Proceedings of 7th Transport Research Arena TRA 2018, April 16-19, 2018, Vienna, Austria

Fault Injection Framework for Time Triggered Ethernet

Onwuchekwa Daniel*, Foo Jia-Jiann, Obermaisser Roman

"Embedded Systems Group, University of Siegen, Siegen, 57076 Germany"

Abstract

Time-Triggered Ethernet (TTEthernet) is increasingly being deployed in avionics, space applications and other safety-critical systems. The main reasons are its real-time determinism, support for mixed-criticality and certification. Due to the application in these safety-critical systems, TTEthernet was equipped with various fault-tolerance capabilities such as anti-masquerading and a guardian functionality. Consequently a challenge for this kind of system is the realistic and effective validation that can evaluate multiple implementations from different vendors including the protocol itself using Fault Injection (FI). This paper presents an FI framework for TTEthernet using an FPGA which is connected to the network under test and injects faults according to a cut-through paradigm. The proposed solution improves previous FI systems, since it addresses time-aware networks and has the capability to target individual traffic classes such as Rate Constrained (RC) and Time Triggered (TT) frames. This provides better controllability, including the ability of the framework to target TTEthernet synchronisation mechanism.

Keywords: "Fault Injection; Dependability Evaluation; Time Synchronisation"

* Corresponding author. Tel.: +49 271 740-2897; fax: +49 271 740-3344.
E-mail address: daniel.onwuchekwa@uni-siegen.de

FAULT INJECTION FRAMEWORK FOR TIME-TRIGGERED ETHERNET

1. Introduction

TTEthernet is a real-time communication protocol that is compatible with standard Ethernet and provides additional services for a distributed real-time system (Kopetz 2008). TTEthernet introduced an implementation that guarantees determinism for critical applications. The provision of partitioning for different streams of network traffic and guarantee of constant latency for critical traffic, promotes its utilization for mixed-critical applications. These benefits have provided a platform to enhance the deployment of both safety- and non-safety-critical applications in a way that reduces cost (e.g. wiring), size, weight and power and yet still guarantees reliability. A variant of TTEthernet technology was successfully deployed in a critical application in aerospace, as part of the on-board data network (ODN) on NASA's Orion spacecraft (Loveless 2015). TTEthernet is standardized as SAE AS6802 (SAE 2011) and the features of TTEthernet are desirable for other safety critical applications such as in the automobile and health industry. One critical set of desirable applications in the automotive industry is the replacement of mechanical systems with electronic systems. The concept of brake-by-wire is a major example, and in this case great emphasis are laid upon safety, and the fulfilment of the timing requirement for these applications. A timing guarantee for the communication involving sensor nodes is strongly required in such a case, however TTEthernet satisfies such timing constraints by providing tight latency, minimal jitter and determinism.

A Failure in such a system that hosts both critical and non-critical applications could not only be catastrophic but also expensive. TTEthernet is equipped with Fault Tolerance mechanisms (FTM) such as redundancy management, clock synchronization and the guardian system, which are implemented to improve its dependability. Due to these FTMs, there is the need for validation activity before deployment. The effect of the various FTMs on host applications needs to be understood, so that the functionality of the TTEthernet application isn't degraded. Also the implementation of safety-critical applications as of today relies on testing to discover possible error states, so as to guarantee fault-free functionality (Ammar and Mohamed 2011). Furthermore for such a system that is entrusted with safety-critical applications, there is a need for a robust verification and validation. Verification is used in the design phase to evaluate components of a system, if they are built correctly. Validation is used to test the functionality of the product and to gain confidence that certain dependability requirements are attained (Yu and Johnson 2003).

Despite the desire to create fault-proof technological devices in today's world, it is obvious that careful design, quality assurance and other fault avoidance technique cannot solely achieve dependability requirements. Proper testing mechanisms are required for this purpose and Fault Injection (FI) can be used to achieve validation of such a system. FI is a dependability evaluation technique that observes the system under test in a controlled experiment by accelerating faults (Yu and Johnson 2003). Two other techniques used for dependability evaluation, when used for either forecasting or verification include Analysis and Field Experience. However dependability evaluation by analysis pose challenges with increasing complexity of a system under test, while that of evaluation by experiment could take a lifetime to gain appropriate confidence. The work herein provides a generic fault injection framework that can be used to satisfactorily validate TTEthernet networks. To the best of our knowledge, this is the first framework that exploits an FPGA cut-through paradigm for direct fault injection on a TTEthernet network. The framework is unique to other FI methods as it has the ability to target and inject faults into individual traffic classes of TTEthernet. The monitoring system for the framework is also abstracted from the TTEthernet network and exhibits no invasive visibility to the TTEthernet network.

