Fault-tolerant Clock Synchronization using Precise Time Protocol Multi-Domain Aggregation

Eleftherios Kyriakakis[*§], Koen Tange [*§], Niklas Reusch[*], Eder Ollora Zaballa[†],
Xenofon Fafoutis[*], Martin Schoeberl[*], and Nicola Dragoni[*]
{elky, kpta, nikre, eoza, xefa, masca, ndra}@dtu.dk
[*]DTU Compute, [†]DTU Fotonik,
Technical University of Denmark

***Abstract*—Distributed real-time systems often rely on time-triggered communication and task execution to guarantee end-to-end latency and time-predictable computation. Such systems require a reliable synchronized network time to be shared among end-systems. The IEEE 1588 Precision Time Protocol (PTP) enables such clock synchronization throughout an Ethernet-based network. While security was not addressed in previous versions of the IEEE 1588 standard, in its most recent iteration (IEEE 1588-2019), several security mechanisms and recommendations were included describing different measures that can be taken to improve system security and safety. One proposal to improve security and reliability is to add redundancy to the network through modifications in the topology. However, this recommendation omits implementation details and leaves the question open of how it affects synchronization quality.**

This work investigates the quality impact and security properties of redundant PTP deployment and proposes an observation window-based multi-domain, PTP end-system, design to increase fault-tolerance and security. We implement the proposed design inside a discrete-event network simulator and evaluate its clock synchronization quality using two test-case network topologies with simulated faults.

***Index Terms*—time-sensitive networking, precise time protocol, clock synchronization, fault tolerance, availability, safety**

I. Introduction

Modern Cyber-Physical Systems (CPS) are becoming increasingly connected to the Internet through the advancements of Fog Computing and Industrial Internet of Things. Thus nowadays, security becomes an essential factor in the design of such systems, in addition to the traditional safety and reliability requirements [1], [2].

Time-triggered communication is often used in distributed CPS that require strict guarantees on the timing of messages. Such systems need a high precision global notion of time to be shared among the nodes in the network to achieve synchronous scheduled communication and computation [3], [4]. Time-Sensitive Networking (TSN) [5] is a newly developed standard that aims to enable deterministic real-time communication for mixed-criticality traffic while preserving the high-bandwidth capabilities of Ethernet. It is developed by the TSN Task Group as an extension to the 802.1 Ethernet standard and consists of many sub-standards for different components. TSN uses a profile (802.1 AS-Rev [6]) of the IEEE 1588 Precision Time Protocol (PTP) standard [7] to enable accurate clock synchronization. Although PTP has been in use for decades, recent research indicates that this protocol's security and safety aspects have been overlooked, leaving it vulnerable to time synchronization attacks [8]. An attack on an automated factory's network time would disrupt the communication and computation schedule leading to missed deadlines and messages. This could have catastrophic consequences both in the production line and operating machinery, as well as possibly endanger human lives or the environment.

To address some of these issues, the IEEE Precise Networked Clock Synchronization Working Group has included various security measures in the updated IEEE 1588-2019 standard [9]. This updated standard proposes several measures involving redundancy to mitigate security and safety issues due to unavailable links. To support the proposed redundancy, the standard recommends using a voting algorithm to derive a converged clock offset from the multiple domains. However, no further information is given, leaving the algorithm's choice and its implementation to the user.

While distributed consensus and voting algorithms are extensively studied [10], to our knowledge, no such work exists in the context of highly time-sensitive PTP networks. We explore the concept of fault-tolerant clock synchronization within TSN and propose a multi-domain synchronization scheme that uses redundant paths combined with frame aggregation and a time-based observation window to achieve secure and fault-tolerant operation. We evaluate the proposed approach by simulating three test-case network topologies in a discrete-event network simulation tool OMNeT++ [11]. The achieved clock synchronization is compared against standard PTP end-systems and evaluated regarding two metrics, accuracy as average mean and jitter as the standard deviation of the clock offset. The proposed multi-domain design is able to preserve microsecond precision despite the existence of network failures. The contributions of this paper are:

- A fault-tolerant PTP end-system design that supports multiple synchronization domains.
- A timed observation window mechanism that aims to increase security by filtering received frames.
- A comparative analysis of clock synchronization quality in different test-case scenarios with faults.

The remainder of this paper is structured in 6 sections:

[§]These authors contributed equally to this work.

Section II presents the fundamental concepts of PTP and fault-tolerant synchronization and introduces the problem statement. Section III discusses the related work in PTP security and fault tolerance. Section IV presents the proposed multi-domain end-system architecture and discusses the required network topology. Section V evaluates the proposed multi-domain design and compares its performance against the standard PTP mechanisms by simulating different test-cases with synthetic scenarios. Section VI provides a discussion on the safety and security implications of the proposed multi-domain aggregation mechanism. Section VII presents the planned future extensions of this work. Section VIII summarizes the presented work and concludes the paper.

II. Background

A. Fault-Tolerant Clock Synchronization

Precise and fault-tolerant time synchronization is an operational requirement of distributed safety-critical real-time systems, such as those found in aerospace and automotive industry. Redundancy is the key to tolerate Byzantine faults in these systems, as any master clock can exhibit arbitrary behaviour and provide false readings of its local clock to connected systems. Consequently, slave clocks can misinterpret this information either because accurate convergence algorithms have not been implemented or simply because the in-place redundancy is not sufficient. It can lead to drift in the relative clock offset of the network-wide time base.

