# Fault Injection Framework for Assessing Fault Containment of TTEthernet against Babbling Idiot Failures

Onwuchekwa Daniel, Obermaisser Roman
University of Siegen, Institute of Embedded Systems, Germany

***Abstract*—In safety critical communication systems, faulty nodes can monopolize a channel by transmitting untimely messages at random intervals and thus resulting in the failure of the system. This failure is known as a babbling idiot failure. This could be costly and catastrophic for safety critical systems, therefore these failures are avoided in time triggered communication systems by implementing fault tolerant functions such as local or central guardians. Research works evaluating the guardian functionality for real time networks such as Flexray and TTP have been carried out. This work evaluates the guardian functionality of the TTEthernet protocol. TTEthernet enforces a TDMA scheme for time triggered traffic and traffic policing for rate constrained traffic thereby protecting the network against babbling idiot failures. However these guardian functionality was not extensively evaluated. Dependability evaluation by fault injection on the entire TTEthernet communication system as a whole has not been extensively studied. In this paper we exploit a novel fault injection framework to generate babbling idiot failures for the purpose of verifying TTEthernet implementations with respect to fault containment. The framework adopts a novel cut-through approach abstracting the fault injector from both the end systems and switches, thereby facilitating portability and ease of use. This work introduces a fault injection framework to effectively verify babbling idiot fault tolerance of TTEthernet hardware implementations. In addition, it provides a means to evaluate the effect of various network faults on applications running on top the protocol. Test results carried out indicate the effect of babbling idiot messages on the latency and jitter of traffic over the TTEthernet network.**

***Index Terms*—Fault tolerance, Fault Injection, Fault containment**

## I. Introduction

Over the last few years there have been increasing interest and developments towards the extension of legacy Ethernet to support safety critical systems. Safety critical applications are increasingly being used in the military, aerospace, medical, automotive and railway industry. The communication requirements of providing real time and safety critical features over Ethernet is the basis for protocols such as TTEthernet and Time Sensitive Networking (TSN). The aforementioned protocols provide different levels of quality of service (QoS) and a synchronized global time service. They are designed to provide both fault tolerance and determinism on legacy Ethernet. The features added by these protocols to the legacy Ethernet extend its suitability for mixed critical applications, providing disparate quality of service for the traffic classes defined in the protocol. Inherently by design these networks use policing and channel redundancy to provide protection against babbling idiot failures (BIFs). TTEthernet is standardized as AS6802 [1] and the TSN [2]–[5] group of standards is currently in progress for standardization. Due to the attractive features provided by these protocols, it is expected that hardware implementations from several vendors would reach the market. Therefore the need for verification and validation at a physical level is essential to increase the confidence placed on a dependable product.

Reliability measurement relies on controlled fault injection experiments that are able to observe the behavior of a system under the effect of faults. Fault injection provides the platform for fault containment assessment, test of error handling and fault tolerance of a system, and the assessment of solutions to improve dependability. Injecting BIFs can be used to accelerates the evaluation of the fault containment of a network. A babbling idiot is characterized by the transmission of arbitrary messages at random points in time. A faulty node that monopolizes the common channel by sending messages at erroneous points in time is perceived to have a babbling idiot failure [6]. This failure mode has the potential to cause a complete system failure by disrupting communication between end systems operating correctly. A babbling idiot phenomenon impedes the fulfillment of real time and safety critical requirements by delaying or causing the loss of other messages. In safety critical systems BIFs could result to catastrophic consequences. BIFs are caused primarily by the node and can originate either from the end system hardware or software. A hardware babbling-idiot occurs when the failure is caused by the direct consequence of a hardware fault. A software babbling-idiot takes place when the fault originates from the application software (e.g a bug in the code or human factor such as a malicious attack). Time-triggered protocols could also take advantage of channel redundancy to overcome babbling-idiot failure. The deterministic transmission of messages on the replica is used to override the hardware babbling-idiot fault [7]. TTEthernet also uses the priori knowledge about the

978-1-5386-2542-2/18/$31.00 © 2018 IEEE

permitted temporal behavior to block untimely messages.