The remainder of this paper is organised as follows. Section II discusses related works, fault injection approaches and fault injection relative to validating network protocols, and it further discuses failure modes covered by the framework. Section III provides a background overview of the fault tolerance mechanism for TTEthernet networks. The implementation of the FI framework is presented in section IV. An experiment is carried out in section V to demonstrate the capability of the FI framework. Section VI discusses, concludes and presents further development of the framework.

2. Related Work

Formal methods were used for verification during the development of TTEthernet fault-tolerant algorithms (SAE 2011). FTMs in TTEthernet are introduced in a modular fashion and several verification activities done relied on formal methods (Ammar and Mohamed 2011), (Steiner and Dutertre 2010), (Dutertre et al. 2012). In (Bartols et al. 2011) performance analysis done on TTEthernet to aid validation only introduced a cost effective means to

measure end to end latency using off-the-shelf components, it did not attempt validation by FI. The measurement of end to end latency alone is not sufficient to validate a system that has various FTMs, as it does not carry out effective tests to evaluate all the fault coverage. The measurement technique used in our framework considered that for a distributed system a synchronised timebase is necessary as described in (Bartols et al. 2011) but the demonstration of the framework presented in our work for a single hop configuration utilized a high performance commercially available off-the-shelf (COTS) Network Accelerator Card (NAC). The NAC has the ability to support 4 ports with a nanosecond resolution. Therefore in our framework measuring up to 4 points does not require any clock synchronisation, but for larger TTEthernet network having multiple hops and more than 4 measurement points require clock synchronisation.

2.1. Fault Injection Approaches

FI techniques are categorized into Hardware-Based FI, Software-Based FI, Simulation-Based FI, Emulation-Based FI and Hybrid FI (Zaide et al. 2004) (Yu and Johnson 2003). The choice of which technique to use can be dependent on many factors such as the accessibility to fault target location, development phase of the system under test and the type of failure modes. In testing communication protocols, the use of these techniques could also depend on whether you are testing only a function of the network, software/hardware participants of the network, or the integrated behaviour of the entire network participants. FI can also be categorized in a different perspective as Invasive FI, and Non-Invasive FI. When testing a complex system, it may be impossible to remove the footprint introduced by the fault injector from affecting the system behaviour (Zaide et al. 2004). For example the testing mechanism developed in our work for TTEthernet networks may introduce a time-footprint added to the latency of message transmission. However this can be managed by integrating this added latency into the link transmission delay during analysis. This kind of intrusion is acceptable if such time –footprint is constant or accurately predictable, having small realistic values that can be equated to a given transmission delay.

2.2. Fault Injection on Network Protocols

Earlier implementations of FI carried out for communication networks included the use of tools such as DOCTOR (Han et al. 1995), NFTAPE (Stott et al. 2000), and VirtualWire (De et al. 2003). These however required modifications or additions to the end systems' software or controllers in order to carryout fault injection. This meant that testing tools such as the VirtualWire had to be installed on a host machine. FI that targets networks designed for some industrial specific applications such as sfiCAN (Gessner et al. 2014), achieves fault injection without making any modification to software on controllers. This promotes the injection of faults at the lowest communication layer as it obviates the interference of the testing mechanism with the software that implements various FTMs located in the node. The FI implementation in (Gessner et al. 2014) was carried out using a hub connected to end system nodes and a PC-management station. The FI was implemented inside the hub. This however does not abstract the implemented FI from the Hub, since the FI implementation is inside the hub. FTMs for TTEthernet are implemented on both end system and switches, therefore if complete abstraction of the FI from the network participant is required, the FI is best not implemented in any of the network participants.
Our approach abstracts the testing mechanism from all network participants and it's targeted on TTEthernet with extensible features to Time Sensitive Network (TSN) under development. The FI is neither implemented in the end system software nor the switch software. This approach was chosen to ensure that the FI implementation does not directly interfere with the role of any network participant. The only interaction between the FI and TTEthernet are the instants of fault injection.

