First Implementation and Test of Reintegration Mechanisms for Node Replicas in the FT4FTT Architecture

Alberto Ballesteros, Sinisa Derasevic, Manuel Barranco and Julián Proenza
Dept. Matemàtiques i Informàtica, Universitat de les Illes Balears, Spain
a.ballesteros@uib.es, sinishadj@gmail.com, manuel.barranco@uib.es, julian.proenza@uib.es

***Abstract*—Distributed Embedded Control Systems (DECSs) used for critical applications must usually abide by strict real-time and dependability requirements. Correspondingly, the FT4FTT project proposes a complete fault-tolerant (FT) architecture for RT DECSs. The Flexible Time-Triggered Ethernet (FTT-Ethernet) communication protocol fulfills the RT requirements, while the FT mechanisms added on top of it, which are based on channel duplication and active replication of nodes, provide the FT behaviour. Temporary faults affecting the channel or the nodes, which are the most probable type of faults in DESs, can manifest in such a way that a node replica loses its coordination with the others and, thereby, it also loses its communication and/or computation capability from then on, leading to attrition of the redundancy initially provided by the active replication of nodes. This paper describes the implementation and test of specific mechanisms that are devised to determine which replicas are temporarily faulty and to promptly reintegrate them.**

I. Introduction

A great deal of Distributed Embedded Control Systems (DECSs) require real-time behaviour due to their firm timing restrictions thus constituting real-time (RT) systems. When RT DECSs are used for critical applications a high level of reliability has to be attained.

Accordingly, the FT4FTT project has developed a fault-tolerant (FT) architecture supporting the stringent real-time and fault tolerance guarantees these systems require. Moreover, when operating in a dynamic environment, the protocol used by the FT4FTT architecture also supports the on-line modification of the RT requirements of the exchanged messages.

More specifically, the protocol used by this architecture is FTT-Ethernet [1]. It is a publisher-subscriber protocol in which the FTT master embedded in the Ethernet switch (HaRTES [1]) controls the slaves' communication. Communication is divided into fixed-time slots called *Elementary Cycles* (ECs). Each EC is further divided into two windows: *synchronous window* for conveying periodic messages and *asyncronous window* for conveying aperiodic ones. Each EC is started by the FTT Master broadcasting a control message called the *Trigger Message* (TM) which conveys the schedule for periodic data messages exchanged by the nodes for that EC.

To acquire a high level of reliability FT4FTT devises fault tolerance mechanisms at different system levels to tolerate both permanent and temporary faults in node and channel.

Permanent faults in the channel are tolerated by switch and link duplication [2] and node permanent faults are tolerated by replicating the critical nodes identically, i.e. by using the same hardware and software (active replication [3]). Replicas then consistently exchange the input/output values after which each replica uses a majority voting to reach an agreement on the exchanged values and thus mask potential faults. Correspondingly, in [4] we implemented the node replication scheme and assessed that the mechanisms used to tolerate permanent faults worked as intended in a real environment.

More recently, in [5] we have dealt with the effects of temporary faults on the system. These faults have a much higher occurrence rate than permanent ones. It is demonstrated a temporary fault can manifest in such a way that from that point on a replica loses its communication and/or computation capability. In the case of multiple temporary faults, the redundancy initially provided by the active replication of nodes can be exhausted quickly, eventually leading to the system failure. Therefore, [5] proposes additional fault diagnosis and reintegration mechanisms used to identify temporary faulty replicas and bring them back to a coordinated operation with the non-faulty ones.

Note that the diagnosis and reintegration have to be done as promptly as possible so as to be able to restore the system FT capabilities and being able to tolerate additional faults.

The present paper extends the implementation described in [4] to add the mechanisms from [5] and perform a thorough experiment campaign to check in which scenarios the proposed mechanisms allow the reintegration of replicas and obtain some preliminary results of the time needed to reintegrate a replica affected by temporary faults.

II. FT4FTT System Design

Our system architecture consists of M nodes interconnected in a duplicated star topology. Critical nodes, i.e. the nodes for which fault tolerance has to be provided for, are replicated N times by means of active replication.

