Designing fault-diagnosis and reintegration to prevent node redundancy attrition in highly reliable control systems based on FTT-Ethernet

Sinisa Derasevic, Manuel Barranco, Julián Proenza
DMI, Universitat de les Illes Balears, Spain
sinishadj@gmail.com, manuel.barranco@uib.es, julian.proenza@uib.es

***Abstract*—Distributed Embedded Control Systems (DECSs) used for Real-Time (RT) critical applications must satisfy stringent time requirements and attain high reliability. FTT-Ethernet provides nodes of DECSs with real-time communication capabilities, but does not include Fault Tolerance (FT) mechanisms. The FT4FTT project aims at proposing a complete FT architecture for RT critical DECSs. It uses a duplicated switched FTT-Ethernet star and active node replication with consistent distributed majority voting to respectively tolerate channel and node faults. However, FT4FTT, in its current state, still lacks mechanisms to prevent node redundancy attrition due to temporary faults affecting the nodes and channel, which are the most likely types of faults in DESs. This paper presents our ongoing work to complete the FT4FTT architecture with appropriate fault-diagnosis and reintegration mechanisms that overcome this limitation.**

I. Introduction

The design of Real-Time (RT) critical Distributed Embedded Control Systems (DECSs), specially adaptive ones, is currently one of the main topics of research in the context of embedded systems. In order to provide these critical systems with the high reliability they require, the FT4FTT project proposes a complete fault-tolerant (FT) architecture.

A system following the FT4FTT architecture has M nodes interconnected by two enhanced HaRTES switches [1]. Each switch embeds an FTT Master that cooperatively controls the communication of multiple slaves. Communication is divided into cycles called *Elementary Cycles* (ECs). Each EC starts by the FTT Masters quasi-simultaneously broadcasting a control message, the *Trigger Message* (TM), that indicates which messages must be transmitted, and by whom, in the EC.

In FT4FTT, permanent hardware faults affecting the channel are tolerated by means of switch/link replication and switch internal duplication and comparison [2]. Link temporary faults are tolerated by *time redundancy*, i.e. by retransmitting critical messages. For instance, the TM is sent multiple times in each EC [3]. Likewise, as in the example of Figure 1, in FT4FTT permanent and temporary hardware faults affecting nodes are tolerated by means of active node replication [4] with *Distributed Consistent Majority Voting* (DCMV), i.e. by error compensation. To simplify the voting procedure each switch policies the traffic coming to its ports, so that any arbitrarily-faulty node is enforced to exhibit incorrect computation failures from the other nodes' point of view.

The basic idea of distributed voting [5] is that each replica, after some partial execution called segment, produces an output

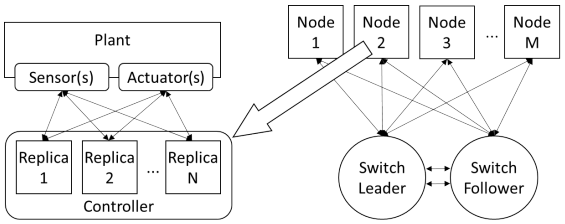

Fig. 1. Example of control application relying on FT4FTT

called cc-vector. Replicas then exchange the cc-vectors and vote on them to agree on the same result. This result is called consensus cc-vector and each replica uses it to compute the next segment. To guarantee that non-faulty replicas produce the same consensus when voting, it is necessary to enforce external replica determinism [6], i.e. that each replica is provided with the same input values (cc-vectors). In [7] and [8] we propose and implement a *Consistent replication Voting Protocol* (CVP) to reliably vote while enforcing external replica determinism. On the one hand it includes a sub-protocol called *Cc-vector Exchange Protocol* (CVEP). Replicas use it to exchange multiple times cc-vectors and their corresponding acknowledgments (ACKs) during a *Voting Communication Round* (VCR), to tolerate temporary faults affecting the links. At the end of each VCR, the switches send to each replica a Message-Status vector (MS-vector) that specifies for each cc-vector both, whether or not its replica has transmitted it, and which replicas have acknowledged it. Once the VCR ends, each replica uses an algorithm called *Voting Set-Up Algorithm* (VSUA), to decide which replicas can consistently vote based on the content of the MS-vector.