2.3. Failure Modes

The cause and consequences of a deviation from the expected functionality of a system are known as the factors of dependability (Zaide et al. 2004). These dependability factors include Fault, Error, and Failure. When a system develops a fault which can be caused by many instances for example electromagnetic interference or physical degradation of the equipment, these faults could produce errors. The errors could be propagated and thus causing failures, the inability of the system to provide the correct output. Generic standards for functional safety such as IEC 61508 specified requirements for data communication when used to implement safety functions. Failure measures of communication processes to be accounted for include transmission errors, repetition, deletion, insertion, re-sequencing, corruption, delay and masquerading (IEC 1997). The detailed discussion in (Steiner et al. 2011) further listed and described additional failure modes such as fail-stop, crash, omission, timing, byzantine/arbitrary failures, Babbling Idiot and Slightly-Off-Specification. One of the major attributes of a TTEthernet network is its ability to maintain a synchronised time base among all network participants. Therefore

this work also included a time synchronisation failure, with the aim of providing means to study the behaviour of the system when various network faults affect the synchronisation capability.

3. Fault Tolerance Mechanisms of TTEthernet

3.1. Fault Tolerant Clock synchronization

The need for clock synchronization is brought about by the drifting of clocks over time. Clock drifts have adverse effects on the performance of time-triggered networks. The major causes of discrepancies between local clocks communicating in a network could be that different clocks have different starting point or different frequency between clocks or different actual tick interval between clocks (Zhang et al. 2016). Various services ensure that devices participating in a safety critical system do meet their timing requirement irrespective of the existence of clock drifts. Some of these synch services include GPS, BD system, NTP, IRIG-B and IEEE 1588. In (Yang et al. 2016), a comparison of these synchronisation services with respect to accuracy, lock Time, cost, Ethernet support and reliability was carried out. It is clear that IEEE 1588 does have some attractive features when it comes to reliability, cost and support for Ethernet. However improving the idea of maintaining a tighter synchronisation led to the development of the TTEthernet synchronisation service. The TTEthernet synchronization service was standardized in AS6802. It added merits to guarantee the precise moments for TT message transmission and reception between network participants. TTEthernet synchronisation service mechanism is clearly detailed in (SAE 2011).

TTEthernet consist of components which participate in a fault tolerant synchronization process. Each of these participants have a local clock. The participants are periodically resynchronized to ensure high precision for TT frames. Each participant can either be a Synchronization Master (SM), Synchronization Client (SC) or a Compression Master (CM). SM delivers synchronization information via frames called Protocol Control Frame (PCF) to establish a global clock baseline, while the CM performs a compression function (SAE 2011) and sends a compressed PCF to all participants in the synchronization service. The SC role functions to only accept the compressed PCF containing the global time, it does not send PCFs. The synchronization services are performed in two steps. During operation, periodically the SMs send PCFs to the CMs. After the CM performs the compression function, secondly it sends the compressed frame to all participants (SM and SC) to correct their local clocks. The PCF frame has the highest priority in the TTEthernet network, since the synchronization service depends strongly on it. Due to fault tolerance, there are multiple devices configured as SM to generate PCFs. Another feature highlighted in the TTEthernet clock synchronization is the concept of a transparent clock in which TTE components takes into account all the transmission delay of this PCF.

The correctness of the fault tolerance synchronization services was proven using formal methods prior to standardization. TTEthernet has been developed and implemented in products and thus this work focuses on validation by fault injection. Due to the increasing drive towards x-by-wire, it is imminent that TTEthernet would continue to thrive, therefore validating implemented or upgraded products by fault injection would increase confidence and enhance certifiability. The fault-injection framework developed in this work has the capability of targeting the PCF which is solely responsible for the clock synchronization. The PCFs have their message format and principle of operation defined in the AS6802.The framework herein provides the platform for inducing various failure modes targeted towards PCF frames. A broad range of fault scenarios can be reproduced to probe the fault coverage of the system.

3.2. Channel Redundancy

Redundancy is a key strategy to tolerating faults in safety critical applications. On a component level TTEthernet can be configured to have either one, two, or three redundant channels. The components (Switches, end systems, and Links) are duplicated to enable concurrent transmission of frames through alternate paths. The receiving components are also equipped with the capability of handling the arrival of frames from alternate paths. In some configurations the first valid frame from the replicated frame that arrives at the receiver wins, and the unused frame is discarded. It is also possible to implore voting wherewith the receiving end system delivers all the redundant messages to the host. The end system then uses a voting mechanism implemented in a software layer for selection of the appropriate frame. Therefore it is expected that with the occurrence of faults in one channel, the redundant channel still delivers a valid frame.

3.3. Commander/Monitor (COM/MON)

TTEthernet realizes a high integrity design by establishing fault containment regions (FCR) using Commander/Monitor (COM/MON) approach (Steiner et al. 2011). COM and MON are self-checking pairs which operate in parallel. The output of both processors are compared and if either of the processors fail or produces a wrong message then the second processor shuts down. This COM/MON feature can be termed as Fail-silent. A fail-silent component provides the correct message or no service at all. It can be seen as a component which produces a message in the event of a fault that is easily recognizable as faulty by all other connected components. The COM/MON is realizable on both switches and end systems, which is intended to ensure that error isn't propagated in the network.