A control system in general is adapted to our existing system architecture as depicted by Fig. 1. The controller subsystem consists of node replicas executing the control law. These replicas are then connected to the system to be controlled (plant) and its instrumentation (sensors and actuators). Although, the specific connection to and the fault tolerance of

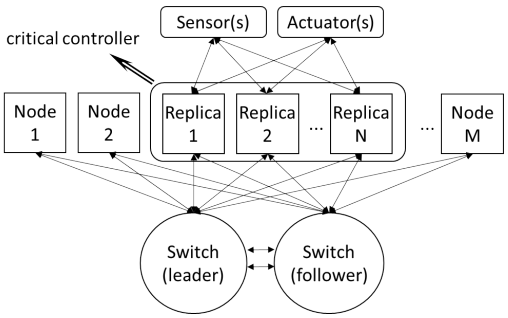

Fig. 1: System Architecture

the sensors and actuators are out of the scope of this work, in this paper we assume that sensor and actuator devices are replicated and each sensor/actuator replica is connected to one node replica. The control law being executed is a PID.

The focus is to provide fault tolerance to the critical nodes and their operation. In our FTT-based system due to node replication and majority voting a typical control application sense-control-actuate cycle is extended to the following 7 phases [6]:

- Sense (S). The data to be measured, i.e. the sensor values, is acquired by each replica from the connected sensor replica.
- Exchange Sensing (ES). Each replica shares with the others the acquired sensor value.
- Vote on Sensing (VS). Each replica executes majority voting on the locally and externally obtained sensor values.
- Control (C). Using the sensor value resulting from the previous vote each replica executes the control law and calculates the actuation value.
- Exchange Actuation (EA). Each replica exchanges with the others the calculated actuation value.
- Vote on Actuation (VA). Each replica executes majority voting on the calculated and received actuation values.
- Actuate (A). Each replica sends its actuation value obtained by the previous voting to the plant (actuator replica) which in turn carries out additional voting called *output consolidation* [7] transforming received actuation values into one, after which the actual actuation takes place.

In order to trigger the phases, both the exchange of values via Data Messages (DMs) and the execution of the application tasks by the replicas, we use the *network-centric* approach described in [6], i.e. the underlying network protocol triggers the phases. Particularly, phases are mapped onto ECs and implicitly or explicitly triggered by the FTT TM. Note that, depending on phases and EC duration, phases can be merged and triggered jointly by a single TM depending on application-specific timing needs.

A. Fault classification and fault tolerance mechanisms

Node permanent faults are tolerated by the active replication of nodes and majority voting, and permanent faults in links and switches are tolerated by duplication [2]

Since switches play a core role in the FTT system we force them to exhibit *crash failure semantics* by using internal duplication with comparison [2]. Therefore, a temporary fault in the switch is manifested as a permanent one (crash) and is tolerated by the aforementioned mechanism of duplication.

Temporary faults in the links are tolerated by means of time redundancy, i.e. critical messages are sent multiple times in the same EC. Apart from this, the existing CVEP protocol [8] designed to retransmit messages for voting throughout multiple ECs grouped in a so called *Voting Communication Round* (VCR) is also used to tolerate temporary faults in the links affecting these messages.

Nodes on the other hand exhibit *incorrect computation failure semantics* from the point of view of the other nodes in the system in order to simplify the majority voting procedure. This is done by the switch port guardians policing the traffic and preventing the propagation of certain errors thus allowing tolerance of Temporary Node Faults (TNF) by error compensation using the existing active replication and majority voting.