Considerable research works on babbling idiot protection for real-time networks were carried out in [6]–[8] using bus guardian functionality to tolerate BIF in FTDM, FlexCAN and earlier TDMA-based communication respectively. The evaluation of fault containment with respect to BIFs was also evaluated by fault injection in Flexray-based networks [9].

However, to the best of our knowledge, there have been no extensive academic research on injecting physical faults into TTEthernet network. There is no existence of an available and suitable framework for evaluating the protocol implementation.

In this work our focus is on evaluating the fault containment of TTEthernet against BIF. In addition, the fault injection framework was designed in a generic manner to be suitable for TSN. We designed a fault injection framework composed of a fault injector capable of injecting BIFs to emulates their occurrence on an actual end system. The framework also has the capability of injecting and observing other network faults such as omission, corruption, delay and masquerading but these are not in the scope of this paper. The framework hides the fault injector module from the network participants (end systems and switches). It is connected to the network under test in a cut-through paradigm, eliminating the necessity to modify the software or hardware of the TTEthernet network participants for the purpose of fault injection. This makes the framework easy to use and portable without requiring any modification on the device to be validated.

This paper is composed of 5 sections. Section 2 describes the TTEthernet protocol operation, section 3 presents the babbling idiot injection framework. The experimental organization and results are presented in section 4 and section 5 gives the conclusion of this paper.

## II. TTEthernet Protocol

TTEthernet is a fault-tolerant communication protocol that extends the legacy Ethernet (IEEE 802.3) to provide reliability and determinism for safety critical applications. By using a decentralized clock synchronization on a switched network, TTEthernet enables message transmission of time triggered frames with reduced jitter. Determinism with strict guarantees is provided through a fixed schedule for the traffic over TTEthernet [10]. TTEthernet was standardized as SAE AS 6802 in 2011. A TTEthernet network is composed of end systems, switches and links. Fig. 1 depicts an example of a redundant setup for a TTEthernet network consisting of four end systems and four switches. As shown in Fig. 1, TTEthernet provides the facility to transmit frames with redundancy. For example a message sent from $end\_system\_1$ to $end\_system\_4$ is duplicated and travels via two channels ($l_0 - l_2 - l_4$ and $l_5 - l_7 - l_9$). The receiving $end\_system\_4$ picks one of the transmitted valid frame and discards the other frame.

Three traffic types namely, time-triggered (TT), rate constrained (RC) and best effort (BE) messages are exchanged between end systems through the connected links and switches. The classification of messages transmitted over the TTEthernet

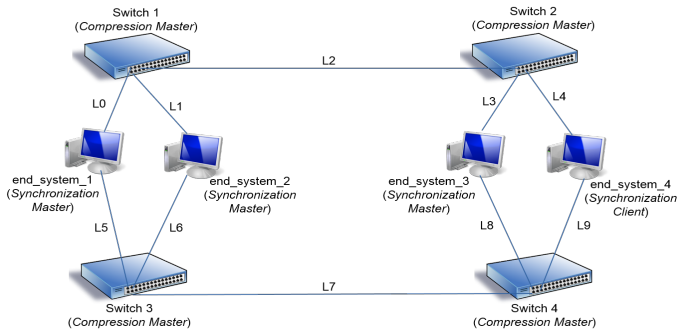



Fig. 1. Example of a redundant TTEthernet setup.

network in decreasing priority order for scheduling, are TT, RC, and BE traffic. The end systems and switches participate in a transparent clock synchronization mechanism. Synchronization messages are realized using protocol control frames (PCF). The end systems and switches can be configured as synchronization masters (SM), compression masters (CM) or synchronization clients (SC). The synchronization algorithm is performed in two steps [11]. In the first step, the end systems configured as SMs initiate the synchronization algorithm by sending PCFs containing their local time to the compression masters (usually switches) and each compression master produces a new PCF called compressed PCF. In the second step, the compressed PCFs are sent by all CMs to SMs and SCs. SCs are only recipients of the compressed PCFs; they do not send PCF frames. Switches and end systems can be configured as SCs. The synchronization service provides a global notion of time amongst all network participants. Having established this global time, periodic time slots are allotted in a predetermined schedule for the transmission of TT messages.

