First Implementation and Test of a Node Replication Scheme on top of the Flexible Time-Triggered Replicated Star for Ethernet

Alberto Ballesteros, Sinisa Derasevic, David Gessner, Francisca Font, Inés Álvarez
Manuel Barranco and Julián Proenza
Dept. Matemàtiques i Informàtica, Universitat de les Illes Balears, Spain
a.ballesteros@uib.es, {sinishadj, davidges}@gmail.com, {manuel.barranco, julian.proenza}@uib.es

***Abstract*—Distributed embedded systems typically have real-time and dependability requirements. Moreover, they must also be flexible to changing conditions when they are deployed in dynamic environments. The FT4FTT project aims at providing a switched Ethernet architecture that can support distributed control applications that are predictable, highly-reliable and adaptive. FT4FTT relies on the Flexible Time-Triggered Replicated Star for Ethernet (FTTRS) to tolerate channel faults. Moreover, nodes' hardware faults are tolerated by means of active node replication with majority voting. In order to coordinately trigger the execution of the tasks in the replicas, we designed the CD4NR mechanism, in which the network assists in deciding what to execute and when. This paper presents the first implementation of the CD4NR mechanism on a real prototype of FTTRS and the first testing of the complete system. For this we developed an experimental setup, based on the hardware-in-the-loop technique, running a real-time control application.**

I. Introduction

Distributed embedded systems (DESs) typically have real-time and dependability requirements. Moreover, when they are deployed in dynamic environments, the communication subsystem and the nodes must also be flexible to cope with changing conditions. The FT4FTT (Fault Tolerance for Flexible Time-Triggered Ethernet) project aims at providing an architecture for a complete distributed system based on switched Ethernet which could support distributed control applications that are predictable, highly-reliable and adaptive.

The communication subsystem of FT4FTT is based on the Flexible Time-Triggered communication paradigm (FTT) [1]. FTT is a master/multi-slave solution that provides predictability and flexibility in the communications. In FTT the master divides the communication into fixed-duration intervals called Elementary Cycles (ECs). Each EC starts with the master transmitting the so-called Trigger Message (TM) which contains the EC schedule, i.e., the list of messages that the slaves must transmit during that EC. This schedule is calculated online according to the communication requirements of the slaves, who can request their modification at any time.

FTT does not provide high-reliability, that is why in FT4FTT we developed the Flexible Time-Triggered Replicated Star for Ethernet (FTTRS) [2], a fault-tolerant network based on FTT. More precisely, as sketched in Fig. 1, slaves are interconnected through two custom switches each one embedding an FTT master. Additionally, FTTRS switches are connected between them by means of several links called *interlinks* in order to coordinate their operation.

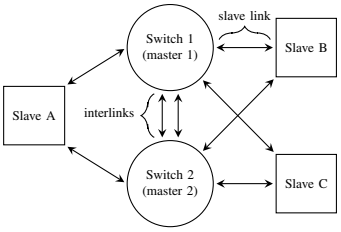

Fig. 1: FTTRS architecture.

FT4FTT also provides mechanisms to tolerate hardware faults affecting the nodes [3]. On the one hand, we use *active replication*, i.e., nodes performing critical tasks are replicated and each of these replicas executes the same code in parallel. On the other hand, we use a distributed majority voting algorithm to compensate the errors that nodes could generate. Specifically, software is partitioned into *segments* that are executed in parallel by replicas. For each segment each replica produces an output, which is exchanged and voted upon to reach a consensus. This consensus value is then used by the replicas as input for the next segment.

The operation of the application is driven by the communication channel. This is because a given segment cannot start without exchanging and voting on the output of the previous one. In [4] we proposed the Coordinated Dispatching of tasks and messages for Node Replication (CD4NR) mechanism, which makes it possible to control the execution of the tasks carried out by replicas and the transmission of the messages they need to exchange. In [5] we already did a first assessment of the CD4NR mechanism. However, this was done in a simulated non-real-time environment and with a non-redundant version of the FT4FTT communication subsystem.