This effect has been described in research [12], [13], where the authors have analyzed the need for $3f+1$ nodes available in a distributed system that can tolerate f faults and have provided static bounds for different convergence algorithms. The most predominant algorithm of these is the Fault-Tolerant Average (FTA), which was first introduced in [14] and is incorporated in the fault-tolerant clock synchronization of TTEthernet [15] that is now part of the aerospace standard AS6802 [16]. In this work, we try to incorporate FTA principles in PTP and evaluate its performance in TSN networks as a means to provide fault-tolerant multi-domain clock synchronization.

B. IEEE 1588-2019 Precise Time Protocol

PTP is a hierarchical clock synchronization protocol based on a periodic exchange of Ethernet frames that estimates the clock offset between end-system ports configured as slaves and masters [17]. Typically, a PTP stack is assigned to each PTP port and is responsible for executing the protocol. A mechanism, called a clock servo, is responsible for correcting the device clock using a proportional-integral filter [18]. The PTP stack on a slave port calculates the time difference from a master by collecting four timestamps using four respective frames:

1) SYNC, from master to slave
2) FOLLOW_UP, from master to slave
3) DELAY_REQ, from slave to master
4) DELAY_REPLY, from master to slave

Moreover, precise time-stamping of the received/sent frames is a crucial part of the protocol, as it directly influences the precision of the estimated clock offset. Select hardware units can be used for this purpose [19] . In the rest of this work, we assume that such time-stamping units are available in all end-systems.

PTP allows for multiple masters to exist, but only one master's synchronization frames are used to calibrate an end system's internal clock at each given synchronization cycle. This selection is made using the best master clock algorithm (BMCA). The BMCA works by comparing an arbitary value, which represents the remote clock quality, connected network end-systems advertised that in dedicated periodic frames called ANNOUNCE frames. From this information it derives the best clock and then it compares that to the quality of its local clock to determine its role as a master or a slave.

The IEEE-1588-2019 standard adds several security features to PTP [9]. Most notably, it adds support for multiple types of authenticated encryption, addressing many of the security concerns that were present in its predecessor, IEEE-1588-2008 [7]. However, none of the introduced features protects against delay attacks, nor do they consider faulty master nodes (e.g., compromised by a malicious party). A malicious master node might try to influence the system time by announcing high accuracy during the BMCA, subsequently moving the time window once it has been elected. Delay attacks assume that an attacker can control a link, and might delay messages for an indefinite amount of time. To mitigate the impact of such attacks, the IEEE-1588-2019 standard only includes two recommendations: to deploy redundant master clocks; and to deploy redundant network topologies. The first recommendation works without any alterations to the protocol. As the PTP protocol is a distributed algorithm, it will eventually select one of the redundant master clocks if the primary one fails; however, this can introduce significant time overhead that leads to jitter. The second recommendation stands out: a PTP system distils a logical minimum spanning tree topology with the elected master clock as a root, and all slaves (i.e., consumers of the synchronization signal) as leaves. A minimum spanning tree does not allow multiple paths between any two nodes to exist by its very definition. The solution to this is to run multiple PTP domains in parallel, ensuring that they choose different physical network paths for their tree topology. A PTP domain is a numerical identifier included in every protocol message. It allows multiple PTP systems to operate on one network without interfering with other PTP systems.

A multi-domain setup combines neatly with the first recommendation of using multiple master clocks. With multiple parallel domains, every slave system needs to execute a deterministic voting algorithm to arrive at the same approximate time. However, the IEEE-1588-2019 standard does not recommend any voting algorithms. Additionally, it is left unclear what the performance impact and effectiveness of these measures will be. Therefore, we attempt to fill this knowledge gap by analyzing two voting algorithms' performance in a simulated PTP system. Further, we explore the impact of link failures on timing accuracy during the execution of a PTP system both with and without redundancy in place.

III. Related Work

In the broad spectrum of network attacks related to PTP, disrupting the synchronization is the primary goal. Lack of message authentication is one of the main attack vectors to break master-slave synchronization. The authors of [8] analyze the security risks associated with PTP by building a testbed that shows synchronization disruption between PTP devices. The tests conducted include master spoof attacks (spoofing ANNOUNCE and SYNC packets), ANNOUNCE DoS attacks (spamming target slave) and master clock takeover attacks. Similarly, Lisova [20] presents a threat model that shows an attack classification that lists several PTP clock synchronization attacks (e.g. replay and delay attacks, flooding/DoS) that target availability among other factors. Lisova proposes a distributed monitoring strategy to detect if an attacker is affecting clock synchronization. While both studies [8] [20] point out existing threats to availability, the current work provides a fault-tolerant design to guarantee availability.

To tackle some of the attacks mentioned earlier, IPSec and MACSec have already been analyzed for time synchronization [21] to provide authentication, encryption, and confidentiality. However, neither provide any availability guarantees or fault tolerance against a compromised endpoint or delay attacks. We consider IPSec and MACsec complementary to the fault tolerance algorithms and mechanism discussed in this work.

In 2014, Mizrahi published an informational Request For Comments (RFC) with the requirements to secure time protocols in packet-switched networks [22]. The document presents a threat model and threat analysis that lists several attack types such as packet manipulation, spoofing or replay attacks. It focuses on listing minimum security requirements such as authentication, authorization, confidentiality. While these requirements could create a security basis for next versions of time synchronization protocols, they do not guarantee availability. Additionally, the document briefly references a few mechanisms to protect against delay attacks or attacks that degrade clock accuracy, such as using of multiple paths [23]. This RFC also proposes that outliers in received time values should be considered erroneous and be ignored. The current study aims to fill the gap of fault tolerance, resilience and availability that the RFC does not cover. Specifically, it presents an implementation and evaluates its resilience to faults.