Schedules for the reception and transmission of TT frames are defined offline at development time, providing temporal isolation. A TT frame is discarded when it arrives outside the acceptance window. This mechanism is called temporal firewall [12]. The RC traffic conforms to the ARINC 664p7 specification [13], while the BE traffic conforms to the legacy Ethernet operation. The combined operational principle of fault tolerant clock synchronization, in combination with TT interaction and the ARINC 664p7 RC operational principle on top of legacy Ethernet make up the TTEthernet SAE AS6802 standard. This is illustrated in Fig. 2, depicting the robust bandwidth partitions that accommodates the three network classes: TT, RC and BE [1].

A priori knowledge about the permitted timing of TT and RC traffic provides protection against BIFs. Frames that are received outside the pre-configured acceptance window are discarded. Likewise the traffic policing functionality described in [13] provides protection against BIFs based on the configuration. One frame per bandwidth allocation gap (BAG) is expected and if an extra frame is found inside a giving BAG it is discarded. In addition, there is the concept of jitter which in ARINC 664p7 specifies the time window around

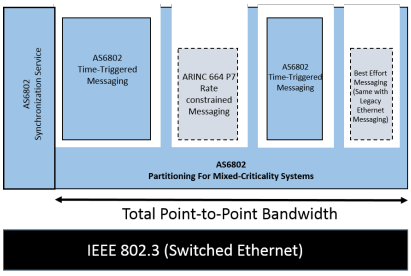



Fig. 2. Robust bandwidth partition of TTEthernet [1].

BAG, measured from last frame reception in which a frame is accepted. Frames that arrive outside maximum jitter bounds are also discarded. The switch discards frames that do not meet the defined BAG and jitter requirement specified in the standard by implementing a policing function.

TTEthernet addresses collision conflicts between frames by assigning priority to each frame. Although the SAE AS6802 standard does not mandate the use of a specific priority scheme, it however recommends in decreasing order : PCF, TT, RC, BE [1]. Priority assignment resolves collision before the start of a transmission, but extra mechanism is needed if a low priority message is already transmitting when a higher priority message is ready for transmission. Three methods [11] that address these conflicts are described as follows:

- Preemption: If a high priority message arrives a switch when a low priority message is being transmitted, the low priority message is halted. The switch then establishes a minimum silence and then transmits the high priority message at the time slot specified in development time.
- Timely block: Considering that the switch knows a priori, the arrival time for the high priority message, the switch will not forward any message at these time slots, in order to ensure that the high priority frame is transmitted without delay.
- Shuffling: If a low priority message is transmitting when a higher priority message arrives, the low priority message is allowed to finish its transmission before the higher priority frame is transmitted. Shuffling presents an optimal solution from an utilization point of view since it does not truncate frames or block the outgoing ports of low priority message. However, the real-time quality is compromised to achieve shuffling.

A verification and validation framework is required for communication vendors that offer products implementing this standard. To verify the functionality and behavior of TTEthernet services, a fault injection framework is required for the purpose of testing. The babbling idiot framework designed herein provides the following capabilities:

- Verifies the conformity of a TTEthernet device to the standard.
- Validates the fault containment coverage of the hardware implementation of TTEthernet.
- Investigates the application behavior in the presence of BIFs.

## III. BIF Injection Approach

The framework designed herein injects BIFs directly into the communication link of the TTEthernet network. The fault injection framework was designed to realize a platform for a controlled experiment, where the behaviour of the TTEthernet network is observed in the presence of BIF. The framework uses a Xilinx field programmable gate array (FPGA) [14] as the injector. The experiment is controlled from a central monitoring station. The central monitoring station performs the following services:

- Collects traffic data over the network via connections to a parallel network setup.
- Controls the experiment via a UART connection to the FPGA.
- Loading of configuration parameters.