In this paper we present the prototyping and testing of the CD4NR mechanism on a real FTTRS network. One relevant contribution of this paper is the development of the experimental setup used to test this mechanism, which runs a real-time control application. Additionally, since the prototype we have built includes the most important aspects of the node replication and the communication subsystem, the testing we did represents a first step towards a complete evaluation of the design of the FT4FTT architecture.

II. System Architecture and Operation

The architecture of the system we have prototyped, as shown in Fig 2, can be divided into two parts: the plant and the replicated distributed control system.

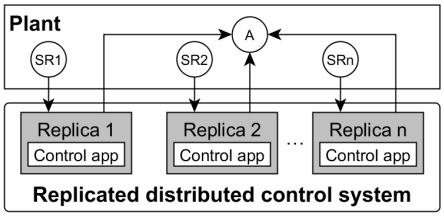

Fig. 2: System architecture

The plant represents a physical system that has to be controlled. The interaction with the control system is carried out by means of *sensor replicas*, each connected to one node replica, and an *actuator subsystem*, which is composed of one or more actuators connected to all the replicas.

The replicated distributed control system is composed of several interconnected replicated nodes. Each of these replicas is a processing unit that executes tasks related to the control of the plant, as well as to the management of the replication. As explained previously, replicas are interconnected through FTTRS, which provides a duplicated star-based communication channel. FTTRS yields important benefits from the point of view of the tolerance to faults. On the one hand, thanks to the duplication of the channel, replicas can still communicate even if one switch and/or some links fail. On the other hand, if replicas fail, switches can prevent the propagation of the errors they produce, which eases the design and implementation of the fault-tolerant mechanisms [6].

A. Control Application Phases

Typical control applications cyclically perform three tasks: *sense*, read the state of the plant by means of sensors; *control*, determine the actuation to be performed at the plant to change its current state to the desired state; and *actuation*, carry out the action to change the state of the plant. The duration of one control cycle is called sampling period (T_s).

In FT4FTT replicas perform additional tasks in which they exchange and vote on the values of the sensors and on the result of the actuation. That is why we proposed a more complex scheme composed of seven phases:

1) *Sense*. Each replica retrieves, from the sensor attached to it, the current state of the plant.
2) *Exchange of sensor values*. Each replica sends to the other replicas, through the fault-tolerant communication channel, its local view of the state of the plant.
3) *Voting on sensor values*. Each replica votes on all the sensor values, i.e., the ones received from the other replicas and the one acquired from the sensor attached. The result of this vote is the so-called *consensus sensor value*. It should be noted that the type of voting performed depends on the type of data on which the vote takes place.
4) *Control*. Each replica uses the consensus sensor value to determine the actuation to be performed at the plant.
5) *Exchange of actuation values*. Each replica sends to the other replicas, through the fault-tolerant communication channel, the output of the control algorithm.
6) *Voting on actuation values*. Each replica votes on all the actuation values, i.e., the ones received from the other replicas and the one obtained from the control algorithm. The result of this vote is the so-called *consensus actuation value*. Note that in this phase the type of voting performed always assumes that all the actuations are identical.
7) *Actuate*. Each replica sends its consensus actuation value to the actuator subsystem. This subsystem consolidates the received values and performs the actuation.

In order for the FT4FTT network to assist in the triggering of the execution of the phases in the replicas, phases are mapped into ECs. The most intuitive mapping would be to devote one EC to each of the phases. However, this is not the only possible mapping. For instance, if the plant needs a very tight control, we can reduce the sampling period by executing several phases in one EC. In contrast, if the control system is working in a very harsh environment in which errors affecting the communication are likely to occur, we can devote several ECs for phases 2 and 5 so that replicas have various chances to exchange their messages.

B. Coordinated Dispatching of Tasks and Messages

To trigger the execution of each of the control application phases in the correct instant, according to the pace dictated by the network, we designed the Coordinated Dispatching of tasks and messages for Node Replication (CD4NR) mechanism.