However, as stated in [5], temporary faults affecting the node replicas and the links can lead replicas to a state which is not recoverable by the existing mechanisms, leading to redundancy attrition. To cope with these faults we designed additional fault diagnosis and reintegration mechanisms [5]. These mechanisms are listed below:

- When a replica looses one or more TMs, the replica's Internal Counter (IC) used for task dispatching [6] has to be resynchronized. Specifically, each replica updates its IC upon each TM reception. To recover the counter's value after TM losses we use an information provided by the TM itself (TM sequence number) and the *TM resync* mechanism, described in [5].
- To recover a replica whose operational state has been corrupted, e.g. after a reset, we exchange and vote upon additional values constituting the replicas' operational state. For this we use the existing vote and exchange phases. After a faulty replica receives and votes on this state information it is considered as reintegrated. Accordingly, this point is called the *Voting reintegration point*.
- Whenever a permanent fault is exhibited, a replica is reset and reintegrated using the mechanism described above.
- The newly introduced error counters maintained by each replica, *Discrepancy Error Counter* (DEC) and *Communication Error Counter* (CEC), are used to diagnose when a replica exhibits a permanent internal fault affecting application execution and a permanent communication fault respectively. This counter is increased when a corresponding fault is detected by a replica and decreased in the absence of faults. When a predefined threshold is reached a replica diagnoses itself as a permanently faulty. In the case of consecutive faults, counters' increase rate is penalized for quicker fault detection and decrease rate is kept constant.
- The watchdog timer mechanism detects a crash of a replica. Watchdog timer is an external device connected to each replica. When the timer expires the replica attached to it is diagnosed as crashed. This timer is reset periodically with reception of a specific *You Are Alive* (YAA) message that is piggybacked on every TM.

III. Implementation

This section describes the extensions carried out in the node replication implementation we did in [4], to include the reintegration mechanisms previously presented. Specifically, as shown in Fig. 2, we added new functionalities in the replicas, both at the application and the FTT level.

First, the *TM resync* was already implemented and no relevant modifications were needed. Specifically, the *Action triggerer* in each replica triggers the execution of each of the application phases upon the reception of a TM. It is important to mention that here we divide the operation of the application in five phases. More precisely, we mix phases VS and C, as well as VA and A phases. This configuration makes the application more robust and faster.

Second, the *Voting Reint. Point* was partially implemented. Specifically, apart from voting on the sensor and actuation values in the VS and VA phases, respectively, we already voted on the control status during the VS phase to ensure that the control is done consistently. For this new implementation the control status is also voted in the VA phase to give more chances to reintegrate during a given control cycle. Additionally, we also include the setpoint in the voting.

Third, the *Communication Error Counter* (CEC) was implemented at the FTT level so that it can update its value upon the reception of a new TM. Every TM conveys the MS-vector which contains information about the DMs exchanged by the replicas and it serves to diagnose communication errors. If any communication error is detected, the value of the counter is increased, otherwise it is decreased unless it is zero. The value of this counter can be read by the application, which resets itself if it surpasses a certain threshold. Note that, for simplicity we perform a soft reset, which consists in reinitializing the operational state information rather than reinitializing the hardware components of the replica.

Fourth, the *Discrepancy Error Counter* (DEC) is similar to the CEC but for errors affecting the application. It is implemented at the application level and increases its value each time the result of the voting is different from what the replica populated.

Finally, the *You Are Alive (YAA) watchdog* is a software module that resets the application when several consecutive TMs are not received, that is, the replica is completely lost and it cannot reset itself.

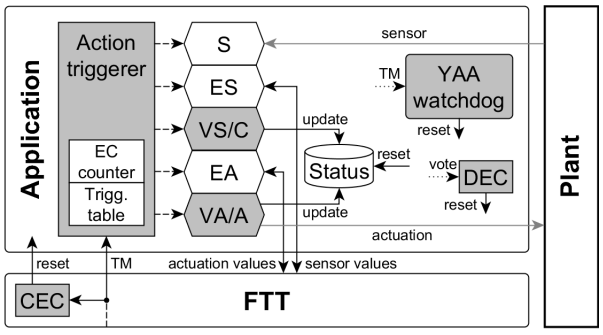

Fig. 2: Implementation of the reintegration mechanisms

IV. Testing the System

Here we present some of the tests we have carried out to check that the implementation of the fault tolerance mechanisms was done properly and that the design is correct. Moreover, we get some preliminary measurements of the reintegration time when temporary faults affecting the links or the replica itself do occur. The reintegration time can be defined as the time elapsed since the fault stops provoking errors until the operational state of the affected replica is consistent with the ones of the other replicas.