Data analysis is carried out off-line after the completion of the experiment.

### A. Fault Injector

The fault injector (FI) is implemented using an FPGA. The FPGA is used to generate frames that emulate the frames generated by a TTEthernet faulty end system. The FPGA is connected and configured with the parameters of the end system, of which it intends to emulate. The three frame classes of TTEthernet TT, RC, and BE are generated from the fault injector in an untimely manner from the FPGA. Fig. 3 illustrates the internal architecture of the FPGA model for the fault injection framework. The FI architecture shown here is part of an ongoing development for a framework capable of injecting a broader variety of network faults, this is the reason for the ingress port component shown. The component FI-IP generates the babbling idiot message. The information about the frame structure and content that is peculiar to an emulated end system is stored in the configuration module. These information include, the destination address (Constant field and Virtual link) [1], source address, frame size, sample payload and activation time. The generated frames exit via the egress port into the network.

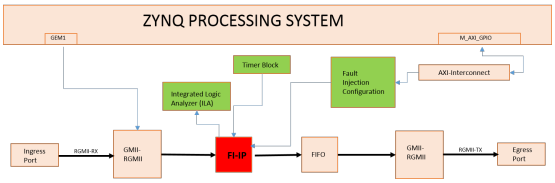



Fig. 3. Internal Architecture of BIF Injection Framework.

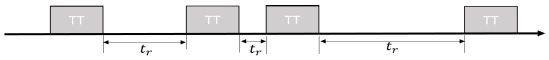
### B. Fault Models

The babbling idiot failure mode is emulated herein by implementing a random interval between two consecutive frames. The time interval between two consecutive frames is known as the inter frame gap (IFG). By making the IFG a random variable, frame transmissions are performed in an untimely manner. Considering the IFG (L) as a positive random variable

in range 0 to K, where K is configurable positive integer denoting the maximum value that L can attain. This can be expressed as $0 < L \leq K$.

The FI-IP module in Fig. 3 is used to implement a random function for the IFG. During a Golden Run (GR) scenario, TT frames are transmitted at constant periodic intervals ($t_p$) as shown in Fig. 4(a). GR describes the experiment when no fault is injected. Therefore L would be constant and equal to $t_p$. Fig. 4(b) illustrates the random interval between two consecutive TT traffic. L becomes a random variable when the BIF is activated. Fig. 4(b) illustrates that $t_p$ assumes a random value $t_r$ that changes at random to emulate the untimely generation of frames.


(a) Behaviour of TT frame during golden run.




(b) TT frame illustrating random time interval when babbling idiot is activated.

Fig. 4. Illustration of inter frame gap between TT frames.

### C. Configuration Parameters

Information regarding the frame structure and content are loaded into the fault injection configuration module shown in Fig. 3. Considering that the FPGA assumes the position and functionality of an emulated end system, it is required to preconfigure the FPGA with the same parameters of the emulated end system. For example, the virtual link identifications (VLIDs), source address and payload size are preconfigured on the FPGA.

### D. Readout Module

A parallel network was setup to monitor and observe the system under test. All frames transmitted over the network are captured using a high speed network accelerator card (NAC) [15] . The frame contents are viewed on a monitoring end station running the Wireshark application [16]. Observation probes are positioned such that latency, jitter and frame delivery can be observed and analyzed. The generated traffic is captured before entering the switch, and if these frames are relayed by the switch, they are also captured. By using the Xilinx's Integrated logic analyzer (ILA) [17], all traffic on the link associated with the FPGA can be observed. Fig. 5 shows a snippet of a frame captured using the ILA core in the FPGA. The row "data_out" displays the frames generated from the FPGA. The row "enable" shows the signal covering the transmission duration. The "Error" row would signal if there is an error in the message during transmission. The observed frame format represents the GMII-RGMII frame handling for a 100 Mbits/s link. In this format each byte is split into two nibbles, where splitted nibbles are duplicated within the block to form a byte and transmitted in an inverted order. The start frame delimeter (SFD) is captured as two bytes x"55" and x"dd". These values corresponds to a single byte x"d5".