3.4. Guardian (Babbling Idiot Protection)

The Guardian FTM is designed to protect the TTEthernet network against arbitrary faulty end system. For example it is intended that babbling idiot failure mode that is exhibited by an end system would be covered by this mechanism. The FTM is implemented through the switch. For TT traffic it is expected that messages which do not arrive at a specified window (Acceptance Window) are discarded. The switch only forwards frame that fall within the acceptance windows. While for RC messages, a rate –enforcement algorithm is used to control the forwarding of frames with respect to a set temporal distance between two succeeding frames having same frame ID (Steiner et al. 2011).

3.5. Anti-Masquerading

The realization of TTEthernet with ARINC 664 enables the avoidance of communication interaction between components with incorrect ID's. Masquerading is avoided by the use of Virtual Links in TTEthernet. A Virtual link could be described as a unidirectional logical path between a sender and a receiver having a unique identifier called VL Identifier (VL ID). An end system is configured to exchange frames using these virtual links as described in (ARINC 2009) to provide protection against masquerading failures. The fault injection framework presented hereby is extendable to inject masquerading failure mode, this FI injection feature is valuable in testing and validating the hardware implementation of the Anti-Masquerading FTM.

3.6. Fault Tolerant Start up Mechanism

This mechanism implements a fault tolerant-handshake at start-up using a sequence of coldstart and coldstart-acknowledge frames. The TTEthernet consist of network participants having different roles synchronization masters (SM), synchronization clients (SC) and compression masters (CM). It works such that an SM wakes up and listens for PCFs, if no PCF is transmitted then the SMs decides to transmit a coldstart PCF. The transmitted PCF is echoed back by the CM to all SM indicating its readiness. Each of the SMs in the network responds to the echoed coldstart PCF with an Acknowledge PCF to the CM. The CM receives this reply and echoes it back as well. Then the SMs accept the acknowledgement and wait for the start of the next integration cycle to be finally integrated. The AS6802 standard fully describes this process in details. Verification of this algorithm was done using mathematical modelling and model checking to ensure its correctness. The FI framework developed in this work provides a solid feature for the validation of this mechanism implemented in hardware. Various network faults such as babbling idiot, corruption and omission can be used to probe this fault tolerance mechanism implemented for start-up.

4. The Fault Injection Framework

The TTEthernet network is referred herein as the Network under Test (NUT). It is simply composed of TTEthernet end systems and switches connected via Ethernet network links.
The fault injection framework designed in this work provides a means to test fault detection capability of the TTEthernet network, and probe its isolation and recoverability mechanism. Fig. 1 (a) and (b) illustrate two possible configurations of TTEthernet representing a multi-hop redundant setup and a single-hop non-redundant setup respectively. The FI framework can be used for either configurations. For detailed and simplified illustrations of the framework, this paper focuses on the single hop configuration. However the FI framework is suitable for both configurations

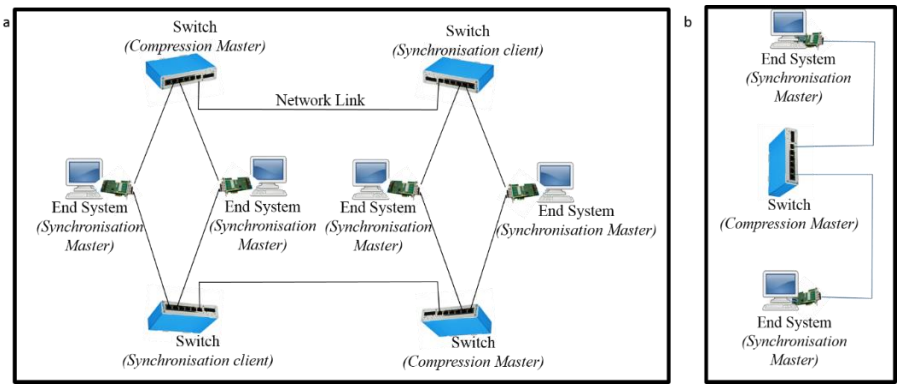

Fig. 1 (a) Multi-hop Redundant configuration for TTEthernet; (b) Single-hop non-redundant configuration for TTEthernet

The multi-hop configuration provides means for utilizing redundancy of the network while the network can also be utilized in a non-redundant manner as shown in Fig. 1 (a) and (b) respectively.