The experimental set-up used was a real prototype [4] composed of two FT4FTT switches and three node replicas. Each switch executes an F4FTT master, whereas each replica executes an FT4FTT slave implementing a typical control application. This application cyclically performs the five phases previously introduced, each one taking one EC. It is important to mention that in each of the tests we provoke errors in one of the replicas. Moreover, for each test we carry out several experiments in order to analyze the effects of the provoked errors in each of the phases of the application. This entails a first step towards a compete characterization of the reintegration time.

Next we describe each of the tests we conducted. To assist in the explanation Table I summarizes the error injection, that is, the type of error provoked and the instant in which it is injected. We finish this section by presenting the reintegration times registered for each test and the conclusions we have extracted from them.

S	ES	VS/C	EA	VA/A	S	ES	VS/C	EA	VA/A
TM									
	TM					DM			
		TM							
			TM					DM	
				TM					
Mem							Mem		Mem
	Mem						Mem		
		Mem							
			Mem						Mem
				Mem			Mem		Mem

TABLE I: Error injection for tests 1, 2, 3 and 4.

A. Tests conducted

In test 1 we simulate temporary link faults that prevent the replica from receiving some of the TMs (see top-left part of Table I). This causes the replica to not execute the associated phases. In order to reintegrate, the replica uses the *TM resync* mechanism, so that the subsequent phases can be executed normally. In some cases, not executing a given phase results in the replica not properly voting, which leads to a loss of consistency with the other replicas. In these cases the replica also uses the *Voting Reint. Point.* mechanism.

In test 2 we simulate temporary link faults that prevent the replica from transmitting and/or receiving DMs. Specifically, this test is composed of three sub-tests in which we analyze the behaviour of the replica when it is not able to receive them, transmit them and receive nor transmit them, during one phase. It is noteworthy that, as shown in the top-right part of Table I,

we only inject these error during the ES and EA phases, that is, when replicas exchange messages. In any other phase the fault does not manifest. Communication problems involving DMs result in the replica voting with a different subset of messages, leads to an inconsistent operational states. When so, the replica uses the *Voting Reint. Point.* mechanism to reintegrate.

In test 3 we simulate temporary node faults that corrupt the operational state of the replica. Specifically, as shown in the bottom-left part of Table I, in each phase we inject an error that modifies the values used and generated by the replica. As a result, said replica uses the *Voting Reint. Point.* mechanism to achieve consistency again.

In test 4 we simulate a replica crash that prevents it from correctly voting until it is restarted by the *DEC* mechanism. For this, as shown in the bottom-right part of Table I, we corrupt the voting values during several consecutive voting phases. Every time the replica is not able to correctly vote the DEC is increased and, once reached a threshold, the replica resets itself. After this reset the replica uses the *TM resync* and the *Voting Reint. Point.* mechanisms to reintegrate in the time and in the value domain, respectively.

In test 5 we simulate a replica crash that prevents it from executing any action until it is restarted by the *YAA watchdog*. For this we force the replica to not receive several consecutive TMs. This error injection is similar to the one carried out in test 1 (see the top-left part of Table I) but with the loss affecting several phases. As in test 4, once the watchdog resets the replica, the *TM resync* and the *Voting Reint. Point.* mechanisms make it possible to achieve consistency again in the time and in the value domain, respectively.

B. Results

In each of the tests executed the faulty replica was able to correctly reintegrate in a number of ECs shown in Table II. Each row corresponds to one of the tests and each of the first five columns corresponds to one of the experiments conducted for each of these tests. Specifically, each of these columns refers to the phase in which the fault finishes. Finally, the last two columns show, for each test, the maximum and average of the reintegration time. Next we present the main conclusions we extracted from the tests and Table II.