As explained previously, in FTT the master uses the TM to periodically trigger the transmission of messages in the nodes. The CD4NR mechanism takes advantage of this service to also trigger the execution of tasks in the nodes.

More precisely, we provide each replica with an EC counter and a dispatching table, i.e., a table specifying the tasks that should be executed in each EC. Every time a replica receives a new TM, it updates the value of its EC counter following Eq. 1, where TM_seqno is the so-called *TM sequence number*, a numerical value masters insert into the TM to indicate number of the current EC; and T_s is the sampling period of the application measured in ECs. It should be noted that the way in which the EC counter updates its value has been slightly modified with respect the first proposal did in [4] to be able to tolerate the omission of TMs. Once the application has determined the value of the EC counter, the dispatching table is consulted to executed the corresponding tasks.

$$EC\_counter = TM\_seqno \bmod T_s \tag{1}$$

The main advantage of this solution is that the communication subsystem remains unaware of the operation of the application. This is because, no changes are needed in the operation of FTT to support this dispatching scheme.

III. Implementation

The implementation of the node replication scheme implied the addition of specific features in various components and in different layers of the architecture. As sketched in Fig. 3, we can distinguish between the plant and the replicated distributed control system, this last composed of a set of replicas (here only one is represented) and the switches. Additionally, the control system can be divided into three layers. First, the Ethernet layer is responsible for transmitting the Ethernet frames among the replicas and the switches. Second, the FTT layer contains the modules providing the FTT services, namely one FT4FTT master inside each switch and one FT4FTT slave inside each replica. Finally, the application layer is only implemented in the replicas and performs the control of the plant, as well as the management of their redundancy. Next we describe the operation of all of these components.

The plant is directly connected to the application of every of the replicas through dedicated Ethernet links. Sensor replicas register the state of the plant, while the actuator subsystem gathers the actuation commands sent by the replicas, consolidates them and performs the result actuation on the plant.

As concerns the control system, its operation starts with the switches broadcasting the TMs, which act as notifications for the replicas to know that the communication channel is available. At this point, replicas request to the masters the creation of all the communication resources necessary to execute their applications. More specifically, the *Initializers* of the replicas inform the masters about the size and periodicity of the messages needed to be transmitted. If there is enough bandwidth available, masters change the schedule so that the TM triggers the transmission of these messages. It is noteworthy that, for the replicas to perform their actions in synchrony, the messages associated with these actions have to be triggered in the same EC. However, there is no mechanism in FTT to specify the precise ECs in which messages have to be triggered. In order to force messages to be triggered in specific ECs, we make the Initializers to send the initialization messages in a certain order and each one in a certain instant.

After the initialization the control system starts its regular operation. More precisely, every time a replica receives a TM the *Task triggerer* is notified so that the value of its EC counter can be update. It should be noted that in this implementation we slightly modify Eq. 1 to also take into account the number of ECs used in the initialization. Specifically, as shown in Eq. 2, we introduce APP_start, which is the EC in which the initialization finishes and, thus, the application starts.

$$EC\_counter = (TM\_seqno - APP\_start) \bmod T_s \quad (2)$$

Next, we describe the actions carried out in every of the three tasks in which we have divided the application.

First, the *sense* task gathers from its associated sensor replica in the plant the last sensor value registered.

Second, the *exch+vote+ctrl* task transmits the sensor value, waits for the reception of the sensor values from the other replicas, votes on all these values and executes the control algorithm with the result of the vote. In this implementation we assume that the state of the plant can be represented as a numerical value like, for instance, the temperature in a room. These values are not expected to be identical even if sensors are operating correctly. Therefore, replicas perform a type of voting that consists in removing the outliers and then calculating the average of the resulting values.

Finally, the *exch+vote+act* task transmits the result of the control algorithm, i.e., the actuation value, waits for the reception of the actuation values from the other replicas, votes on all of them and sends the result of this vote to the actuator subsystem in the plant. As explained in Sec. II-A, the type of voting performed on the actuations considers that their values are identical and, thus, here we implemented a majority voting.