Mizrahi presents the concept of *slave diversity* [23] to obtain high clock accuracy and reduce time error using multiple paths. Similarly, Shipiner et al. present a multi-path approach [24] that evaluates path diversity. While both studies demonstrate the applicability of multiple path time synchronization, there are significant differences with this work. First, Mizrahi [23] does not tackle the master redundancy and availability features into fault tolerance and Shipiner et al. [24] do not provide simulation and performance results. In contrast, this work, uses different PTP domains with multiple masters to guarantee availability and evaluates the fault-tolerance in simulation.

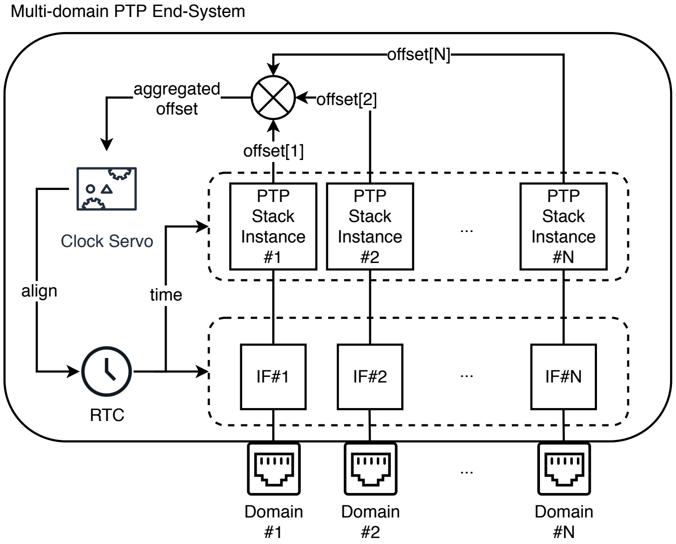


Fig. 1: Extended PTP end-system architecture to support multi-domain aggregation. Each domain uses a separate network interface and PTP stack. The calculated offsets are fed into an aggregation function, which corrects the clock.

IV. Multi-domain Node and Algorithm Design

Our proposed approach consists of multiple design elements and considerations spread over multiple layers. Firstly, we introduce a redundant variant of a typical PTP node. A node's ability to interact with singular (i.e., non-redundant) nodes is preserved, leaving room for hybrid PTP systems. Secondly, we discuss network topology requirements that should be taken into account when designing redundant PTP systems. Finally, we describe the implemented convergence algorithms.

The design proposed in this section aims to mitigate link failures and protect against Byzantine actors on the network, but it does not guarantee the communicated messages' integrity or authenticity. It is intended to complement existing resilience and security features proposed by the IEEE-1588-2019 standard, which provide these properties.

A. Node Architecture

A redundant PTP node has to support running PTP on n domains at once. To this end, we design a node architecture that maintains n parallel PTP stacks and aggregates their computed offsets. As is usual for Byzantine fault-tolerant systems, to protect against f faults, n should be picked as $n = 3f + 1$. Figure 1 presents the design of the proposed node. Each PTP stack is assigned an individual network interface port and executes isolated from the others. Using only one network interface is possible, but it would turn this into a bottleneck and the weakest link for each node. If the node is a slave, each stack periodically receives PTP messages that have to be aggregated somehow. Each stack distils an offset from the incoming messages. The calculated offset is combined with the latest PTP frame ingress timestamps as a tuple and fed into a convergence algorithm. If the node is a master node, it simply has to transmit PTP messages on every domain.

The convergence algorithm aggregates the most recently received offsets for each domain within an observation window, and produces a single aggregated offset correction for the real-time clock (RTC). This convergence algorithm is transparent to the PTP stacks, the clock servo, and any applications depending on the synchronized time of the RTC.

B. Network Topology

To effectively mitigate link/node failures and malicious PTP actor nodes, network paths for each domain should be entirely disjoint. Therefore, one can specify the main goal for the network topology is to introduce redundancy where possible. The observation window should be tuned according to the maximum expected latency of all the redundant domain paths. Thus, to minimize the observation window span , a design using redundant network paths should strive to preserve a symmetric topology with the same number of hops between slaves and master nodes. Further optimization on the asymmetry of links has been investigated by [25], [26]. Note that while a fully symmetric topology describes an ideal situation, it is not an explicit requirement. A symmetric topology allows for balanced network delays with equal worst-case end-to-end latency (WCEL), and thus it is hypothesized to lead to better convergence algorithm performance. In the remainder of this work, we thus assume a fully symmetric topology to explore the ideal case.

C. Convergence Algorithms

The convergence algorithm is run on each PTP slave node individually and takes as inputs a collection of latest observed offsets from each domain PTP stack. We implement two different convergence algorithms for evaluation. The first offset aggregation algorithm is a simple averaging function (AVG) over the available offsets. The second algorithm implements a Byzantine Fault Tolerant approach (FTA) for clock synchronization.

1) Observation Window Filtering: Figure 2 illustrates how individual PTP frames from the different PTP stacks are converged by initiating separate observation windows. Each received SYNC, or FOLLOW_UP frame initiates a new observation window based on the ingress timestamp over which the convergence algorithm operates. Only frames within an observation window time are taken into account to calculate the converged clock offset for that specific point in time. The duration of the observation window, controls the accepted time difference threshold of the received master frame timestamps from the last received PTP frame timestamp. This parameter should be tuned proportionally to the WCEL that the PTP master frames can experience, i.e. the longest path delay between a redundant master and the receiving slave.

The windowed decision algorithm is listed in Algorithm 1. This algorithm takes a new (incoming) offset o from its local clock as input, together with its ingress time i and PTP domain d. The algorithm's output is an approximate offset and ingress time, which can be used by the clock servo to correct the RTC. First, it stores the tuple (o,i) in a table structure using the domain as an index, ensuring that only one offset per domain is considered. After this, the table S is filtered to S', excluding offsets that were not received within a given delta `WINDOW` from the new ingress timestamp. Then, the ingress of all offsets in S' are averaged to i_a, and an approximate offset o_a is calculated using either the FTA or the AVG algorithm.