4.1. The Fault Injector

The FI is designed to inject faults into the NUT according to a cut-through paradigm. The FI is implemented using an FPGA. The TTEthernet traffic passes through the FPGA in a cut-through manner while the desired fault is injected on the selected traffic class, location and specified time instant. The injector exploits three techniques in carrying out fault injection namely; byte injection, frame jamming, and frame delay. Byte injection overwrites a byte by inserting a corrupt byte to the payload. Frame jamming blocks and discards a specified frame. Frame delay adds a specified delay to the target traffic class. Table 1 shows the selected techniques used in achieving the different failure modes.

The entire frame is not stored by the FPGA, the frames are forwarded immediately after the headers are read. The FI was modelled using a finite state machine, and is composed of three concurrent states namely; Receiver logic state, retriever logic state and the controller logic state. The receiver logic state implements a shift register of a size only enough to scan through the destination address of a frame. The shift register introduces a constant delay Δt_c. The frame destination address is read in this time and with the frame structure of TTEthernet this information is sufficient to determine the class of the frame (TT, RC or BE). The receiver logic state is a complex state and consist of sub-states such as golden run, reset and the fault injection sub states, as shown in Fig. 2 (a). In the receiver logic, transition cannot be made between FI sub-states and every experiment for the individual FI sub state is ran until reset before another one is started. Since the current internal chip state of the FPGA cannot be guaranteed after introducing a fault, it is best practice to reset the system before running another FI experiment. The controller logic state shown in Fig. 2 (b) is responsible for the transition made in the receiver logic states and it controls the flow of the experiment. The retriever logic state shown in Fig. 2 (c) functions to continually output the data in the shift register. This state runs in parallel to the receiver logic state but functions with the same clock speed as the receiver logic state.

4.2. The Monitoring System

The monitoring system is composed of COTS devices; network taps, high resolution NACs, and a monitoring computing station. These devices are used to obtain readout with nanosecond accuracy for timestamps on frames, in order to determine the latency of the transmitted frames over the NUT. The setup illustrated in Fig. 3 shows two parallel networks, the measurement network and the NUT. The measurement network is depicted with dotted lines over the NUT. It was necessary to deploy a parallel network for measurements in the FI framework because there cannot be any reliance on the network with the injected faults. Hardware timestamping was also necessary to ensure accurate measurements that would meet the timing requirement of measuring time-triggered frames. The monitor probes are placed in acquisition points next to the end-systems. The monitor probe consists of a passive network TAP and is connected to the high resolution 4 port NAC for precise time stamping of frames in hardware. The network TAP extracts a copy of the frame from the NUT in a passive manner while the NAC is used to provide a hardware timestamp before the frames are received in the monitoring station. The NAC is connected to the monitoring station while the original transmitted frame travels completely unaffected through the NUT.

Table 1. Illustration on how frame injection, delay and jamming are used to implement different failure mode.

Failure Mode	Description	Byte Inj.	Frame Delay	Frame Jam.
Omission failure	An end system fails to send a frame or a receiver fails to receive.			x
Corruption failure	The message transmitted is corrupted.	x		
Delay failure	A message transmitted over the network fails to meet its timing requirement.		x	
Unintended message repetition	A receiver gets a message more times than intended.	x		x
Link failure	The link fails to transmit data from end system to end system- (All incoming and outgoing traffic is jammed)			x
Crash failure	Either component of the TTEthernet network (Switch or End system) fails to produce any output (All links connected to device is cut out by jamming incoming and outgoing frames on the affected links)			x
Babbling Idiot	The end system or switch send untimely messages sporadically over the network	x		
Time Synch Failure	This failure mode is designed specific to Time triggered communication. This involves a synchronization failure which is implemented by using any of the aforementioned failure modes to target PCF frames.	x	x	x
Masquerading Failure	This is when an end system assumes the identity of another end system	x		

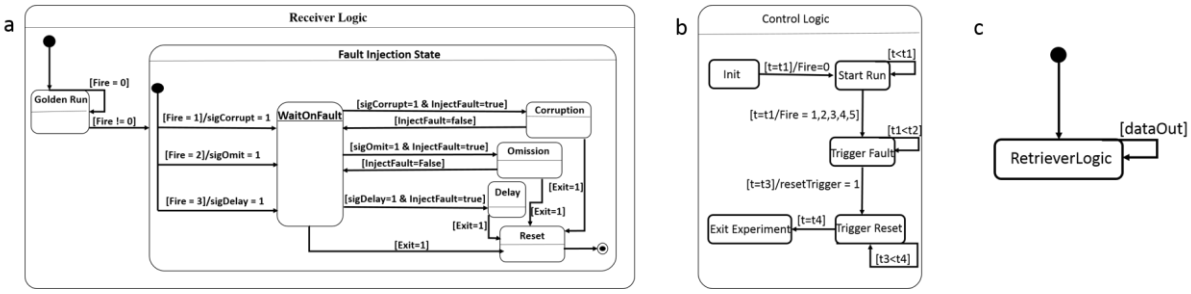

Fig. 2 (a) Receiver logic state; (b) Control logic state; (c) Retriever logic state

