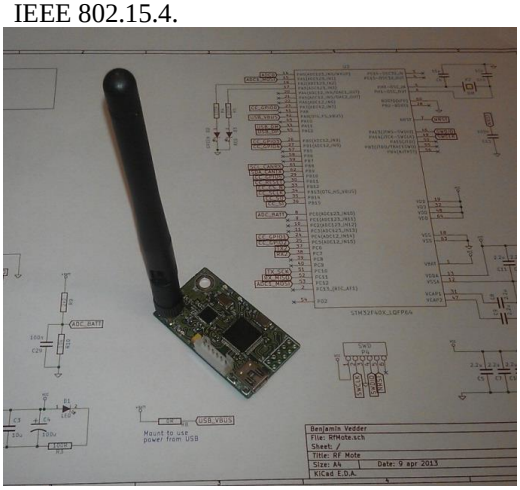

Figure 5. RF board with STM32F4 and CC2520 based on [Vedder].

The prototype fault injector also provides packet logging capabilities, which are useful for debugging purposes. The CC2520 communication chip provides hardware support for packet sniffing, which can be used as a non-intrusive method of observing wireless traffic. The fault injector can output captured packets in the packet capture (pcap) format using a named pipe. The logged traffic can then be analyzed in real-time using tools such as Wireshark which is an open source network protocol analyzer. Figure 6 shows an example of logged traffic in Wireshark.

The following failure modes are currently supported by the fault injection prototype: message corruption, delay, loss, insertion, unintended message repetition, masquerading, and blocking access.

 ISBN: 978-1-61208-378-0

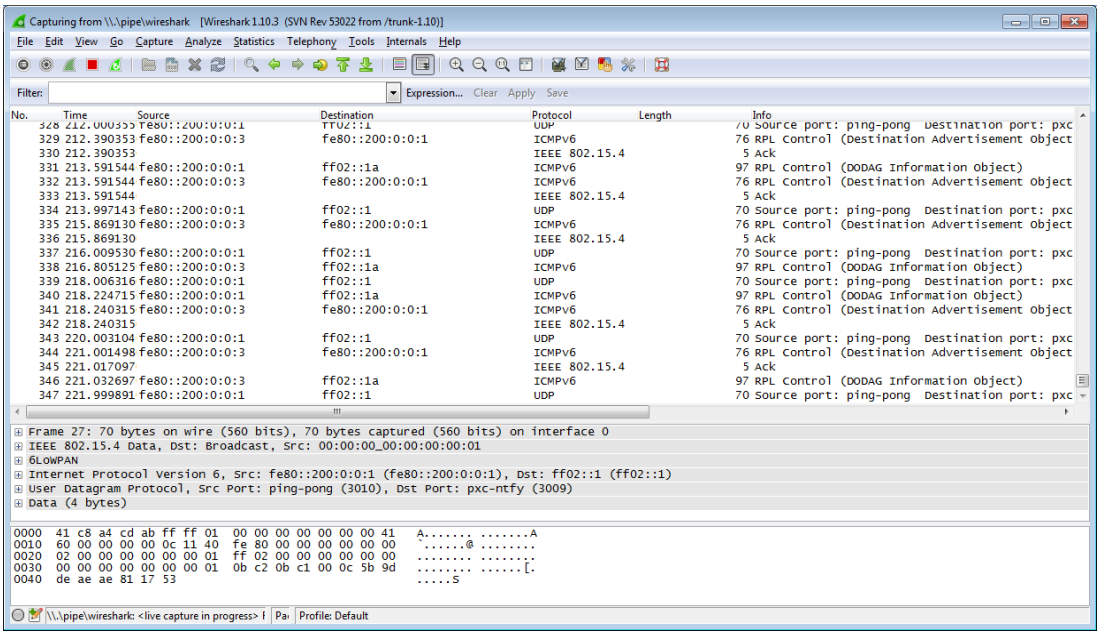

Figure 6. Packet sniffing using the fault injection node and Wireshark.

For some of the failure modes, e.g. message delay, message payload need to be known a priori. Proof-of-concept fault injections have been successfully performed, but no complete fault-injection campaigns have been run yet.

IV. CONCLUSIONS AND FUTURE WORK

A prototype fault injector for digital communication, in particular wireless communication, has been described. One limitation with the approach is that communication chips require some time to switch between receiving and sending. For the CC2520 chip, the RX/TX turnaround time is 192µs. For packets with a small payload, it might therefore not be possible to trigger the fault injection and jam the packet currently being received. This is something which will be investigated in the near future.

The prototype has been tested on IEEE 802.15.4 communication, but the concept is straightforward to adapt to other communication techniques, such as IEEE 802.11p.

The prototype shows that it is feasible to inject most faults needed in a safety assessment according to the requirements in functional safety standards.

ACKNOWLEDGMENT

This work has been supported by the EU under the FP7-ICT program, through project 288195 Kernel-based ARchitecture for safetY-critical cONtrol (KARYON).

REFERENCES

[1] ISO26262-5, “Road vehicles – Functional safety – Part 5: Product development at the hardware level”, 2011.

[2] T. L. Willke, P. Tientrakool, and N. F. Maxemchuk, “A survey of inter-vehicle communication protocols and their applications”, IEEE Communications Surveys & Tutorials, vol. 11(2) , pp. 3-20, 2009.

[3] H. Mei-Chen, T. K.Tsai, and R. K. Iyer, “Fault injection techniques and tools”, IEEE Computer, vol. 30(4), pp. 75-82, 1997.

[4] R. Alena, R. Gilstrap, J. Baldwin, T. Stone, and P. Wilson, “Fault tolerance in ZigBee wireless sensor networks”, Proc. 2011 IEEE Aerospace Conference, March 2011, pp. 1-15.

[5] C. A. Boano et al., “Controllable radio interference for experimental and testing purposes in wireless sensor networks,” Proc. IEEE 34th Conference on Local Computer Networks, Oct. 2009, pp. 865-872.

[6] D. de Andrés, J. Friginal, J.-C. Ruiz, and P. Gil, ”An attack injection approach to evaluate the robustness of ad hoc networks”, Proc. 15th IEEE Pacific Rim International Symposium on Dependable Computing, Nov. 2009, pp. 228-233.

[7] Homepage of Kernel-Based ARchitecture for safetY-critical cONtrol, http://www.karyon-project.eu/, accessed on 25th of June 2014.

[8] IEC 61508-2, “Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 2: Requirements for electrical/electronic/programmable electronic safety-related systems”, 2010.

[9] H. Kopetz, “Real-time systems”, Kluwer, 1997.

[10] F. Cristian, “Understanding fault-tolerant distributed systems”, Communications of the ACM, vol. 34, pp. 56-78, 1991.

[11] A. D. Wood, J. A. Stankovic, and G. Zhou, "DEEJAM: Defeating energy-efficient jamming in IEEE 802.15. 4-based wireless networks," Proc. 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON'07), June 2007, pp. 60-69.

[12] J.-P. Ebert and A. Willig, “A Gilbert-Elliot bit error model and the efficient use in packet level simulation”, Technical Report, TKN-99-002, Technical University of Berlin, 1999.

[13] Homepage of B. Vedder “CC2520 and STM32 RF boards”, http://vedder.se/2013/04/cc2520-and-stm32-rf-boards/, accessed on 25th of June 2014.

[14] Homepage of ChibiOS/RT, http://www.chibios.org, accessed on 25th of June 2014.

 ISBN: 978-1-61208-378-0