# Temporal Replication of Messages for Adaptive Systems using a Holistic Approach

Alberto Ballesteros, Manuel Barranco, Sergi Arguimbau, Marc Costa and Julián Proenza
*DMI - Universitat Illes Balears, Palma, Spain*
{a.ballesteros, manuel.barranco, sergi.arguimbau, marc.costa.marquez, julian.proenza}@uib.es

***Abstract*—Critical Adaptive Distributed Embedded Systems (ADES) must meet high real-time and dependability requirements, while autonomously rearranging themselves to operate in dynamic operational contexts. The DFT4FTT project proposes a self-reconfigurable complete infrastructure, whose different architectural levels provide a set of real-time (RT), fault-tolerance (FT) and flexibility mechanisms that collaborate to adequately support critical ADESs. To efficiently tolerate transient faults in the network of an ADES, this paper describes our ongoing work on providing a dynamic temporal replication of messages that takes into account all the DFT4FTT fault-tolerance mechanisms from a holistic point of view.**

## I. Introduction

A critical Adaptive Distributed Embedded Systems (ADES), e.g. an autonomous vehicle, must automatically rearrange itself to adequate its operation as the *operational context* unpredictability changes. The *operational context* encompasses both: (1) the *operational requirements*, which include its functionalities, its RT guarantees, and the reliability it has to exhibit; and (2) the *operational conditions*, i.e. the environment and the system itself (which can change due to faults).

Adaptivity enables taking full advantage of the resources so as to not over-dimension the system to face unpredictable operational contexts, e.g. by devoting a given resource to different purposes as the functionality changes. Moreover, by orchestrating the resources of the different levels of the system architecture from a holistic point of view, it is possible to use them in an even more efficient manner. In particular this applies to the FT mechanisms, which can dynamically change from a system-wide perspective to tolerate faults more efficiently than classic static FT mechanisms can do.

To support RT highly-reliable ADESs while taking into account the above-mentioned aspects, the *Dynamic Fault Tolerance for the Flexible Time-Triggered Ethernet* (DFT4FTT) project [1] is devoted to providing a *complete infrastructure* with advanced FT capabilities. By a complete infrastructure we refer to a set of interrelated hardware and software components (the architecture) together with the necessary in-built mechanisms both at the node and at the network level.

At the node level, DFT4FTT provides high reliability by means of active replication with majority voting. Each critical task is executed in parallel in several nodes of the ADES (called *Computational Nodes*, CNs). To provide flexibility at this level, we proposed a centralized architecture in which a so-called *Node Manager* (NM) can reconfigure at runtime the allocation and replication of tasks into the CNs [1].

At the network level, DFT4FTT relies on the *Flexible Time-Triggered Replicated Star* (FTTRS) [2], a switched-Ethernet implementation of the Flexible Time-Triggered (FTT) communication paradigm. FTT makes it possible for the nodes of an ADES to exchange traffic with RT guarantees. Moreover, FTT provides *full flexibility* in the communications to support periodic and aperiodic traffic with different RT requirements, as well as changes in these requirements at runtime. To attain high reliability FTTRS both replicates the network to tolerate permanent network faults, and proactively retransmits the critical messages to tolerate transient ones.

However the temporal replication of messages in DF4FTT was static so far. Such static replication can be inefficient, or even ineffective, when facing the changing operational conditions in which ADESs operate. Thus, in this paper we propose how to make the proactive retransmission mechanism of DFT4FTT dynamic. Specifically, we propose to change at runtime the number of message replicas to be sent (referred to as $k$ hereafter) depending on the current operational context, which also includes the available FT mechanisms at the different levels of the architecture.

It is important to highlight that, although all this work has been developed in the scope of the DFT4FTT project, the ideas herein presented are quite generic and, thus, they can be applied in different communication subsystems.

## II. Related Work

During the last decades several architectures have been proposed to provide RT and FT services for the execution and/or the communication of tasks in distributed systems, e.g. MAFT, FTP-AP, Delta-4, GUARDS, EMC$^2$, DREAMS [3]. Some of them like Delta-4 and DREAMS [4] do even provide services to reallocate and reschedule tasks at runtime. However, some of these architectures require complex communication protocols at the application or transport layer to provide node FT, others require costly adhoc network topologies, while others are generic architectures not providing any specific strategy for replicating the nodes or the network.