Nevertheless, FT4FTT currently lacks mechanisms to prevent temporary faults from causing *node redundancy attrition*, i.e. to prevent temporary faults from making replicas to be perceived as permanently faulty. Time redundancy is used to tolerate temporary faults in the links, but not in the replicas. Thus, a single temporary fault affecting a replica itself can make it to behave incorrectly from then on, e.g. a replica is permanently desynchronized if its reception buffer is temporarily corrupted so that it does not receive the TMs of an EC. Preventing redundancy attrition is mandatory to attain high reliability, as temporary faults are the most probable ones. Moreover, although FT4FTT provides time redundancy, preventing node redundancy attrition also caused by link temporary faults opens room for flexible FT mechanisms, e.g. time redundancy can be reduced temporarily to devote more

bandwidth to other (FT) mechanism. Because all of these reasons, current paper presents our ongoing work to complete the FT4FTT architecture with adequate Fault-Diagnosis and Reintegration (recovery), FDR, mechanisms that thoroughly prevent node redundancy attrition due to temporary faults.

Finally, despite FDR mechanisms for RT critical DES have been proposed in the past, there is a lack of literature addressing the recovery of replicas in active node replication schemes for such systems. The few works that do it either assume too restrictive failure semantics, or provide a very complex solution as they consider an Event-Triggered architecture. The most recent Time-Triggered control networks, such as TTP, FlexRay or TTEthernet [9], [10], [11], propose membership and clock synchronization mechanisms that tolerate the failure of fail-silent groups of node replicas, e.g. called FTUs in [9]; but do not address how replicas, or even FTUs, recover.

Section II clarifies concepts essential to this work; presents the main structural decisions; and sketches the strategy used to identify and design the fault-diagnosis and reintegration mechanisms needed for the different faults. Section III explains the classification of faults and outlines the mechanisms. Section IV concludes the paper and points out future work.

II. Design Rationale

A. Definitions

Next we clarify some concepts that are fundamental to understand the rest of the paper. First, for non-faulty replicas to provide a correct service they need to be *replica determinate*, i.e. they must deliver the same outputs and perform the same service state changes [6]. In this sense the DCMV does not only allow nodes to compensate errors, but also enforces node replica determinism. This is because each divergence among replicas that is not provoked by a fault is resolved in each segment when obtaining the consensus cc-vector.

Moreover, in DCMV each replica votes locally and uses the consensus cc-vector to start the next segment. Thus, besides compensating errors at the system level, DCMV also allows each replica suffering from a temporary fault in a segment to seamlessly *recover*, i.e. to correct its internal state, by simply voting and, then, to become replica determinate again. However, note that a temporary fault may affect several segments, or it can manifest as permanent in which case the replica has to be reinitialized. To recover when a temporary fault manifests as such, a replica may need to diagnose it and, then, perform additional actions prior to simply voting at the end of a segment. For the sake of simplicity, we will use the concept of *reintegration* to refer to the actions a replica carries out to become replica determinate, independently of whether these actions are complex or just a simple voting.

Finally, we consider that a non-faulty replica is replica determinate at the end of a given segment when its *operational state* allows the replica to produce, at the next segments and in absence of faults, results that are correct and consistent with respect to the rest of non-faulty replicas. In a control application relying on FT4FTT, the *operational state* includes the value of the sensors, actuators, set point (SP) and, also, the *control status*, e.g. the error, the derivative and the integral terms in the case of a PID controller.

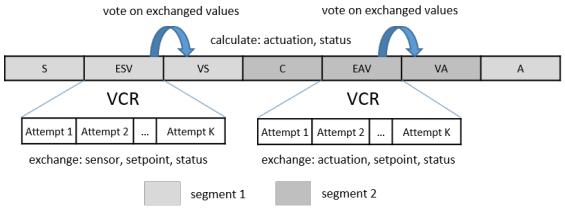

Fig. 2. Extended Control Application Cycle

B. Extended Control Application Cycle

In FT4FTT each replica continuously repeats a sense-control-actuate cycle that is divided into 7 phases, some of which are used to vote on the sensors or actuators [12]. In the present work we both extend the cycle by adding more actions (*tasks* in [12]) to some phases, and change the way in which the TM is used to trigger (*dispatch* in [12]) those actions.

Figure 2 depicts the phases and actions of the Extended Control Application Cycle. The phases are renamed, but the actions already proposed in [12] for each one of them are kept. Specifically, in *Sense* (S) the replica acquires the value of each sensor; in *Exchange Sensor Values* (ESV) it exchanges the value of each sensor with the other replicas during one VCR using the CVEP (see Section I); in *Vote on Sensor values* (VS) the replica obtains the consensus value of each sensor (for each consensus value it votes on the value it proposes and the values received from the other replicas during the ESV); in *Control* (C) it uses the consensus sensor values to calculate both the actuation/s value/s and the *control status*; in *Exchange Actuation Values* (EAV) the replica exchanges the calculated actuation values with the other replicas during one VCR using the CVEP; in *Vote on Actuation values* (VA) it obtains the consensus of each actuation value (by voting on the value it proposes and the ones received from the other replicas in the EAV); and in *Actuate* (A) it sends each consensus actuation value to the corresponding actuator.