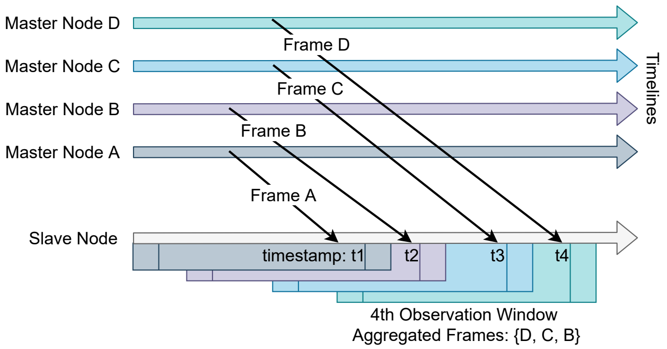


Fig. 2: Observation windows are generated by new SYNC/-FOLLOW_UP frames. Received frames that are within the time window are used in the aggregated offset calculation.

Algorithm 1 Windowed Decision Algorithm

```
procedure WindowedDecision($(o,i,d)$)    ▷ Executes a windowed decision algorithm using the latest received timestamps
State: $S$    ▷ A table $d \rightarrow (o_d, i_d)$ mapping all domains $d \in D$ to (offset, ingress) tuples
    $S[d] \leftarrow (o,i)$
    $S' \leftarrow \{x \rightarrow (o_x, i_x) \in S \text{ where } |i - i_x| \leq WINDOW\}$
    $i_a \leftarrow \begin{cases} 0 \text{ if } |S'| = 0 \\ \frac{\Sigma_{x \in S'} i_x}{|S'|} \text{ otherwise} \end{cases}$
    $o_a \leftarrow \text{FTA}(S') \text{ or } \text{AVG}(S')$
    return $(o_a, i_a)$
end procedure
```

2) Averaging Algorithm (AVG): The AVG consists of a simple averaging function that extracts all offsets from the given map and returns the average of these, or 0 if there are no offsets.

3) Fault Tolerant averaging Algorithm (FTA): The FTA [14] is an algorithm that provides bounded clock synchronization even in the presence of faulty and possibly malicious master clocks (see also Section II). Algorithm 2 describes the implemented FTA algorithm.

In the general case where k faults should be tolerated, this algorithm drops the earliest and last k offsets and averages the remaining offsets. First, usable offsets are extracted from the given map structure S', and special assignments are made for the $2k$ most extreme offsets. Then, it distinguishes 3 cases: firstly, if there is only one offset, we return that offset; secondly, if there are only two offsets, their average is returned, and finally, if there are three or more offsets, it drops the extremes

Algorithm 2 Fault Tolerant Algorithm

```
procedure FTA($S$)        ▷ Executes a fault-tolerant convergence algorithm over a set of offsets
    if $|S| = 0$ then
        return 0
    end if
    $O = \{o_x | x \in S\}$
    $o_{min} \leftarrow k$ earliest offsets in $O$
    $o_{max} \leftarrow k$ latest offsets in $O$
    if $|O| = 1$ then
        return $o_{min}$
    else if $|O| = 2$ then
        return $\frac{o_{min}+o_{max}}{2}$
    else if $|O| \geq 3$ then
        $O' \leftarrow O \setminus \{o_{min}, o_{max}\}$
        return $\frac{\Sigma_{x \in O'} x}{|O'|}$
    end if
end procedure
```

and returns the average of the remaining offsets.

The first two cases will usually only trigger if there are remote failures, and the system does not receive enough offsets. In this case, the failures are regarded as faulty nodes, thereby exceeding the number of tolerated faults, and the most we can do is a best-effort execution of the algorithm. The third case covers the standard execution of the algorithm. By dropping the $2k$ outer offsets, adversaries are forced to operate within a limited time offset range. By taking the average of the remaining offsets, adversaries would have to control more master nodes than our model tolerates to have a considerable effect on the aggregated offset. For a formal proof, we refer the interested reader to [14], [12].

V. Evaluation

To demonstrate the fault-tolerance of the proposed redundant PTP scheme and evaluate the synchronization quality, we generate two test-case network topologies [1]. These topologies are simulated witin the OMNeT++-4.6 [11] discrete-event network simulator using our extended version [2] of a PTP simulation library named LibPTP [27]. LibPTP [28] is a complete simulation framework for OMNeT++ that allows the simulation of standard PTP devices. To the RTC oscillator noise and yield more realistic clock drift results, we utilize a Power-law noise library (LibPLN [29]) as described in the LibPTP documentation [27]. All experiments are done on a 64-bit i7-7700HQ CPU system running at 2.8 GHz with 32GB RAM.

A. Simulation parameters

The presented experiments are based on the following assumptions. Firstly, we assume that every node has multiple network interfaces, one for each domain, which is in line with the standard's recommendations, where it is advised that each domain operates over a separate network interface. Secondly, to optimize the simulation time and isolate the PTP evaluation, we assume that the network is used exclusively by PTP, so no other network traffic is simulated in the experiments. Empirically, we assume that every link has a bit-rate of 1 Gbps and is 1 meter long. Finally, every PTP stack uses the recommended gPTP profile for TSN [30] as shown in Table I and a peer-to-peer (P2P) delay mechanism.