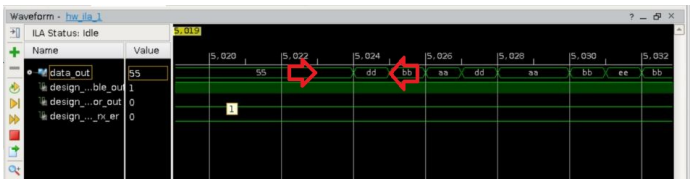



Fig. 5. Vivado ILA tool illustrating SFD.

## IV. Experimental Organization and Results

### A. Experimental Setup

The experiment observed the behavior of a TTEthernet switched network connected with four end systems using 100Mbits/s links. Fig. 6 illustrates the network under test, the network was configured with five virtual links (VL1-VL5). VL1 and VL2 were configured for TT traffic while VL3 to VL5 were configured for RC traffic as illustrated. In Fig. 6 the direction and type of traffic on the VLs are illustrated with the dotted arrows. The arrows associated with each virtual link point in the direction of the traffic. Messages originate from a single sender but can be received by multiple receivers, as shown in the case of VL1 and VL2. Four end systems configured as synchronization masters are connected to a TTEthernet switch configured as a compression master. Passive network taps are placed in the positions A, B, C and D. The network taps are connected to a central monitoring station. All traffic through the links is sniffed in a passive manner using these network taps. The monitoring station is equipped with commerical off-the-shelf (COTS) NACs that receive all traffic from the network taps. Wireshark tool is used to collect all the captured packets. Computations of latency and jitter of the different virtual links are handled offline via a custom C++ program, designed specifically for analysing TTEthernet traffic.

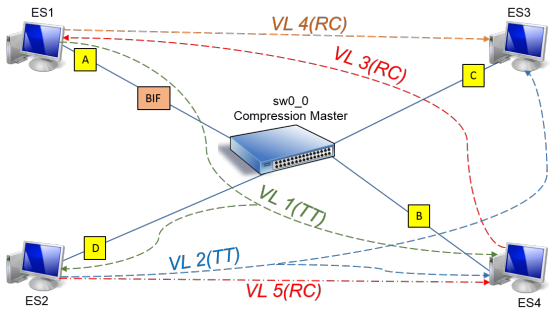



Fig. 6. Experimental setup for Babbling Idiot Fault injection.

The FPGA is used to inject untimely messages on VL1 (TT) and VL4 (RC) with total frame size of 165 bytes. Best effort messages are also injected in the background according to a random interval. The scenario in Fig. 6 is such that message transmission from ES1 is emulated for a case of BIF. On

startup, the FPGA acts completely as a cut-through interface. This cut-through approach introduces a constant delay of 2 $\mu$s to the usual traffic over the channel. When the BIF is activated, ES1 is cut off and messages from it are emulated on the FPGA, except that these messages take an untimely interval, emulating the babbling idiot failure. Information regarding the virtual links of ES1 are preloaded into the FPGA on startup.

The frame size and BAG for each application running on the virtual links were configured for the experiments as follows:

- Configured with a payload size of 100 bytes indicated herein as 100B.
- Configured with a BAG of 15000 $\mu$s.

The above listed configurations are set on each VL. Firstly, we obtained latency, jitter and the number of frame loss for a GR scenario. These initial measured GR values are used to establish a baseline that makes it possible to ascertain the effect of faults on the network when BIF is injected.

### *B. Results and Discussion*

The experimental observation for frame loss, average latency and Jitter on VL2, VL3, and VL5 are shown in Table I. Values for GR scenario were first recorded before the activation of BIF on VL1 and VL4.

