A Tool for the Evaluation of the TTP/C Protocol

Roman Pallierer, Emmerich Fuchs
Institut für Technische Informatik, Technische Universität Wien,
Treitlstrasse 3/182/1, A-1040 Vienna, Austria
E-mail: {roman, emmerich}@vmars.tuwien.ac.at

March 13, 1997

Abstract

The TTP/C protocol is a fault-tolerant communication protocol designed to fulfill the stringent temporal and functional requirements of distributed hard real-time applications. To demonstrate that these requirements have been met corresponding test and evaluation techniques have to be applied. The main objective of the evaluation described in this paper is to reveal possibly existing protocol design and implementation errors rather than the evaluation of system dependability through statistical metrics.

We present a tool for testing and evaluating a recently developed TTP/C prototype by means of fault injection. The tool exploits the time-triggered approach of the TTP/C protocol minimizing the intrusiveness of the fault injection technique. Extensive analysis facilities are provided to investigate the protocol execution within every node of a cluster.

Keywords: communication protocol testing, fault injection, real-time, experimental evaluation.

1 Introduction

Recently a prototype of the Time-Triggered Protocol, Class C[1] (TTP/C) [KG94, KHK+97] has been developed to demonstrate that the protocol meets the requirements of fault-tolerant hard real-time applications. Simultaneously, an evaluation tool has been developed to test and validate the prototype implementation. The objective of this tool has been to provide extensive testing facilities for the high-level TTP/C protocol services. Similar to the work on protocol testing by Dawson et al. [DJMT96] our work has been motivated by several observations:

- Testing a protocol requires to orchestrate a distributed computation into several states, also including "hard-to-reach" ones.
- Testing the fault-tolerance capabilities of a protocol requires a certain behavior from a set of nodes that may be impossible under normal conditions. Therefore, fault injection facilities may be required to emulate some "misbehaving" nodes.
- A deterministic approach for testing a fault-tolerant protocol achieves a considerable better coverage than probabilistic (random) test runs [EC91].

This paper is structured as follows: First, an overview of the principle of operation, the design rationales and the fault model is given. Second, the current implementation of the evaluation tool is described. Thereafter, experiments that are of major interest for the TTP/C protocol and results of the performed experiments are presented. Finally, the paper concludes with a summary and an outlook on possible improvements of the tool.

2 Overview

2.1 Principle of Operation

The evaluation tool provides an interactive user interface that allows to control and monitor the protocol execution within every controller of the TTP/C cluster. Any number of controllers and various MEDL[2] data files can be configured in order to check the behavior of the protocol mechanisms. Basic fault injection methods are supported to analyze the fault-tolerant properties of the protocol. These mechanisms allow to test and evaluate the current implementation of the high-level protocol services.

2.2 Design Rationales

The design of the evaluation tool is based on two rationales: the *simulation of the hardware module* and the *simulation of the global timebase* in order to minimize the intrusiveness of our fault injection approach.

The functionality of the hardware module is simulated by a software module in order to provide extensive testing facilities of the protocol services. However, this simulation has to

[1]Class C refers to the SAE Class C requirements definition for safety-critical protocols, SAE J2056/1.

[2]MEssage Descriptor List: configuration file for each controller that determines what kind of messages are transmitted and received.

provide an adequate functionality of the hardware module for the protocol layer. Due to proper interface specification, the same source code of the high-level protocol routines can be executed either on the hardware prototype as well as within the simulated software module contained in the evaluation tool.

In the time-triggered protocol TTP/C the global timebase, established within every controller due to the clock synchronization mechanism, determines the protocol execution of the whole cluster. Within the evaluation tool this global timebase is simulated by one global clock that is accessible by every controller. The progression of the global clock can be controlled by the tool. Fault injection is only performed at the ticks of this global clock therefore the temporal internal behavior (state) of an individual controller is not affected reducing the intrusiveness of our approach.

2.3 Fault Model

The fault hypothesis of TTP/C is based on the assumption that any controller failure is a fail-silent failure and all communication failures are omission-failures. Consequently, all link, controller, and bus failures can be simulated by message mutilation failures. The evaluation tool provides the facilities to mutilate or delete messages to emulate the fault types mentioned above.

3 Implementation

The evaluation tool has been implemented on a UNIX workstation with an X11 windows system. A controller is simulated by a process, consisting of the original source code of the high-level protocol routines and a software module that simulates the hardware prototype. The message transmission and the global time base are simulated by using the interprocess communication mechanisms of the operating system. For analyzing the protocol execution, every controller continuously prints detailed debug information that may be displayed in a separate window for each process. The protocol execution can be interrupted any time – for removing or re-starting controllers (= processes), and for injecting communication failures.

3.1 Architectural Structure

The evaluation tool consists of a tool environment and a number of controllers that communicate with each other. A basic overview is given in Figure 1.

Environment: The tool environment provides the *global timebase*, the *simulated communication channels*, and the *control program.*

The *global timebase* is simulated by a global counter that is visible to all nodes. The granularity of the counter represents the action times of the protocol execution, i.e., the points in time when a transmission starts and ends. The two redundant *communication channels* of TTP/C are simulated by a shared memory containing the two frames transmitted in a TDMA[3] slot.