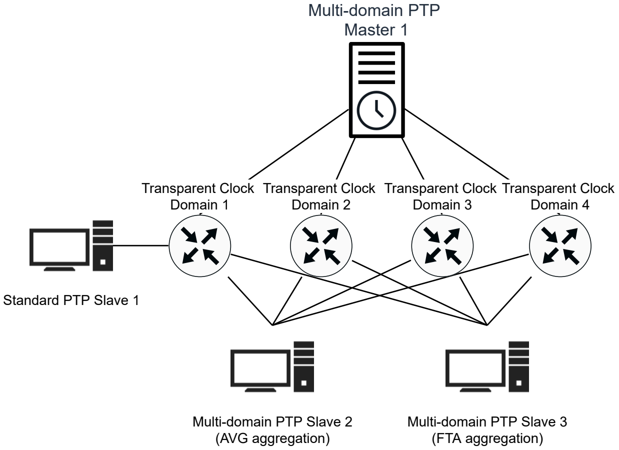

Multi-domain PTP Master 1
Transparent Clock Domain 1
Transparent Clock Domain 2
Transparent Clock Domain 3
Transparent Clock Domain 4
Standard PTP Slave 1
Multi-domain PTP Slave 2 (AVG aggregation)
Multi-domain PTP Slave 3 (FTA aggregation)

Fig. 3: First test-case network topology of single multi-domain PTP master on four isolated redundant domain paths. The domains are isolated using four different switches.

TABLE I: PTP port profile options. Values correspond to the interval of the respective messages in seconds and are represented as powers of two.

Parameter	Value
logAnnounceInterval	1
announceReceiptTimeout	3
logSyncInterval	-3
logMinDelayReqInterval	-3
logMinDelayReqInterval	-3

B. Test-case 1: Single PTP master on four redundant domains

This experiment aims to evaluate the stability of the proposed multi-domain aggregation scheme using the custom design of Figure 1 for both master and slave nodes. We generate a synthetic topology with three nodes and four switches as shown in Figure 3. A single multi-domain PTP master is connected to four redundant transparent clock nodes over four different domains. We integrate three different types of PTP slaves in the network: (A) a standard PTP slave connected only to the first transparent clock switch (domain), (B) a multi-domain PTP slave that uses the AVG algorithm and is connected to all domains and (C) a multi-domain PTP slave that uses the proposed FTA algorithm and connects to all domains.

We evaluate the synchronization quality in terms of average mean clock offset and standard deviation using a synthetic

[1] https://github.com/dtu-ese/ptp_multidomain

[2] https://github.com/dtu-ese/libPTP

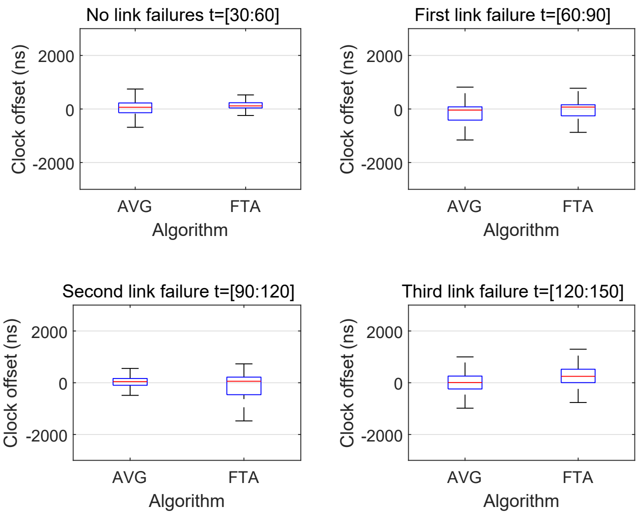


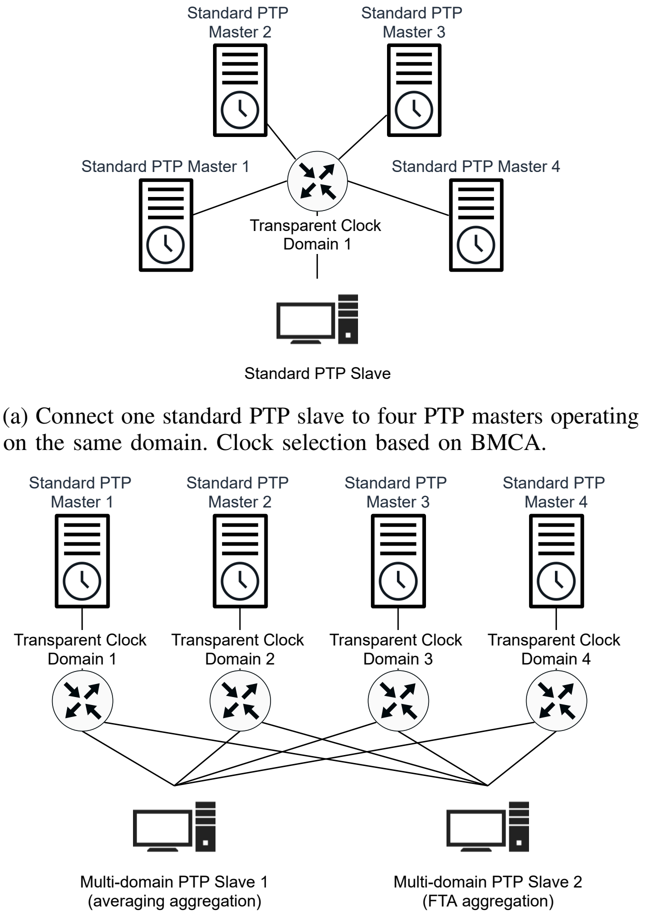
Fig. 4: Comparison of the mean clock offset and std. deviation measurements through the link failures of the experimental test-case 1 (see Section V-B) with topology from Figure 3.

scenario. We simulate a simple scenario of consecutive link failures where at 60 seconds the first link between Master 1 and Transparent Clock 1 is disconnected. The rest of the links between Master 1 and the transparent clocks are disconnected/fail similarly in intervals of 30 seconds. We simulate the scenario for a total run-time of 180 seconds.