Note that, thanks to the clever initialization previously described in this section, masters generate the TMs according to the control scheme previously presented. That is, during the sense task no transmission is triggered, but during the exch+vote+ctrl and exch+vote+act tasks, masters trigger the transmission of all the sensor and actuation values respectively.

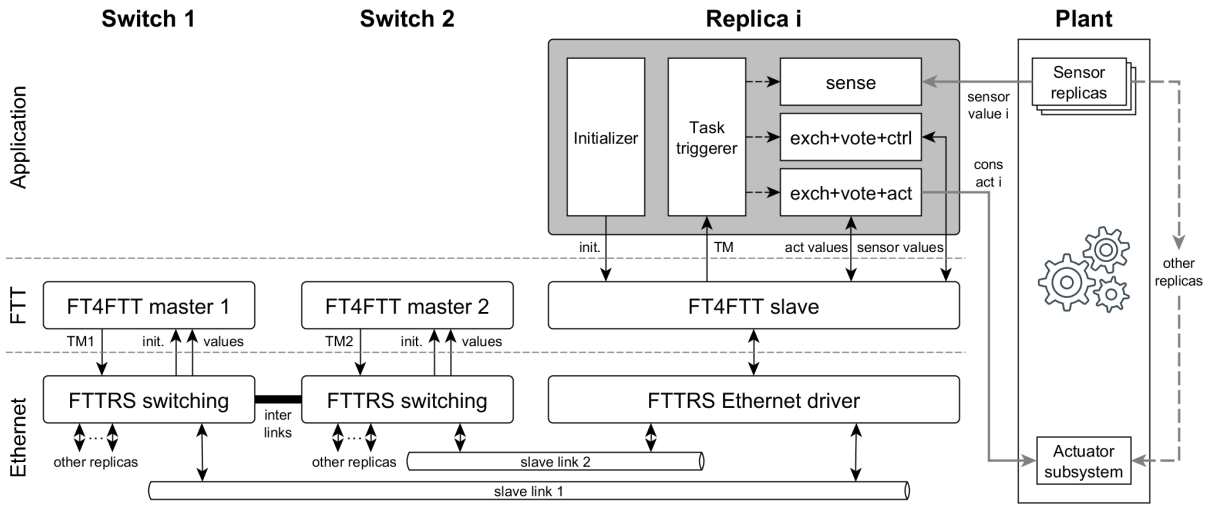

Fig. 3: Detailed architecture of the control system

IV. Testing the System

In this section we present the test campaigns we have carried out to validate the CD4NR mechanism, as well as to verify its implementation and its integration with the rest of the FT4FTT fault tolerant mechanisms.

For this we have developed a new experimental setup that allows us to test the behaviour of the system when running in a real-time environment and with mixed traffic requirements. This setup implements an inverted pendulum using the *hardware in the loop* technique. That is, all the complexity of the inverted pendulum (the plant) is simulated using Simulink, but the control system is implemented in hardware. As shown in Fig. 4, the final prototype is composed of one PC simulating the plant, two switches and three node replicas implementing the control system and two nodes exchanging a video stream.

This prototype executes three applications. First, replicas execute two PIDs controllers to control both the angle and the position of the inverted pendulum. This allows us to test the node replication. Second, replicas 2 and 3 exchange the value of a counter, whose periodicity is set by the network and can be modified online. This allows us to test the flexibility of the communications. Finally, a video server transmits a video stream to a video client. This allows to test the support of legacy nodes and the transmission of non-real-time traffic.

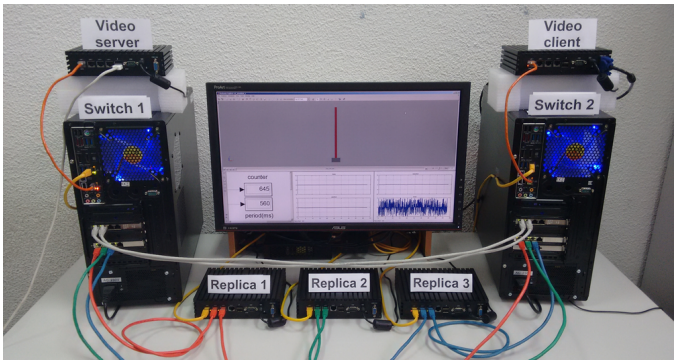

Fig. 4: Final prototype.