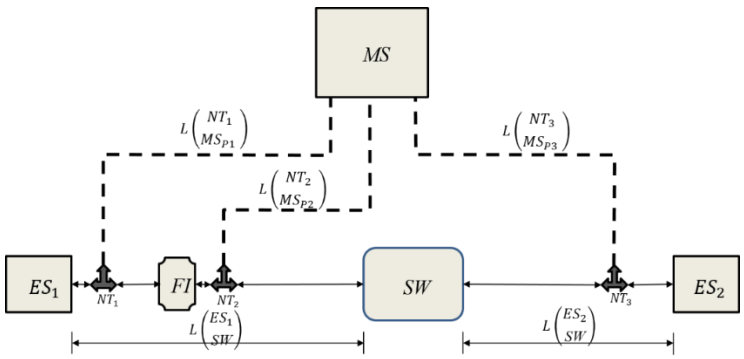

Fig. 3 Single Hop Non-redundant network connected to a monitoring station

5. Experiment and Discussion

Recent FI techniques implemented to validate network protocols carries out injection from either the switches or end systems. This usually requires the modification of the hardware/software design of the network components. These modifications restricts the FI to the specific protocol of which the switch or end system was designed for. However since the method implemented in our work abstracts the FI from both the end systems and switches, its advantage is the provision of generic capability to the FI to target other network protocols. Such portability is made possible as the FI mechanism is not built into a network specific switch or end system. In addition, the possibility of introducing unintended errors into the switches or end system due to modifications is eliminated. This unique method is attained by placing the FI in accordance to a cut-through paradigm, where network faults are injected on the link between the end systems and switches.

Byte injection, frame delay and frame jamming were used both in combination and individually in some cases to achieve all the various failure modes represented in this framework. The experiment performed in this work demonstrates the capability of the framework with a focus on delay, omission and corruption faults. Three sets of experiment performed included;

- Corruption faults were also injected targeting all frame classes.
- Omission faults were injected into the TTEthernet network targeting all frame classes.
- Delay faults were injected to evaluate the capability of the FI in targeting all frame types (PCF, TT, RC, and BE)

The experiment is performed on the single-hop non-redundant configuration shown in Fig. 3. The two end systems (ES_1 and ES_2) are connected via a TTEthernet switch. Mixed-critical traffic (TT, RC, BE) is transmitted from ES_1 to ES_2 on a 100 Mbit/s link. The parallel network consisting of three passive network taps is used to measure latency, from which jitter can be derived. However the focus was to determine packet loss caused by the faults injected. It was expected and confirmed that corrupted frames are dropped as shown in the Table 2. The position of the three network taps is such that NT_1 and NT_3 are used to derive the latency of frames over the network while NT_2 is used to confirm that the injected faults were carried out precisely. All messages from the network taps (NT_1, NT_2 *and* NT_3) are accurately time stamped by the high-resolution NAC card. The results are further analysed using Wireshark. The monitoring network introduces no significant delay to the NUT, since the network taps used are passive by design.

Corruption and omission faults were injected individually into the TTEthernet network targeting TT, RC, and BE traffic classes respectively. Table 2 illustrates the results of injecting corruption fault with an FER (Frame error rate) of 3. The FER indicates the frequency at which corruption fault was applied. FER of 3 corrupts every 3rd frame.

Table 2. Results of Corruption Failure (FER = 3)

Frame Type	Total Number of observed Frames (Cor.fault)	Number of Injected Faults (Cor.fault)	Number of Frame Received (Cor.fault)
TT	110	35	75
RC	102	34	68
BE	105	36	69

Table 3. Results of Omission Failure (FER = 3)

Frame Type	Total Number of observed Frames (Omi.fault)	Number of Injected Faults (Omi.fault)	Number of Frame Received (Omi.fault)
TT	120	40	80
RC	117	39	78
BE	114	39	75