[3]Time Division Multiple Access

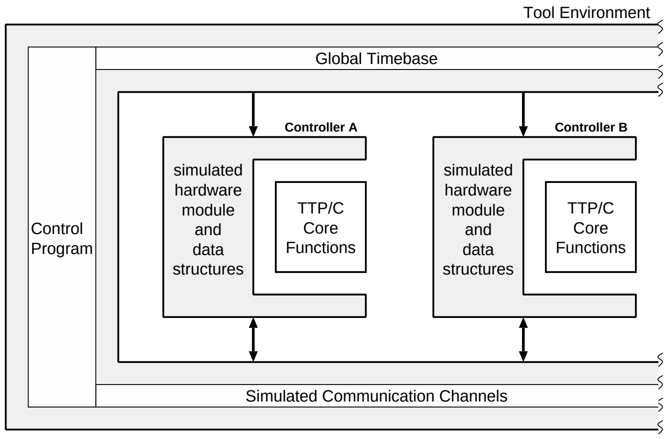

Figure 1: Structure of the Evaluation Tool for TTP/C

The *control program* allows the user to increase the value of the global timebase (= global counter). Since the protocol execution of each controller is time driven, i.e., it depends on the progression of the global timebase, the control program determines the protocol flow by incrementing the value of the global counter. Thus, the protocol execution of the whole cluster can be stopped and continued at any time. In addition, the control program has access to the shared memory in order to inject communication failures.

Controller: The original source code of the *TTP/C core functions* (= high-level protocol tasks) is embedded in a controller environment that simulates the interfaces to the hardware devices and the interfaces to the data structures of the controller. These functions of the controller environment and the original TTP/C core functions are linked, creating one process for each controller.

All functions of the controller environment print detailed debug information including the current value of important protocol variables and the function call parameters. Therefore, the protocol execution can be analyzed in detail.

4 Experiments

Within the experiments the correct behavior of the protocol implementation is checked for the major protocol services. The following scenarios are of interest:

Cold Startup: Upon a power-on or reset the controller detects no frame transmission and bus activity on both communication channels. The controller assumes it is the first to be activated and starts transmitting frames.

Integration into a running Cluster: Upon power-on or reset the controller detects bus activity. After the reception of an I-frame[4] the controller integrates itself into the cluster of

[4]Initialization Frame: is periodically transmitted and contains all necessary information a controller requires for (re)-integration into a running cluster

active controllers.

Normal Operation: After a cold start or re-integration, the controller transmits and receives frames, synchronized with the other active controllers of the cluster.

For these scenarios the following protocol activities can be analyzed:

Message Handling: Message transmission and reception according to a TDMA scheme.

Membership Service and Acknowledgment Handling: Provision of a consistent fault tolerant membership information among a set of communicating nodes.

Mode Change Handling: Support of different operational modes and consistent transitions between these modes.

Results

The experiments performed with the evaluation tool have been subdivided into two main groups. First, several scenarios and protocol activities are analyzed under the assumption that no component of the cluster fails. This set of experiments allows to check the basic operation of the protocol. It was demonstrated that a controller correctly re-integrates into a operational cluster in order to transmit and receive messages according to the TDMA scheme of TTP/C.

Second, the implementation of the fault-tolerant protocol mechanisms (message handling, membership, etc.) are analyzed by simulating failures according to the failure model and fault hypothesis of TTP/C. The experiments showed that a reliable broadcast service is established by the TTP/C protocol even in the case of transmission failures. It was also demonstrated that the ensemble of the active controllers consistently agree on the failure of a controller and the membership is updated accordingly.

During these experiments we found a number of implementation errors as well as some omissions in the protocol specification. Implementation errors included, for example, an erroneous re-integration of a node into an incorrect TDMA position, or frames assembled with wrong data resulting from MEDL access failures. Omissions of the protocol specification, on the other hand, included a too simple scheme of frame diagnosis (ok, not ok). Furthermore, an FTU must not send the same I-frames within its two succeeding TDMA slots like when sending normal data frames. Rather, it has to update the contents of the I-frame after every TDMA slot in order to specify the correct position of the TDMA sequence for re-integrating nodes. Another point in the specification that had to be clarified was the membership service, which also has to take into account the activity status of the host computer.

5 Conclusion

In this paper an evaluation tool for the TTP/C protocol was presented that provides extensive testing facilities for a recently developed prototype. A number of experiments have been performed to check the basic protocol operation and some fault-tolerance mechanisms. These test runs on a workstation have considerably reduced the number of implementation errors and helped to improve the specification. However, since the current implementation of the evaluation tool provides an interactive user interface, only a small set of relevant test runs have been performed. Extensive fault injection experiments covering a larger set of test runs are therefore still required to obtain confidence in the correct protocol behavior. Such experiments could be supported by an extension of the evaluation tool that allows the script-driven execution of a large number of automatically generated experiments. This offers additionally an efficient opportunity to run regression tests even on minor changes of the protocol.

References

[DJMT96] S. Dawson, F. Jahanian, T. Mitton, and T.-L. Tung. Testing of fault-tolerant and real-time distributed systems via protocol fault injection. In *Proc. 26th Int. Symposium on Fault-Tolerant Computing*, pages 404–414, Sendai, Japan, June 1996. IEEE Computer Society.

[EC91] K. Echtle and Y. Chen. Evaluation of deterministic fault injection for fault-tolerant protocol testing. In *Proc. 21st Int. Symposium on Fault-Tolerant Computing*, pages 418–425, Montreal, Canada, June 1991. IEEE Computer Society.

[KG94] H. Kopetz and G. Grünsteidl. TTP — A Protocol for Fault-Tolerant Real-Time Systems. *IEEE Computer*, pages 14–23, January 1994.

[KHK+97] H. Kopetz, R. Hexel, A. Krüger, D. Millinger, R. Nossal, A. Steininger, Ch. Temple, T. Führer, R. Pallierer, and M. Krug. A Prototype Implementation of a TTP/C Controller. In *Proc. of the SAE Congress 1997*, Detroit, MI, USA, Feb. 1997. Society of Automotive Engineers, SAE Press. SAE Paper No. 970296.