Figure 4 compares the measured mean clock offset and jitter of the two clock servo aggregation methods (AVG and FTA). Although the mean of the AVG and FTA aggregation methods are similar when no failures occur, we measure significantly less jitter using FTA throughout the experiment's run-time, resulting in more predictable clock synchronization. This is likely due to the nature of the FTA: outliers are discarded , ensuring that the system will take the average of the most consistent master clocks. If there are some master clocks that drift at different rates, or if these clock oscillators are very noisy, then it is likely that they are often discarded for the aggregated timestamp.

C. Test-case 2: Four PTP masters on four redundant domains

For the second test-case, we generate and simulate two network topologies comparing the standard BMCA against the proposed multi-domain scheme. The first topology (see Figure 5a) has four PTP master capable standard nodes and a standard node that is configured as a PTP slave. All nodes operate over the same domain and are connected through a transparent clock switch in a star topology. The second topology (see Figure 5b) has four standard PTP masters connected and two redundant PTP slave nodes. The PTP masters operate over four different domains and are respectively connected to four different transparent clock switches. For simplicity, we assume that individual PTP master node clocks are synchronized to each other in order for the observation window to use all available domains. This requirement is further discussed in Section VI. PTP slave nodes 1 and 2 use respectively, the multi-domain aggregation methods described in Section IV. We evaluate the performance of the synchronization by simulating two synthetic scenarios.


Standard PTP Master 2
Standard PTP Master 3
Standard PTP Master 1
Standard PTP Master 4
Transparent Clock Domain 1
Standard PTP Slave
Standard PTP Master 1
Standard PTP Master 2
Standard PTP Master 3
Standard PTP Master 4
Transparent Clock Domain 1
Transparent Clock Domain 2
Transparent Clock Domain 3
Transparent Clock Domain 4
Multi-domain PTP Slave 1 (averaging aggregation)
Multi-domain PTP Slave 2 (FTA aggregation)

(a) Connect one standard PTP slave to four PTP masters operating on the same domain. Clock selection based on BMCA.

(b) Two multi-domain PTP slaves connected to four PTP masters operating on separate domains. Clock offset calculation uses multi-domain aggregation.

Fig. 5: Second test-case parallel network topologies evaluation.

1) Link/node failure scenario: In the first scenario, each of the PTP masters fails in sequence every 30 seconds after the first minute of stable operation. This scenario covers a variety of real-life failures such as device failures, cable failures or denial-of-service attacks. We simulate the experiment for a total run-time of 180 seconds.

Figure 6 presents the mean time difference of the three PTP slave nodes and compares the upper/lower bounds of the three PTP slave nodes. We observe that in contrast to Test-case 1, the standard PTP slave node can stay synchronized to the master through the consecutive link failures as it can now select a new master, from each operating domain, after each link failure. However, BMCA suffers from significant synchronization drift of more than 2 μs. The FTA and the AVG aggregation manage to achieve better clock synchronization accuracy with tighter bounds than the standard BCMA during

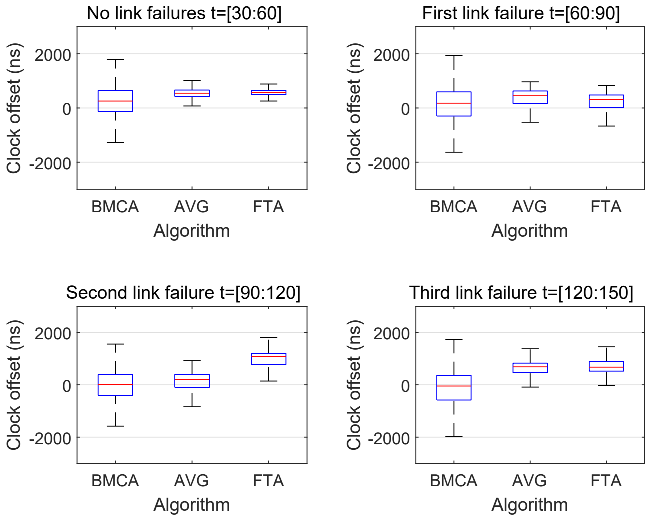


Fig. 6: Comparison of the mean clock offset and std. deviation measurements through the link failures of the experimental test-case 2 (Section V-C1) with topology from Figure 5.

the first two link failures. As more links fail this difference between the methodologies is normalized because fewer nodes are available to aggregate.

2) Malicious PTP master scenario: In this scenario, we investigate the effects of a malicious PTP master clock that tries to offset the synchronized network time. The malicious end-system is connected to the network at a specific point in time and advertises that it has a higher clock quality that the existing master clocks. We emulate this scenario by simulating the instantaneous connection of a new PTP master with higher quality clock attributes after one minute of run-time at the first switch. The malicious master has its local clock offset by 100 μs than the existing masters. Due the implemented observation window's properties, a malicious master must be carefully implemented so that its local clock offset is within the observation window's bounds.

We measure this attack's effects on the clock synchronization precision of the topology's three PTP slaves relative to node Master 1. Figure 7 presents the measured results of the time-difference for the three PTP slaves. The top plot corresponds to the measurements taken from the Standard BMCA slave shown in Figure 5a. In comparison, the bottom plot presents the measurements from the multi-domain slaves shown in Figure 5b. We run the experiment for 120 seconds of simulation time.