The results show that all corrupted frames were dropped by the switch. Table 3 illustrates the impact of omission fault. The results indicate that the omitted frames were not forwarded to the switch and receiving end system. The integrated logic analyser (ILA) was used in figure 4 and 5 to illustrate the effect of delay fault on TTEthernet network. The ILA is a Xilinx IP (Intellectual Property) core that is used to monitor internal signals of a design (Xilinx 2016). The ILA was set to capture at 25 MHz which corresponds to the 100 Mbit/s clock setting needed in GMII-RGMII (Gigabit Media Independent Interface – Reduced Gigabit Media Independent Interface) block (Xilinx 2016).The ILA captured the inverted nibble format for the 100 Mbit/s network. This is to say that every byte transmitted on the 100 Mbit/s settings through the GMII-RGMII block is split into 2 nibbles and the less significant nibble is transmitted before the higher significant nibble. The signal *flag_delayFault* is the trigger for capturing the wave form, it is the instance when delay fault was fired. The cut through paradigm introduces a fixed

30 clock cycle computational delay as shown in Figure 4. Figure 5 shows when 1 µs delay was injected on TT frame, it takes extra 5 clock cycles for the buffer operation which stores the frame while it was being delayed. The injected 1µs delay is equivalent to a 25 clock cycles delay. The sum of the 5 clock cycle buffer operation, 25 clock cycle delay and 1 clock cycle instance for outputting the data is equal to the 31 clock cycle evident in the ILA shown in figure 5.

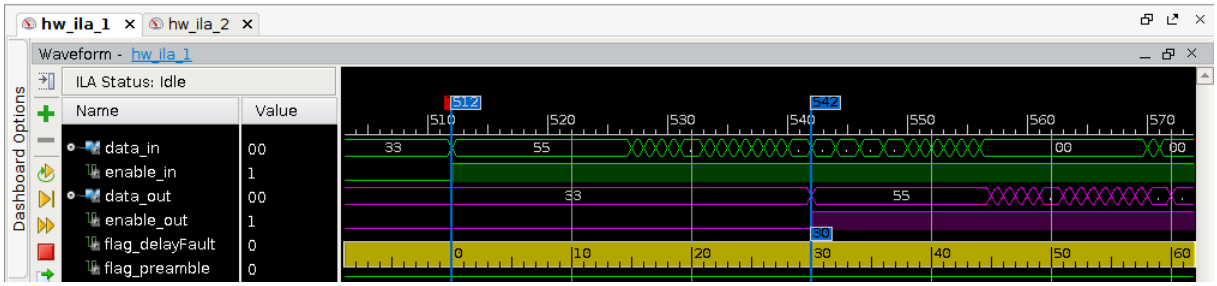

Fig. 4 ILA illustration of Golden run

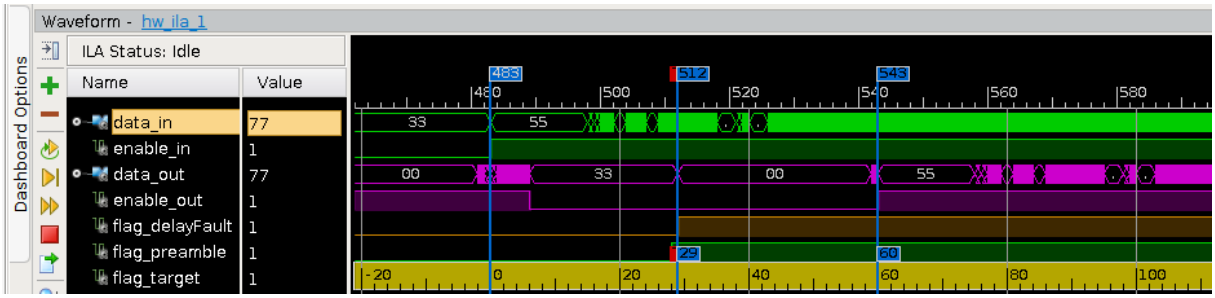

Fig. 5 ILA illustration for delay fault of 1µs

6. Conclusion

The designed FI framework provides the capability of targeting the individual traffic classes of the TTEthernet network. It provided a means that could be utilized in evaluating the FTM discussed in this work. It also utilizes a high precision COTS NAC that is capable of measuring latencies in a distributed manner, however current measurement nodes are limited to 4 for each NAC. The results obtained satisfy our expectation that frames subjected to corruption and omission faults were dropped at the switch level. The framework was also able to inject specific delay faults targeting any of the TTEthernet traffic class. Future work is focused on extending the framework to test redundant networks that implements all the FTMs of TTEthernet. The framework will thus be used to validate hardware implementations of TTEthernet. In addition to cover more measurement points using the COTS NAC card, an efficient synchronisation mechanism shall be used to ensure that the same global time is used for measuring latency on a distributed system.