TABLE I
EXPERIMENTAL MEASUREMENT - GOLDEN RUN SCENARIO AND BIF INJECTED WITH A PAYLOAD SIZE(100B) AND BAG (0.015S)

| Mode | VL | Avg.Latency | max. Jitter | Frame Loss |
|---|---|---|---|---|
| GR | VL2(ES2-ES3) | 204.626 $\mu$s | 1 $\mu$s | 0 |
| BIF | VL2(ES2-ES3) | 210.96 $\mu$s | 89 $\mu$s | 2 |
| GR | VL2(ES2-ES4) | 204.659 $\mu$s | 1 $\mu$s | 0 |
| BIF | VL2(ES2-ES4) | 210.498 $\mu$s | 88 $\mu$s | 3 |
| GR | VL3(ES4-ES1) | 19.7907 $\mu$s | 8 $\mu$s | 0 |
| BIF | VL3(ES4-ES1) | 19.9522 $\mu$s | 17 $\mu$s | 2 |
| GR | VL5(ES2-ES4) | 19.7551 $\mu$s | 7 $\mu$s | 0 |
| BIF | VL5(ES2-ES4) | 28.9759 $\mu$s | 181 $\mu$s | 1 |

*1) Average Latency:* The experiments considered latency in the context of end to end delay between two nodes. It expresses the duration taken by frames to get from one end system to another. The network schedule was configured with a period of 20 ms for TT traffic, and a BAG of 8 ms for RC traffic. Although the application was configured to send all messages with a BAG of 150 ms, these messages only leave the TTEthernet network card according to the predefined network schedule. When BIF is activated the average latency on VL2 (TT) is slightly increased. For the RC VLs, the increase in average latency when BIF is triggered is more pronounced compared to TT VL. These differences in average latencies are 6.334 $\mu$s and 9.2208 $\mu$s for TT VL(ES2-ES3) and RC VL(VL2-ES4) respectively.

*2) Jitter:* Increased jitter was observed across all VLs when BIF is activated. Under GR scenario, the jitter observed across TT VLs was bounded at 1 $\mu$s. To accomplish a detailed evaluation on the effect of BIF on TTEthernet network, it is important to know what conflict resolution (between TT and other traffic) method is implemented on the switch. This has

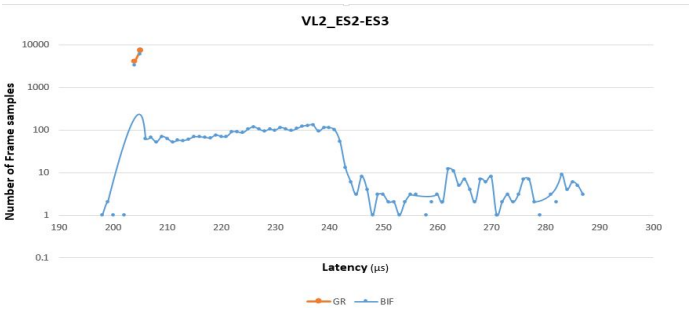



Fig. 7. Virtual Link 2 (End system 2 to End system 3) TT traffic.

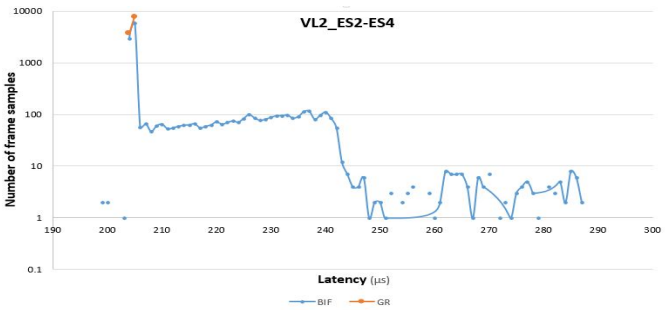



Fig. 8. Virtual Link 2 (End system 2 to End system 4) TT traffic.

an impact on the handling of traffic priority by the switch. The switch in this experiment utilized the shuffling mechanism [1] for conflict resolution. Due to the store-and-forward principle and unfragmented frame transmission implemented by the shuffling mechanism, TT traffic can be delayed by the RC traffic resulting in a pronounced jitter. However, this jitter is bounded as it does not exceed the transmission duration for a single frame.