In the standard PTP topology 5a, the newly connected malicious master is quickly elected as the best clock by the BMCA. We note a significant initial drift of the PTP slave relative to Master 1 after which the network is synchronized to the time of the malicious master clock. In the redundant PTP topology 5b, the connection of the malicious master cannot influence the independent masters as they operate in different domains. The simple approach of averaging the aggregated multi-domain master clocks is not sufficient as it is easily disturbed by the malicious clock's offset. In this scenario, the FTA proves to be the most resilient as the malicious master's relative clock offset is discarded according to Algorithm 2.

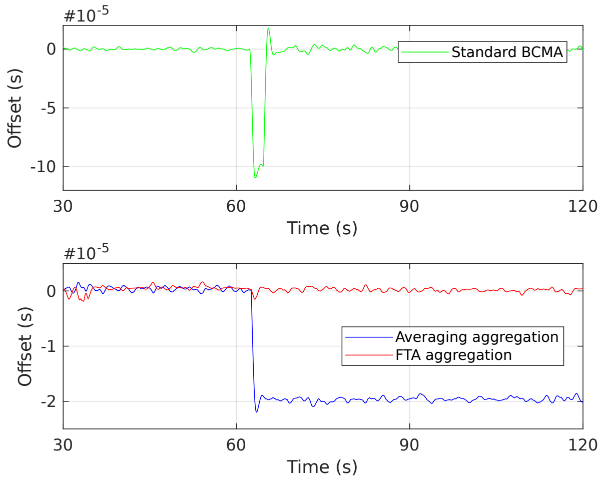


Fig. 7: Measured PTP-Slave clock offset relative to Master 1 in the test-case scenario of a new malicious PTP master node connection at t=60s (Section V-C2).

VI. Discussion

In the evaluated test-cases, we experimentally showed that a multi-domain approach could guarantee synchronized network time availability despite network failures and malicious actions.

The platform designer has to guarantee that the PTP stack processes are isolated and cannot affect each other if the security of one PTP stack is compromised. This can be achieved using specialized hardware or sandboxing techniques such as virtualization. Considering the capabilities of modern industrial computing systems [31], the software cost for running the redundant PTP stacks in-parallel is minimal, especially if the proposed design is implemented completely in software. Preliminary results show that the CPU overhead generated by the PTP stack is less than 1% of the available computing resources. Nevertheless, the system designer should consider the additional cost for the redundant network topology based on the safety requirements of the application, as there is a significant cost increase in the number of links and switches.

The results showed that the FTA convergence algorithm could mitigate against link or node failures, as well as a compromised master node broadcasting incorrect timestamps. This work illustrates the importance of a fault-tolerant method of converging the calculated offset from the multiple PTP domains. It is worth noting that although the averaging aggregation performed as well as the FTA method, it was easily influenced by a malicious node and failed to provide secure synchronization. While our design does not enforce authentication and integrity of PTP messages by itself, the FTA algorithm leaves very little room for tampered messages, as it discards everything outside of a margin known to have a majority of correct offsets. What this approach does inherently provide is protection against various forms of DoS, timing, and delay attacks where the number of affected links/nodes is less than k. As already noted in Section IV, this can be combined with the security measures proposed in IEEE-1588 (2019) to further harden the security by providing authenticity, confidentiality and integrity of messages. Thus, the combined application of the measures proposed in this work and the standardized security measures results in a secure PTP system that in addition to the standardized measures is difficult to disrupt with DoS and timing attacks.

Finally, although the proposed multi-domain PTP end-system scheme was tested with both master and slave roles, its functionality is based on the assumption that the redundant master clocks of each separate domains are synchronized to each other. This assumption is easily achievable using the proposed multi-domain PTP end-system design (see Figure 1), however standard PTP master clocks on separate devices require an external fault-tolerant mechanism of clock synchronization. One possible solution to this would be to use dual roles for master nodes, were on specific domains they would act as slaves to each other and other domains as masters in an interleaved scheme. It is hypothesized that the standard PTP boundary clock component can support this dual role functionality, but its implementation in a multi-domain network topology requires further investigation.

VII. Future Work

As future work, we plan to explore the implementation and characterization of boundary clocks as a mechanism to enable standard PTP master clock synchronization in redundant domains. Additionally, we plan to extend the evaluated scenarios and investigate different types of attacks on PTP, such as frame spoofing. This will allow us to characterize further the proposed multi-domain design performance and identify its tuning parameters.

Moreover, one can think of a scenario where only a limited subset of all nodes are connected to multiple domains. This raises questions such as how many multi-domain nodes are necessary to meet a certain required timing accuracy? For this, we plan to explore the integration of the proposed design in boundary clocks that are connected to multiple domains, each maintaining slave clocks connected to only one of these domains.

VIII. Conclusion

The presented work investigated the requirements for fault-tolerance in TSN clock synchronization and proposed a PTP end-system design that supports multi-domain aggregation. The proposed design implements isolated PTP stacks that use an FTA-based aggregation mechanism to correct the clock servo. This is combined with a time-based observation window for additional security. The multi-domain PTP end-system was evaluated and compared against standard PTP nodes in two scenarios with emulated link failures and possible malicious PTP masters. Overall, this work illustrated empirically the necessity for fault-tolerance in PTP and multi-domain aggregation design that manages to overcome network faults.

Acknowledgment

This is work was part of the Fog Computing for Robotics and Industrial Automation (FORA) European Training Network (ETN) funded by the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 764785.

References

[1] T. Pereira, L. Barreto, and A. Amaral, "Network and information security challenges within Industry 4.0 paradigm," *Procedia Manufacturing*, vol. 13, 2017.