On the other hand, several Ethernet protocols do provide some RT and/or FT properties. Some of the newer ones such as PRP, AFDX, TTEthernet and specific TSN standards can even provide zero recovery times by means of spatial redundancy,

This work was supported by project TEC2015-70313-R (Spanish *Ministerio de economía y competividad*) and by FEDER funding

e.g. TSN's IEEE 802.1CB. However, none of these protocols provide temporal replication of messages to efficiently tolerate transient link faults, i.e. at most, they use proactive message replication to send critical messages through the available redundant paths so as to tolerate permanent link faults. Moreover, they either do not support online reconfiguration or imply a latency for doing so, e.g. in the CUC/CNC approach of TSN [5], that is not adequate [2] to timely react to critical situations.

## III. Overview of DFT4FTT

As just said, DFT4FTT relies on FTTRS [2] to implement the communication subsystem. As Fig. 1 shows, the *Computational Nodes* (CNs) are interconnected by means a duplicated star to tolerate permanent network faults. Each switch embeds a replica of the FTTRS *master*, which organizes the communication in fixed-duration slots called *Elementary Cycles* (ECs). The EC starts with the master transmitting the so-called *Trigger Message* (TM), and it is divided into windows each accommodating a different kind of traffic. Transient network faults are tolerated by proactively retransmitting $k$ times in advance the critical messages [2].

At the node level, each *functionality* in the system is implemented by means of an *application* that is composed of several interconnected tasks that can be executed in parallel or sequential manner. A task is the minimum unit of computation and can be deployed in any CN. As an example, Fig. 2 shows a control application composed of three tasks: a sensing task ($S$); a triplicated control task ($C$), where each one of its replicas executes in parallel; and an actuation task ($A$).

Note that an application is executed as a set of phases involving execution of tasks and transmission of messages. Thus, a given *system configuration* basically includes the allocation and replication of tasks, the number of proactive retransmissions of critical messages ($k$), and the schedule of tasks and messages.

The *Node Manager* (NM) is the responsible for reconfiguring the system at runtime when necessary. Since it is still to be duplicated to tolerate its own faults, it is depicted as a separate component; however, we plan to embed each one of its replicas in the FTTRS switches.

## IV. Dynamic message replication

We propose to dynamically replicate the messages according to the holistic approach outlined in Sec. I. Specifically, the NM must make its decisions on how to temporally replicate messages considering the operational context from the perspective of the whole system, i.e. considering the mechanisms of the different levels of the DFT4FTT architecture that are still available to tolerate node [6] and network [2] faults. We believe that this dynamic replication strategy is more efficient than others in which the decisions on how to temporally replicate messages are taken exclusively from a network-level perspective, e.g. by considering the quality of the links [7].

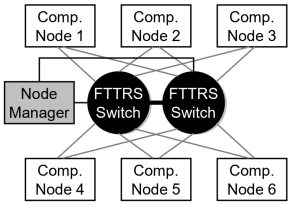


Fig. 1: System architecture.

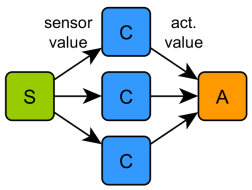


Fig. 2: App example.

Next we explain each one of the steps in which we propose to divide the dynamic replication of messages, namely: (1) detecting when the operational context changes; (2) determining, accordingly, on the specific number of message replicas for each type of message; and, finally, (3) propagating the new replication parameters to the nodes of the ADES.

### A. Detection of the need for changes

Next we outline which changes of the operational context make it necessary to update the number of message replicas to be proactively retransmitted, i.e. $k$, as well as which information can be monitored to detect these changes.