BIF was injected on VL1 (TT) and VL4 (RC) with a frame size of 165 bytes. In a 100Mbit/s link, a byte is sampled in a bridge every 80 ns. Therefore the switch takes 13.2 $\mu$s in this experiment to relay a complete BIF message. Since shuffling is implemented in the switch, if there is an ongoing transmission of an RC message by the switch when a TT frame arrives, the TT frame is delayed by a maximum of 13.2 $\mu$s plus the IFG and switch processing time. The shuffling method is a tradeoff between optimal bandwidth utilization and real-time

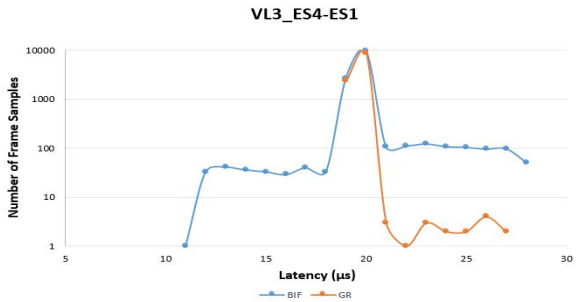



Fig. 9. Virtual Link 3 (End system 4 to End system 1) RC traffic.

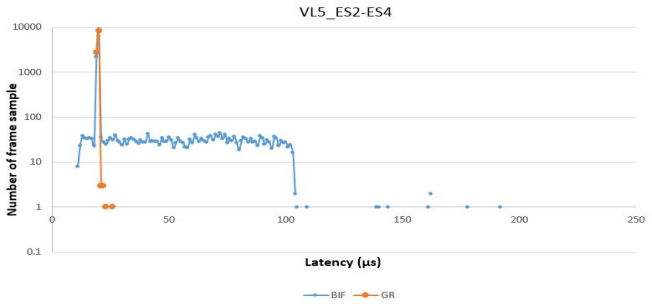



Fig. 10. Virtual Link 5 (End system 2 to End system 4) RC traffic.

quality. Real-time quality is degraded by the observed bounded jitter, however bandwidth is utilized efficiently as there are no truncated frames. Compared to TT VL, the jitter on RC VL5 is much more significant. The buffer size specification of the switch plays a major role on how RC messages are handled. Flooding the switch with BIF has more significant consequences as the buffer limit is exceeded resulting in more pronounced jitter for RC traffic.

Fig. 7 and Fig. 8 illustrate a plot on the number of frame samples against latencies observed in the experiment for VL2 (ES2-ES3) and VL2 (ES2-ES4) respectively. The deduced jitter from the plot is bounded within a maximum of 89 $\mu$s for ES2-ES3 and 88 $\mu$s for ES2-ES4. The jitter effect on the TT virtual link under BIF is affected by the frame size of the BIF message. The RC VLs illustrated in Fig. 9 (ES4-ES1) and Fig. 10 (ES2-ES4) are not only affected by the frame size of the BIF message. They are in addition affected by the frequency of frame generation and size of the configured switch buffer. As a result the jitter observed for the RC VL was disproportionate.

*3) Frame loss:* Frame loss was recorded for both TT and RC traffic under BIF. The arrival of frames to a receiving end system outside an acceptance window that was specified a priori are discarded. In addition frames that did not meet the bandwidth allocation gap (BAG) requirement of ARINC 664 were also discarded. Both RC VLs and TT VLs were prone to frame loss under BIF.

## V. Conclusion

A fault injection framework capable of evaluating the fault containment of TTEthernet implementations against babbling idiot failures was developed in this work. The framework utilized COTS devices in establishing a monitoring station and an FPGA connected according to a cut through paradigm to inject babbling idiot faults. Experimental results obtained demonstrated the use of the framework in providing statistically useful data for the evaluation of TTEthernet fault containment against babbling idiot failures.The framework provided a platform to observe the impact of BIF on RC and TT communication. It was observed that an end system which generates babbling idiot messages on a given virtual link could have a bounded effect of jitter (up to 123 $\mu$s for maximum frame size) on a TT virtual link sharing the same switch due to shuffling. Shuffling thus affects the fault containment of TT communication under the presence of BIF. Timely blocking however will be a recommended option to decrease the jitter for TT communication. The framework thus provide the platform to apply direct measurement necessary for quantification of availability and reliability of TTEthernet from different vendors.