Note that phases A, S, ESV and VS can be considered as a first segment; whereas phases C, EAV and VA constitute a second one. The voting carried out in VS was already used in [12] to compensate erroneous sensor values, enforce replica determinism, and seamlessly reintegrate any replica acquiring a sensor incorrect value. Similarly, the voting in VA was used to compensate erroneous actuation values, enforce replica determinism and, also, to seamlessly reintegrate any replica producing an actuation incorrect value.

In the extended cycle the replicas use the ESV/VS and the EAV/VA to exchange (and vote on) not only the values of the sensors and actuators respectively, but also the SP and the control status. How and when the replica acquires the SP is application dependent, e.g. it can receive the SP through the network or can calculate it. In any case, divergences or errors in the SP or control status would propagate from one control cycle to the next ones, leading the replica to indefinitely calculate the control status and the actuation values incorrectly. Fortunately, on the one hand, exchanging and voting on these values in the ESV/VS corrects discrepancies due to either divergences/errors when obtaining them in the previous control cycle, or errors in their storage. Thus, this exchange and voting enforces replica determinism and provides seamlessly

reintegration with respect to these values at the end of the first segment, i.e. these values can correctly serve to calculate and update the actuation and control status in phase C. On the other hand, exchanging and voting on these values in the EAV/VA corrects discrepancies caused by divergences/errors when calculating and/or storing them in C. This correction does serve neither to enforce replica determinism nor to seamlessly reintegrate. This is because the SP and the control status are not needed to carry out the first segment and will be corrected there in any case. However, the extra exchange and voting in EAV/VA increases the robustness, as the replica will be able to propose correct SP and control status in the ESV of the next control cycle.

Finally, as concerns the change of how the TM is used to trigger the actions of each phase, we propose to substitute the internal EC counter used in [12] for that purpose by a local TM sequence number counter. As it will be explained in Section III, this allows the replica to properly reintegrate after receiving no TM during one or more ECs.

C. Analysis of faults and identification of mechanisms

To identify and properly design the necessary FDR mechanisms, we first classified the faults a replica may encounter. Then we exhaustively analyzed how each one of these faults can affect the communication and operation of the replica, what errors do these faults generate in each case, and how the replica can address them.

Table I summarizes the results of this classification and analysis. Rows represent the different classes of faults, and columns which communication or operational capabilities of the replica can be affected. Each cell sketches the mechanism/s that are used to tolerate, diagnose and/or reintegrate from the fault. Due to space limitations we cannot detail these mechanisms and how they are used in each case. Next section, though, briefly describes each kind of fault and outlines the mechanisms used to address them.

III. Fault-diagnosis & reintegration mechanisms

The types of faults and the FDR mechanisms we are designing to address them are as follows.

1. *Temporary hardware Faults affecting Links* (TFL). They corrupt messages and are tolerated using the time redundancy provided by the TM replication or the CVEP. These mechanisms were respectively proposed in [3] and [7].

2. *Temporary Long Lasting hardware Faults affecting Links* (LLFL). They are TFLs for which the aforementioned time redundancy does not suffice. They are tolerated by means of node replication and majority voting. In addition, each affected replica reintegrates by using a mechanism called *TM resync* and/or a mechanism called *Voting Reint. Point*.

As concerns the *TM resync*, note that each TM conveys a number called *Trigger Message SeQuence number* (TMSQ) the switches increment every EC. As mentioned in Section II-B, each replica increases a local *Trigger Message SeQuence number Counter* (TMSQC) in each EC. As long as a replica and its corresponding links are non-faulty, replica's TMSQC matches TMSQ. When the replica detects a discrepancy between them, it determines that it has lost all TMs of (TMSQ - TMSQC) consecutive ECs. Moreover, since each value of the TMSQ corresponds to a given EC of the control cycle, the replica also knows during which ECs of which phases it has been black out. Then, the replica reintegrates at the EC/phase level, i.e. regains EC/phase synchronization, by forcing its TMSQC to the value of the just received TMSQ. Afterwards, depending on which ECs/phases it has lost, the replica determines which actions it can perform in the current EC, e.g. if the current EC is the last one of ESV and the replica was able to sense in S, then it sends its sensor cc-vector and acknowledges the ones by other replicas.