	Last phase affected by the fault					**Statistics**	
	S	**ES**	**VS/C**	**EA**	**VA/A**	**max**	**avg**
1	2	3	2	0	0	3	1
2	-	3	-	0	-	3	1.5
3	2	3	2	0	0	3	1.4
4	-	-	2	-	3	3	2.5
5	2	3	2	4	3	4	2.8

TABLE II: Time to reintegrate (in ECs) for every test and exp.

As can be concluded, the time needed to reintegration is the time until the next successful voting. Specifically, according to the Voting Reint. Point mechanism, this happens after receiving the DMs from the other replicas and voting on them. In this sense, as can be seen in Table II, the values for each column are almost identical. The only difference occurs when injecting errors after voting on the sensor values.

A temporary communication fault occurring after the VS/C phase does not make the replica inconsistent. This is because the actuation is already calculated and it must be identical in all the replicas. In contrast, when a restart is needed to stop the error, the initialization of the operational state forces the replica to wait for the next reintegration point, which occurs in the next application cycle.

V. Conclusions and Future Work

In this paper we presented the first implementation and test of the FT4FTT fault diagnosis and reintegration mechanisms, which make it possible to recover faulty node replicas that suffered from temporary faults, in order to restore the initial redundancy and, thus, fault tolerance of the system. For this we extended the FT4FTT prototype we already developed for a previous work, in which we added new functionalities, mostly at application level but also at and the communication level. Finally, we performed various tests to assess the correctness of the design and implementation, as well as to get some preliminary measurements of the reintegration time.

The next steps involve implementing the YAA watchdog in hardware and also the hard reset. With this, the measurements of the reintegration time will also contain the time needed to start the application and to enter into the communication. Finally, we also intend to carry out a dependability evaluation of the entire system and measure its reliability.

Acknowledgments

This work was supported by grants DPI2011-22992 and TEC2015-70313-R funded by the Spanish Ministerio de Economia y Competitividad (MINECO) and by the Fondo Europeo de Desarrollo Regional (FEDER). Sinisa Derasevic was supported by a scholarship of the EUROWEB Project, which is funded by the Erasmus Mundus Action II programme of the European Commission.

References

[1] R. Santos, "Enhanced Ethernet switching technology for adaptive hard real-time applications," Ph.D. dissertation, PhD Thesis, University of Aveiro, Aveiro, Portugal, 2010.

[2] D. Gessner, J. Proenza, and M. Barranco, "A proposal for master replica control in the flexible time-triggered replicated star for Ethernet," in *Proc. 10th IEEE World Conf. on Factory Comm. Systems (WFCS)*, 2014.

[3] M. Wiesmann, F. Pedone, A. Schiper, B. Kemme, and G. Alonso, "Understanding replication in databases and distributed systems," in *Proc. 20th Int. Conf. on Distributed Computing Systems*. IEEE, 2000.

[4] A. Ballesteros, S. Derasevic, D. Gessner, F. Francisca, I. lvarez, M. Barranco, and J. Proenza, "First Implementation and Test of a Node Replication Scheme on top of the FTTRS," in *Proc. 12th IEEE World Conf. on Factory Comm. Systems (WFCS)*, 2016.

[5] S. Derasevic, M. Barranco, and J. Proenza, "Designing fault-diagnosis and reintegration to prevent node redundancy attrition in highly reliable control systems based on FTT-Ethernet," in *Proc. 12th IEEE World Conf. on Factory Comm. Systems (WFCS)*, 2016.

[6] S. Derasevic, J. Proenza, and M. Barranco, "Using FTT-ethernet for the coordinated dispatching of tasks and messages for node replication," in *Proc. 19th IEEE Int. Conf. on Emerging Tech. and Factory Automation (ETFA)*, 2014.

[7] D. Powell *et al.*, *A generic fault-tolerant architecture for real-time dependable systems*. Springer, 2001.

[8] S. Derasevic, M. Barranco, and J. Proenza, "Appropriate consistent replicated voting for increased reliability in a node replication scheme over FTT," in *Proc. 19th IEEE Int. Conf. on Emerging Tech. and Factory Automation (ETFA)*, 2014.