## Acknowledgment

This work has benefited from funding from the Shift2Rail Joint Undertaking under grant agreement No. 730830. This Joint Undertaking receives support from the European Unions Horizon 2020 research and innovation program. TTTech Vienna played a major role in the provision of hardware and technical support for the work done in this project.

## References

[1] SAE, *The Time Triggered Ethernet AS6802*, Society for Automotive Engineers Std., 2011.
[2] *Timing and syncrhonisation for time-sensitive applications in birdged local area networks, IEEE 802.1*, Institute of Electrical and Electronics Engineers Std., 2011.
[3] *Virtual bridged local area network amendment 14: Stream reservation protocol, IEEE 802.1 Qat*, Institute of Electrical and Electronics Engineers Std., 2010.
[4] *Virtual bridged local area network amendment 12: Forwarding and queuing enhancement for time-sensitive streams, IEEE 802.1 Qav*, Institute of Electrical and Electronics Engineers Std., 2009.
[5] *Audio Video Bridging (AVB) Systems, IEEE 802.1 BA*, Institute of Electrical and Electronics Engineers Std., 2009.
[6] K. Wang, A. Xu, and H. Wang, "Avoiding the babbling idiot failure in a communication system based on flexible time division multiple access: A bus guardian solution," in *2009 IEEE International Symposium on Industrial Electronics*, July 2009, pp. 1292–1297.
[7] G. Buja, A. Zuccollo, and J. Pimentel, "Overcoming babbling-idiot failures in the flexcan architecture: a simple bus-guardian," in *2005 IEEE Conference on Emerging Technologies and Factory Automation*, vol. 2, Sept 2005, pp. 8 pp.–468.
[8] C. Temple, "Avoiding the babbling-idiot failure in a time-triggered communication system," in *Digest of Papers. Twenty-Eighth Annual International Symposium on Fault-Tolerant Computing (Cat. No.98CB36224)*, June 1998, pp. 218–227.
[9] V. Lari, M. Dehbashi, S. G. Miremadi, and M. Amiri, "Evaluation of babbling idiot failures in flexray-based networkes," *IFAC Proceedings Volumes*, vol. 40, no. 22, pp. 399 – 406, 2007, 7th IFAC Conference on Fieldbuses and Networks in Industrial and Embedded Systems. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474667016351825
[10] J. Dvok, M. Heller, and Z. Hanzlek, "Makespan minimization of time-triggered traffic on a ttethernet network," in *2017 IEEE 13th International Workshop on Factory Communication Systems (WFCS)*, May 2017, pp. 1–10.
[11] W. Steiner, G. Bauer, B. Hall, and M. Paulitsch, *Time-Triggered Communication*, 2nd ed. Boca Raton: CRC press, 2011, ch. 8 : Time-triggered Ethernet In R. Obermaisser, Ed., pp. 181–220.
[12] M. Heller, "Scheduling of the ttethernet communication," Master's thesis, Czech Technical University in Prague, Czech Republic, Prague, May 2016.
[13] ARINC, *Aircraft Data Network PART 7: Avionics Full Duplex Switched Ethernet (AFDX) Network*, AIM Avionics Databus Solutions Std., 2006.
[14] XILINX. (2018) All programmable soc with hardware and software programmability. [Online]. Available: https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html
[15] Napatech. (2018) Napatech smartnics. [Online]. Available: https://www.napatech.com/
[16] G. Combs. (2018) Wireshark. [Online]. Available: https://www.wireshark.org/
[17] XILINX. (2016) Integrated logic analyzer v6.2 logicore ip product guide.