Since the system must be (re)configured in a holistic manner, which changes trigger the update of $k$ must be explained from a system-wide perspective. At system's start-up the NM sets $k$ to a conservative value, considering the different FT mechanisms of DFT4FTT, but assuming that the network has to deal with the harshest environment in which the system is going to operate. Then, the NM decides to dynamically reconfigure the system, including $k$, when it encounters the following situations. First, if the environment actually becomes more benign, then the NM can reduce $k$ to save energy, or to ease future configurations in which it could need to fit new messages (e.g. if new tasks are put into execution). In any case, $k$ should always be conservative enough to prevent the system from failing while, in the future, it reconfigures itself to increase $k$ again so as to deal with an increasingly harshly environment. In this later case, if the network has no available-enough bandwidth to increase $k$ (e.g. if it had to accommodate new tasks and their messages), then the NM would need to reconfigure the system at its other levels. This reconfiguration can consist in evicting non-critical tasks (or even reduce the number of task replicas), and thus their messages, to free bandwidth so as to accommodate higher values of $k$. In any case, the NM has to find a configuration in which the FT mechanisms at the different levels of its architecture guarantee, as a whole, the desired system reliability. Second, if the system loses (spatial) redundancy at any of its other levels due to faults - e.g. if a node, task, or link fails -, then the NM should increase $k$ taking into account the just-mentioned considerations about the available bandwidth. Analogously, if the system regains redundancy thanks to its reintegration mechanisms [6], then the NM can conservatively reduce $k$ as explained before. Third, if the operational requirements (functionality, RT guarantees, or reliability) of the system change, then the NM may need to change the set of tasks to be executed (and with them their interdependencies and messages). If so, it will need to find a new configuration (including a conservative value for $k$).

It is important to note that it may be impossible to find a new configuration that fulfills all the requirements when the operational context changes. If so, the NM needs to find a new configuration in which the system provides its services with an adequate/acceptable level of reliability (even though it is in a degraded manner from the functional point of view).

To detect changes in the environment affecting the network, the NM can use different mechanisms. First, it can use several radiation sensors to measure how harshly the environment is and then estimate, e.g. using a failure rate model like [8], the expected rate of transient link faults. Second, the NM can estimate the rate of transient faults affecting each link through which it receives messages, by using the counter of dropped incoming messages (due to errors in the channel) provided by the Ethernet card of the corresponding switch port. Third, each task within a CN periodically transmits an *I Am Alive* (IAA) message which piggybacks information contained in the TM. The NM can use the percentage of IAA omissions to estimate the probability with which the transmitted TM does not reach a CN and, thus, the rate of transient faults affecting the link through which it transmits messages to that CN. Finally, DFT4FTT includes a retransmission mechanism - called CVEP [6] - to tolerate bursts, which requires task replicas to send ACK messages. The NM can use the omissions of ACKs to estimate the rate of transient link faults as well.

To detect changes in the available spatial redundancy, the NM can use the just-mentioned mechanisms (except the first one). For instance, if a task omits its expected IAA during a given period of time, the NM will diagnose that task as permanently faulty. Conversely, if that task reintegrates and successfully transmits its IAA, then the NM will detect that the task is available again.

Finally, to detect changes in the operational requirements, the NM includes application-dependant knowledge about the environment and how the system should operate accordingly [1], e.g. in an autonomous vehicle, the NM can use this knowledge to detect different terrains and, then, determine the new operational requirements to adequately drive.

### *B. Determination of the new configuration*

To determine a new system configuration we propose that the NM searches among all the possible configurations to find a valid one, i.e. one that fulfills all the operational requirements. Some of the search techniques we are considering for this are: heuristic-based techniques like branch and bound with a greedy algorithm, metaheuristic-based techniques like Tabu search [9] and solvers like SMT solvers [10].

In particular, a valid configuration must fulfill the tasks and messages RT requirements as well as the needed reliability. Thus, the search technique must include a holistic scheduler and a reliability analyzer. In any case, the search can require more or less computation time and storage capacity. The number of configurations can be huge depending on the number of considered aspects. Moreover, the scheduling and reliability analyses of each configuration can take a non-negligible amount of time. Thus, we are assessing the performance of the above-mentioned search techniques so as to decide whether the search should be done at runtime or completely/partially pre-calculated offline.

Independently of the search techniques the NM will finally use, next we outline the strategy we propose to decide how many replicas should be proactively retransmitted for each type of message. In this sense, we differentiate among *data messages*, the *Trigger Message* (TM) and *control messages*.