This setup has been used to test the tolerance of the system to permanent faults in the channel. We have carried out two test campaigns similar to the ones already carried out in [7] but with a more complete implementation of the prototype.

In the first test campaign we test the tolerance of the system to switch crashes. Specifically, we carried out two experiments each one involving the crash of one of the switches. In both experiments the system was able to continue its operation normally. Moreover, no disturbances where noticed in the control when provoking the crashes.

In the second test campaign we test the tolerance of FTTRS and the node replication to permanent faults affecting the links of the control system. More precisely, we provoke all the possible combinations of failures in the replicas' links and in the interlinks. Since the control system contains 8 links and each link can be online or offline, there are $2^8 = 256$ different error scenarios. However, for the system to work correctly, some assumptions were done in the design: at least one interlink must be online and at least a majority of the replicas (2 in this case) must be online. Consequently, we test 162 scenarios of these 256 possible scenarios. Note that, 81 of the tested scenarios affected both links of one of the replicas, i.e., they simulated the failure of said replica. In all the 162 error scenarios the system operated correctly and no disturbances were noticed in the control of the plant.

V. Conclusions and Future Work

We presented the first prototyping and testing of CD4NR, a fault-tolerant mechanism to control the dispatch of tasks and the transmission of messages in replicated distributed control systems based on FTT. The communication subsystem of the prototype is FTTRS, a fault-tolerant network based on FTT that provides a duplicated communication channel. Moreover, we developed a new experimental setup, using the hardware-in-the-loop technique, that runs a real-time control application. This setup allowed us to test the behaviour of the system when nodes have different real-time, dependability and flexibility communication requirements. The testing performed validated the design of the CD4NR mechanism and verified its implementation and integration with FTTRS.

The next steps involve finishing the design and implementation of the recovery mechanisms that make it possible for the replicas to reset and reenter into the system when they are faulty. Moreover, after that we will be able to perform a complete evaluation of the system.

Acknowledgments

This work was supported by projects DPI2011-22992 and TEC2015-70313-R (Spanish *Ministerio de economía y competividad*) and by FEDER funding. 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] P. Pedreiras and L. Almeida, "The Flexible Time-Triggered (FTT) Paradigm: An Approach to QoS Management in Distributed Real-Time Systems," *Proc. Int. Parallel and Distributed Processing Symp.*, 2003

[2] D. Gessner, J. Proenza, M. Barranco, and L. Almeida, "Towards a Flexible Time-Triggered Replicated Star for Ethernet," in *Proc. 18th IEEE Int. Conf. on Emerging Tech. and Factory Autom. (ETFA)*, 2013

[3] 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 Autom. (ETFA)*, Barcelona, 2014

[4] 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 Autom. (ETFA)*, Barcelona, 2014

[5] S. Derasevic, M. Barranco, and J. Proenza, "An OMNET++ Model to Assess Node Fault-tolerance Mechanisms for FTT-Ethernet DESs," in *Proc. 20th IEEE Int. Conf. on Emerging Tech. and Factory Autom. (ETFA)*, Luxemburg, 2015

[6] A. Ballesteros, D. Gessner, J. Proenza, M. Barranco, and P. Pedreiras, "Towards preventing error propagation in a real-time Ethernet switch," in *Proc. 18th IEEE Int. Conf. on Emerging Tech. and Factory Autom. (ETFA)*, Cagliary, 2013

[7] D. Gessner, A. Ballesteros, A. Adrover, and J. Proenza, "Experimental evaluation of network component crashes and trigger message omissions in the Flexible Time-Triggered Replicated Star for Ethernet," in *2015 IEEE World Conf. on Factory Comm. Systems (WFCS)*, 2015