[2] I. Studnia, V. Nicomette, E. Alata, Y. Deswarte, M. Kaaniche, and Y. Laarouchi, "Survey on security threats and protection mechanisms in embedded automotive networks," *Proc. DSN*, 2013.

[3] S. S. Craciunas and R. S. Oliver, "An overview of scheduling mechanisms for time-sensitive networks," *Proceedings of the Real-time summer school LÉcole dÉté Temps Réel (ETR)*, pp. 1551–3203, 2017.

[4] E. Kyriakakis, J. Sparsø, P. Puschner, and M. Schoeberl, "Synchronizing real-time tasks in time-aware networks: Work-in-progress," in *2020 International Conference on Embedded Software (EMSOFT)*. IEEE, 2020, pp. 15–17.

[5] *Official Website of the 802.1 Time-Sensitive Networking Task Group*, http://www.ieee802.org/1/pages/tsn.html, IEEE Std., 2016, accessed: 17.12.2020.

[6] *802.1AS-Rev - Timing and Synchronization for Time-Sensitive Applications*, http://www.ieee802.org/1/pages/802.1AS-rev.html, IEEE Std., 2016, accessed: 17.12.2020.

[7] *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE Std., 2008.

[8] C. DeCusatis, R. M. Lynch, W. Kluge, J. Houston, P. Wojciak, and S. Guendert, "Impact of cyberattacks on precision time protocol," *IEEE Transactions on Instrumentation and Measurement*, 2019.

[9] *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE Std., 2020.

[10] S. Bogomolov, C. Herrera, and W. Steiner, "Verification of fault-tolerant clock synchronization algorithms." in *ARCH@ CPSWeek*, 2016, pp. 36–41.

[11] A. Varga and R. Hornig, "An Overview of the OMNeT++ Simulation Environment," in *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, ser. Simutools '08. Brussels, BEL: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[12] H. Kopetz, A. Damm, C. Koza, M. Mulazzani, W. Schwabl, C. Senft, and R. Zainlinger, "Distributed fault-tolerant real-time systems: The mars approach," *IEEE Micro*, vol. 9, no. 1, pp. 25–40, 1989.

[13] P. Ramanathan, K. G. Shin, and R. W. Butler, "Fault-tolerant clock synchronization in distributed systems," *Computer*, vol. 23, no. 10, pp. 33–42, 1990.

[14] D. Dolev, N. A. Lynch, S. S. Pinter, E. W. Stark, and W. E. Weihl, "Reaching approximate agreement in the presence of faults," *Journal of the ACM (JACM)*, vol. 33, no. 3, pp. 499–516, 1986.

[15] W. Steiner and B. Dutertre, "The TTEthernet synchronisation protocols and their formal verification," *International Journal of Critical Computer-Based Systems 17*, vol. 4, no. 3, pp. 280–300, 2013.

[16] TTTech, *AS6802: Time-Triggered Ethernet*, SAE International Std., 2011.

[17] J. C. Eidson, *Measurement, control, and communication using IEEE 1588*. Springer Science & Business Media, 2006.

[18] G. Giorgi and C. Narduzzi, "Modeling and simulation analysis of PTP clock servo," in *2007 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, 2007.

[19] E. Kyriakakis, J. Sparsø, and M. Schoeberl, "Hardware Assisted Clock Synchronization with the IEEE 1588-2008 Precision Time Protocol," in *Proceedings of the 26th International Conference on Real-Time Networks and Systems*, ser. RTNS '18. New York, NY, USA: ACM, 2018, pp. 51–60. [Online]. Available: http://doi.acm.org.proxy.findit.dtu.dk/10.1145/3273905.3273920

[20] E. Lisova, "Monitoring for securing clock synchronization," Ph.D. dissertation, Mälardalen University, 2018.

[21] T. Mizrahi, “Time synchronization security using IPsec and MACsec,” in *2011 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, 2011, pp. 38–43.

[22] T. Mizrahi, *Security Requirements of Time Protocols in Packet Switched Networks*, RFC 7384, Std. 7384, Oct. 2014. [Online]. Available: https://rfc-editor.org/rfc/rfc7384.txt

[23] T. Mizrahi, “Slave diversity: Using multiple paths to improve the accuracy of clock synchronization protocols,” in *2012 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication Proceedings*, 2012, pp. 1–6.

[24] A. Shpiner, Y. Revah, and T. Mizrahi, “Multi-path time protocols,” in *2013 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS) Proceedings*, 2013, pp. 1–6.

[25] O. Gurewitz and M. Sidi, “Estimating one-way delays from cyclic-path delay measurements,” in *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213)*, vol. 2. IEEE, 2001, pp. 1038–1044.

[26] S. Lee, “An enhanced IEEE 1588 time synchronization algorithm for asymmetric communication link using block burst transmission,” *IEEE communications letters*, vol. 12, no. 9, pp. 687–689, 2008.

[27] W. Wallner, “Simulation of time-synchronized networks using ieee 1588-2008,” Ph.D. dissertation, Wien, 2016.

[28] W. Wallner. (2016) LibPTP: A Library for PTP Simulation. https://github.com/ptp-sim/libPTP.

[29] W. Wallner. (2016) LibPLN: A Library for Efficient Powerlaw Noise Generation. https://github.com/ptp-sim/libPLN.

[30] K. Sridharan, K. Goossens, N. Concer, and H. B. Vermeulen, “Investigation of time-synchronization over ethernet in-vehicle networks for automotive applications,” Master’s thesis, Eindhoven: Eindhoven University of Technology, 2015.

[31] *Intel’s Fog Reference Design Overview*, Intel, April 2018. [Online]. Available: https://www.intel.com/content/www/us/en/internet-of-things/fog-reference-design-overview.html