Data messages are used by CNs to transmit the application-level data among tasks. Of those messages, we propose to temporally replicate only the critical ones, e.g. the ones that replicated tasks use to exchange the data they need to reliably vote on. To reduce the complexity of finding an adequate system configuration, the NM must calculate a conservative value of $k$ that is common to all critical data messages. However, this value should not be calculated based on the transient failure rate of the most error-prone link. This is because the cause for a link to be specially error-prone is not necessary a harshly environment, but it can also be a local mechanical/electrical defect of that link. Thus, the common value of $k$ should be based on the radiation sensors' measurements or on a trimmed mean of the estimated link transient failure rates. Then, if a given link shows to be more error-prone than expected, the NM should find a new configuration with a higher value of $k$ for that specific link. In any case, to prevent unnecessary reconfigurations, the DFT4FTT fault diagnosis mechanisms [6] should diagnose a specially error-prone link as permanently faulty and, then, discard it from the system.

The TM triggers the communications and, thus, it is vital to transmit it in a reliable manner. However, we already constructed a reliability model of FTTRS and made sensitivity analyses to determine the impact of the replication of the TM in the system reliability [6]. The results showed that the system reliability improves as the $k$ of the TM increases; but that this improvement starts to become negligible when $k$ is increased from 3 to 4. Moreover, the difference in bandwidth usage when using 2 or 4 TM replicas is also negligible. Thus, to reduce the need for reconfigurations, we propose to use a constant value of $k = 4$ for the TM.

Control messages convey the data needed for managing the reconfigurations, e.g. the IAA message (see Sec. IV-A) and the *Master Command Message* (MCM) described later in Sec. IV-C. We propose to temporally replicate only the control messages that are critical. In principle these messages should be dynamically replicated as critical data messages are. However, to reduce the need for reconfigurations, if the difference in the bandwidth used by the message starts to be negligible when $k$ increases from $i$ to $j$, we propose to use a constant value of $k = j$ for it. For instance this is the case of the MCM, which is rarely transmitted since it is used to consistently propagate new configurations (as explained next).

### *C. Propagation of the configuration*

Once the NM has decided on a new configuration (including the values of $k$), it needs to propagate it to the CNs. In particular, the new $ks$ are then used by each CN to both transmit and

check the reception of the correct number of message replicas. Here we propose a mechanism to reliably and consistently propagate a new configuration, i.e. to guarantee that the NM and the CNs update their databases with the corresponding configuration information at an equivalent time. We describe it in terms of the databases used to store the messages' attributes; but everything said here can be applied to the other databases.

The messages database of the NM and each CN is composed of two databases: (1) the *read database*, which is used in normal operation; and (2) the *write database*, which keeps all the changes needed to step into the new configuration.

Once the NM has introduced all the changes of a new configuration into its write database, it broadcasts - in an aperiodic control message called *Master Command Message* (MCM) - the full list of changes. The MCM is proactively retransmitted several times to ensure that it is received by all the CNs and, thus, that there is no data inconsistency.

Upon the reception of the MCM, each CN updates its write database with the content of said message. After that, the NM decides in which EC the read databases should be updated with respect to the write databases. Then, the NM sends a commit order inside the TM of said EC. This order indicates that the NM is going to update its read database at the end of the EC and, thus, instructs each CN to do so with its own read database. Specifically, the NM and the CNs update their read database in a new dedicated window placed at the end of the EC, we call the *Commit Window*. Neither the NM nor the CNs are operating with their databases during this window and, thus, they can update the read database without any risk.

## V. Experimentation

Since DFT4FTT is an ongoing project, its implementation is subjected to the fully definition of its mechanisms. Thus, we have implemented all the work presented here in a real prototype; except the mechanism that make it possible for the NM to search a new configuration. This is because we are still deciding on the most suitable search algorithm to efficiently determine, among other parameters, the value of $k$ for each message. In this sense, what we have specifically implemented in the context of the present work (the dynamic replication of $k$), are the mechanisms that make it possible for each kind of message to: generate $k$ message replicas, modify $k$ at runtime, and propagate the new $k$ to the CNs.