The replica uses the *Voting Reint. Point* mechanism to reintegrate when it loses replica determinism with respect to the operational state. The replica applies this mechanism when after voting it detects that a fault led it to consider an erroneous operational state value; or when after reintegrating at the EC/phase level, it determines that it has not correctly updated any of these values. This mechanism simply consists in updating any incorrect value by the consensus one obtained when voting. Also note that for voting, the replica does not use the VSUA [7] any longer; but simply votes on both its value and all cc-vectors it receives. The VSUA was designed in absence of the FDR mechanisms herein presented and, thus, it proposed a complex voting to take as much as advantage as possible from the node/links redundancy in a best effort basis.

3. *Permanent hardware Faults affecting Links* (PFL). They corrupt messages, manifest as permanent transmission/reception omissions, and are tolerated by link replication.

4. *Temporary hardware Faults affecting Node replicas* (TFN). They affect replicas internal circuitry. If a TFN hampers the replica capacity for receiving/transmitting, it may be tolerated by means of the TM replication and/or the CVEP. However, if these mechanisms do not suffice or the fault affects the replica internal operations, e.g. the replica cannot calculate the actuation, then the same mechanisms as for LLFLs apply. In other words, the fault is tolerated by node replication and majority voting, and the replica reintegrates using the *TM resync* and/or the *Voting Reint. Point*.

5. *Temporary hardware Faults affecting Node replicas manifesting as Permanent* (TFNP). They are TFNs preventing a replica from correctly communicating and/or operating as long as it is not reinitialized. A TFNP is diagnosed by different mechanisms, namely the *Communication Error Counter* (CEC), the *Discrepancy Error Counter* (DEC) and the *You Are Alive (YAA) watchdog*. When so, the replica is reset and, then, reintegrated by means of *TM resync* and *Voting Reint. Point*.

The replica increases/decreases the CEC after each VCR depending on whether or not it successfully transmitted/received cc-vectors and acknowledgments in that VCR. For this, the replica uses both the content of the MS-vector (see Section I) and the status of its reception buffers. Likewise, the replica increases/decreases the DEC after each voting depending on whether or not it could successfully vote on each value of the operational state, and whether or not its local values disagreed with the consensus ones. When any of these counters exceeds a given threshold, the replica resets itself.

The *YAA watchdog* is an independent device attached to the replica. It is used to guarantee that the replica resets even

TABLE I. FAULT ANALYSIS SUMMARY

	rx TM	rx/tx cc-vec./ACK/SP	sensor acquisition	actuator/control calculation	majority voting
TFL	TM replication	CVEP	x	x	x
LLFL	node rep. & maj. vot. TM resync Voting Reint. Point	node rep. & maj. vot. Voting Reint. Point	x	x	x
PFL	link replication	link replication	x	x	x
TFN	TM replication node rep. & maj. vot. TM resync Voting Reint. Point	CVEP node rep. & maj. vot. Voting Reint. Point	node rep. & maj. vot. Voting Reint. Point	node rep. & maj. vot. Voting Reint. Point	node rep. & maj. vot. Voting Reint. Point
TFNP	node rep. & maj. vot. YAA watchdog reset TM resyn. Voting Reint. Point	node rep. & maj. vot. diagnosis (CEC) reset TM resync. Voting Reint. Point	node rep. & maj. vot. diagnosis (DEC) reset TM resync. Voting Reint. Point	node rep. & maj. vot. diagnosis (DEC) reset TM resync. Voting Reint. Point	node rep. & maj. vot. diagnosis (DEC) reset TM resync. Voting Reint. Point
PFN	node rep. & maj. vot.	node rep. & maj. vot. degraded mode diagnosis degraded mode notification	node rep. & maj. vot. degraded mode diagnosis degraded mode notification	node rep. & maj. vot. degraded mode diagnosis degraded mode notification	node rep. & maj. vot. degraded mode diagnosis degraded mode notification

when a TNFP makes it to crash or to be lost so that it does not process the TM anymore. The YAA watchdog waits for the periodic reception of a *You Are Alive message* (YAA), sent from the switches within the TM, that has to traverse the node application on its way to the watchdog. If the replica crashes or gets lost, then the watchdog expires and resets the replica. We are exploring the possibility of including within the switches mechanisms to diagnose a replica as faulty; so that the switches can force the replica to reset by not sending the YAA. Moreover, to prevent the replica from forging the YAA, we would piggyback within the YAA a sequence number based on a look up table only known by the switches and watchdogs.