Acknowledgements

This work has benefited from funding from the Shift2Rail Joint Undertaking under grant agreement No. 730830. This Joint Undertaking receives support from the European Union's 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.

7. References

Ammar, M., Mohamed, O., 2011. Formal Verification of Time-Triggered Ethernet Protocol using PRISM Model Checker. In: IEEE International Conference on Microelectronics, ICM 2011 Proceeding 1-5.

ARINC, A.D.N.P., 2009. 7: Avionics Full-Duplex Switched Ethernet Network-ARINC Specification 664 P7-1.

Bartols, F., Steinbach, T., Korf, F., Schmidt, T.C., 2011, March. Performance analysis of time-triggered ether-networks using off-the-shelf-components. In Object/Component/Service-Oriented Real-Time Distributed Computing Workshops (ISORCW), 2011 14th IEEE International Symposium on (pp. 49-56). IEEE.

De, P., Neogi, A., Chiueh, T., 2003. VirtualWire: A Fault Injection and Analysis Tool for Network Protocols, in proceeding of the 23rd International Conference on Distributed Computing System (ICDCS'03). pp. 214-221.

Dutertre, B., Easwaran, A., Hall, B., Steiner, W., 2012. Model-based Analysis of Timed-Triggered Ethernet, in 2012 IEEE/AIAA 31st Digital Avionics System Conference (DASC). pp. 9D2-1-9D2-11.

Gessner, D., Barranco, M., Ballesteros, A., Proenza, J., 2014. SfiCAN: A Star-Based Physical Fault-Injection for CAN Networks, in IEEE Transaction on Vehicular Technology 63, 3, 1335-1349.

Han, S., Shin, K.G., Rosenberg, H.A., 1995. Doctor: an integrated software fault injection environment for distributed real-time systems, in proceeding of 1995 IEEE International Computer Performance and Dependability Symposium. pp. 204-213.

IEC 61508, 1997. Functional safety of electrical/electronic/programmable electronic safety-related systems. Parts 1-7. IEC, Geneva.

Kopetz, H., 2008. The Rationale for Time-Triggered Ethernet, In RTSS'08: Proceeding of the 2008 Real-Time Systems Symposium. IEEE Computer Society, pp. 3-11.

Loveless, A., 2015. On TTEthernet for Integrated Fault-Tolerant Spacecraft Network, in AIAA SPACE 2015 Conference and Exposition. American Institute of Aeronautics and Astronautics, Reston, Virginia.

SAE 2011. Time-Triggered Ethernet, in "SAE Standard AS6802"

Steiner, W., Bauer, G., Hall, B., Paulitsch, M., 2011. Time-Triggered Ethernet, in "Time-Triggered Communication". In: Obermaisser R., (Ed). CRC press, Boca Raton. pp. 181-220.

Steiner, W., Dutertre, B., 2010. SMT-Based Formal Verification of a TTEthernet Synchronization Function, in "Formal Methods for Industrial Critical Systems". In: Kowalewski S., Roveri, M. (Eds.). Springer, Heidelberg, Berlin, LNCS 6371, pp. 148-163.

Stott, D., Floering, B., Kalbarczyk, Z., Lyer, R.K., 2000. A framework for assessing dependability in distributed systems with lightweight fault injectors, in 4th IEEE International Computer Performance and Dependability Symposium. pp. 91-100.

Xilinx 2016. Integrated Logic Analyzer v6.2 "logiCore IP Product Guide"

Xilinx 2016. GMII to RGMII v4.0 "logiCore IP Product Guide"

Yang, X., Zhang, S., Li, H., 2016. Research and implementation of precise time synchronisation system of microgrid based on IEEE 1588, in 2016 IEEE Advanced Information Management, Communicates, Electronics and Automation Control Conference (IMCEC). pp. 258-261.

Yu, Y., Johnson, B. W., 2003. Fault Injection Techniques, in "Fault Injection Techniques and Tools for Embedded Systems Reliability Evaluation". In Benso, A., Prinetto, P. (Ed.). Kluwer Academic Publishers, Boston, pp. 7-39.

Zaide, H., Ayoubi, R., Velazco, R., 2004. A Survey on Fault Injection Techniques. International Arab Journal of Information Technology 1, 2, pp. 472-481

Zhang, Y., He, F., Lu, G., Xiong, H., 2016. Clock synchronisation compensation of Time-Triggered Ethernet based on least squares algorithm, in 2016 IEEE/CIC International Conference on Communication in China (ICCC Workshops). pp. 1-5