The prototype consists of 3 CNs connected to a central embedded device that includes both the NM and the FTTRS master/switch. The central device has a Celeron processor, 4 GB of RAM, and 10 Ethernet interfaces. Each CN is implemented separately in a dedicated embedded device with an Intel Atom processor and 2 GB of RAM. The operating system running in the central device and in each CN is Ubuntu 16.04. The Elementary Cycle (EC) is 20 ms long.

We have used this prototype to test the correct operation of the just-mentioned mechanisms in conjunction, by switching between configurations where $k$ varies for all kind of messages. These tests demonstrated that these mechanisms worked as intended, thereby demonstrating their feasibility.

For instance, one test consisted in running a single task (in a given CN) and, then, instruct the NM to apply a new configuration in which this task becomes critical so that (1) the task is triplicated, (2) each one of its replicas runs in a different CN, and (3) the message of each replica is also triplicated. Specifically, in the $i$ th EC we instructed the NM to reconfigure the system so as to apply the new configuration. Then, we observed the following autonomous actions: (1) the NM propagates all the changes to the CNs during the $i+1$ th EC; (2) the NM and all the CNs consolidate the changes in their read databases at the end of that $i+1$ th EC; (3) the NM triggers the execution of the new task replicas and their messages in the $i+2$ th EC; (4) and each task replica executes and transmits 3 replicas of its message accordingly in that EC.

## VI. Conclusions

In this paper we present the design and partial implementation of the DFT4FTT mechanism for the dynamic temporal replication of messages. This mechanism has the potential to increase ADESs tolerance to transient network faults in a cost-effective manner; by taking advantage not only from the temporal redundancy of messages, but also from the FT mechanisms of the different levels of the DFT4FTT architecture.

We explain how the system can detect when to reconfigure itself to adequately tolerate transient faults; the guidelines for how to decide on a proper number of replicas for each kind of message in any new configuration; and how to reliably and consistently propagate such a configuration to all the nodes of the ADES. Then, we outline the tests we carried out in our ongoing prototype of DF4FTT to demonstrate the feasibility of the ideas presented here.

In the short term we will propose how to use an adequate search technique to find proper system reconfigurations by considering, among other aspects, the degree of temporal redundancy of messages ($k$).

## References


[1] A. Ballesteros, J. Proenza, and P. Palmer, "Towards a Dynamic Task Allocation Scheme for Highly-Reliable Adaptive Distributed Embedded Systems," in *Proc. 22th IEEE ETFA*, Limassol, 2017.

[2] D. Gessner, J. Proenza, M. Barranco, and A. Ballesteros, "A fault-tolerant ethernet for hard real-time adaptive systems," *IEEE TII*, May 2019.

[3] I. Álvarez, A. Ballesteros, M. Barranco, D. Gessner, S. Derasevic, and J. Proenza, "Fault Tolerance in Highly-Reliable Ethernet-based Industrial Systems," in *Proceedings of the IEEE (Early Access)*, 2019.

[4] A. Larrucea, I. Martinez, J. Perez, V. Brocal, S. Peiró, H. Ahmadian, and R. Obermaisser, "DREAMS: Cross-Domain Mixed-Criticality Patterns," in *WMC 2016*, Nov. 2016.

[5] M. L. Raagaard, P. Pop, M. Gutiérrez, and W. Steiner, "Runtime Reconfiguration of Time-Sensitive Networking (TSN) Schedules for Fog Computing," in *2017 IEEE FWC*, Oct 2017.

[6] S. Derasevic, "Node Fault Tolerance for Distributed Embedded Systems based on FTT-Ethernet," Ph.D. dissertation, UIB, 2018.

[7] I. Álvarez, M. Barranco, and J. Proenza, "Mixing Time and Spatial Redundancy over Time Sensitive Networking," in *Proc. 48th IEEE/IFIP Int. Conf. on Depend. Systems and Networks (DSN)*, Luxemburg, 2018.

[8] DOD, *MIL-HDK-217F-2 Military Handbook, Reliability Prediction Of Electronic Equipment*. Department of Defense Washington DC, 1995.

[9] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Computers & Operations Research*, vol. 13, no. 5, 1986.

[10] W. Steiner, "An Evaluation of SMT-Based Schedule Synthesis for Time-Triggered Multi-hop Networks," in *31st IEEE RTSS*, Nov 2010.