However, implementing fault-diagnosis mechanisms in the switches is strongly limited. First, switches cannot rely on any information received from the replica, as the replica is not trustworthy. Second, to keep switches application-independent, switches should only include error counters related to the observed traffic, i.e. switches should only include error counters they can manage based on the content of the MS-vector they generate in each VCR, or on statistics about the transmissions (scheduled in the TM) the replica omits.

6. *Permanent hardware Faults affecting Node replicas* (PFN). They permanently affect replicas internal circuitry, so that replicas cannot correctly communicate and/or operate anymore. They are tolerated by active node replication with majority voting. We are exploring the feasibility of switches being able to diagnose, after several resets, when a replica can communicate/operate in a degraded mode. If so the switches could use a special version of the YAA to instruct the replica to not reset but try to communicate/operate in that mode.

IV. CONCLUSIONS & FUTURE WORK

FT4FTT aims to provide a complete fault-tolerant architecture for real-time critical DECSs based on FTT-Ethernet. Particularly, FT4FTT tolerates node faults by active node replication. But as all other architectures based on even the most recent Time-Triggered (TT) control networks, currently FT4FTT lacks mechanisms for nodes to diagnose and reintegrate from temporary faults. This leads to node redundancy attrition that ultimately limits system reliability and, thus, the benefits of redundancy investment. This paper systematically outlines the Fault-Diagnosis and Reintegration (FDR) mechanisms we are designing to thoroughly cover all temporary faults leading to node redundancy attrition in FT4FTT. We discovered that the TT nature of FTT and some of its mechanisms, e.g. the TM, allow designing FDR mechanisms simpler than the ones literature proposes for Event-Triggered systems. We believe that our FDR mechanisms can be used in other TT control networks to some extent. We plan to formally verify these mechanisms and quantify their reliability benefits.

ACKNOWLEDGMENTS

Supported by DPI2011-22992 and TEC2015-70313-R (Spanish *Ministerio de economía y competividad*), by FEDER funding and by the EUROWEB Project funded by the Erasmus Mundus Action II programme of the European Commission.

REFERENCES

[1] D. Gessner, J. Proenza, M. Barranco, and L. Almeida, "Towards a Flexible Time-Triggered Replicated Star for Ethernet," in *Emerging Technologies & Factory Automation (ETFA), 2013 IEEE 18th Conf.*

[2] B. W. Johnson, *Design and analysis of fault-tolerant digital systems*. Addison-Wesley Reading, MA, 1989, vol. 6.

[3] D. Gessner, J. Proenza, and M. Barranco, "A proposal for master replica control in the Flexible Time-Triggered Replicated Star for Ethernet," in *Factory Communication Systems (WFCS), 2014 10th IEEE Workshop*.

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

[5] L. Chen and A. Avizienis, "N-Version Programming: A fault-tolerance approach to reliability of software operation," in *Proc. 8th IEEE Int. Symp. on Fault-Tolerant Computing (FTCS-8)*, 1978.

[6] S. Poledna, "The problem of replica determinism," in *Fault-Tolerant Real-Time Systems*. Springer, 1996, vol. 345.

[7] S. Derasevic, M. Barranco, and J. Proenza, "Appropriate consistent replicated voting for increased reliability in a node replication scheme over FTT," in *Emerging Technology and Factory Automation (ETFA), 2014 IEEE*.

[8] S. Derasevic, M. Melia, A. Ballesteros, M. Barranco, and J. Proenza, "First experimental evaluation of the consistent replicated voting in the Hard Real-Rime Ethernet Switching architecture," in *Emerging Technologies & Factory Automation (ETFA), 2015 IEEE 20th Conference on*. IEEE, 2015.

[9] H. Kopetz, "Fault containment and error detection in TTP/C and FlexRay," *Vienna, Technical University of Vienna*, 2002.

[10] FlexRay Consortium *et al.*, "FlexRay communications system-protocol specification," *Version*, vol. 2, no. 1, 2005.

[11] W. Steiner, G. Bauer, B. Hall, and M. Paulitsch, "TTEthernet: Time-Triggered Ethernet," 2011.

[12] S. Derasevic, J. Proenza, and M. Barranco, "Using FTT-Ethernet for the coordinated dispatching of tasks and messages for node replication," in *Emerging Technology and Factory Automation (ETFA), 2014 IEEE*.