FT-RSS: A Flexible Framework for Fault Tolerant HLA Federations

Johannes Lüthi[1] and Steffen Großmann[2]

[1] University of Applied Sciences FHS KufsteinTirol
Business Informatics, Kufstein, Austria
Johannes.Luethi@fh-kufstein.ac.at
[2] German Army Signal School
Feldafing, Germany
steffengrossmann@bundeswehr.org

Abstract. The absence of fault tolerance mechanisms is a significant deficit of most current distributed simulation in general and of simulation systems based on the high level architecture (HLA) in particular. Depending on failure assumptions, dependability needs, and requirements of the simulation application, a choice of different mechanisms for error detection and error processing may be applied. In this paper we propose a framework for the configuration and integration of fault tolerance mechanisms into HLA federates and federations. The administration and execution of fault tolerant federations is supported by the so-called fault tolerant resource sharing system based on a ressource sharing system previously developed by the authors.

Keywords: Dependable distributed simulation, fault tolerance, high level architecture, resource management

1 Introduction

In its current state, no formal failure model is included in the High Level Architecture (HLA) [1]. Services of the runtime infrastructure (RTI) such as e.g. the save/restore services may be used to implement fault tolerant federations. However, failure detection as well as failure processing and all corresponding management acivities are left to the federation developer. Only little effort has been made in combining *fault tolerance* (FT) mechanisms with distributed simulation. Exceptions include the work of Agrawal and Agre proposing replicated objects in time warp [2], and the optimistic fault tolerant simulation approach presented by Damani and Garg [3]. An overview of possibilities for fault tolerant distributed simulation can be found in [4]. Anyway, these approaches have only restricted applicability to distributed simulation systems based on the HLA.

There is a wide spectrum of dependability enhancing mechanism that may be integrated into distributed simulation. However, there is also a wide spectrum of different types and application domains for HLA-based distributed simulation. Depending on the specific dependability, realtime, repeatability, and other requirements, and depending on the failure assumptions for the federates and the

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3038, pp. 865–872, 2004.

computing nodes, an appropriate choice of FT mechanisms has to be used. Moreover, not every federate of an HLA federation will have the same requirements with respect to dependability and fault tolerance. In this paper, we present a framework for the development, administration, and execution of fault tolerant HLA simulations. The proposed framework is based on the *resource sharing system* (RSS), presented in [5]. Hence it is called *fault tolerant RSS* (FT-RSS). The framework and its components allow the individual configuration of FT mechanisms to be included in federates and federations. In the design and implementation of a fault tolerant federation, the developers are supported by an FT configuration tool. Furthermore, in its final state the FT-RSS will provide code modules and interfaces for error detection and error processing functions and associated HLA interactions. During the execution of a distributed HLA simulation using the FT-RSS, the manager and client components of the FT-RSS are responsible for the enforcement of FT mechanisms.

The paper is organized as follows. In the next section, the general concept and architecture of the framework is presented. The configuration options for HLA federates are discussed in Section 3. Federation configuration is considered in Section 4. The work is summarized in Section 5.

2 Concept for Configurable Fault Tolerance

2.1 Introducing the Framework

The objective of the proposed framework is to support a flexible configuration of HLA federates and federations with respect to fault tolerance mechanisms. The three main components of the proposed framework are the *FT configurator*, a set of *FT code modules and interfaces*, and the *fault tolerant resource sharing system* (FT-RSS):

- During the design and implementation of a federation, FT requirements determine the choice of error detection and error processing[1] mechanisms for the federates. The configuration tool can be used to select the appropriate options for these mechanisms. The developers of the federates are supported with code segments and interfaces corresponding to the selected FT mechanisms. The developer is responsible to integrate these mechanisms into the HLA federates.
- Depending on the available computing resources, prior to simulation runs at the target environment, the migration configuration of an FT HLA federation is specified. For that purpose, again the configuration tool can be used.
- The federation with integrated FT mechanisms can be run using the FT-RSS environment. An FT monitor can be used to visualize detected errors and initiated error processing mechanisms.

[1] In our terminology, error processing includes error recovery, error passivation, and error compensation.

2.2 Architecture of the FT-RSS

The proposed architecture of the runtime environment for configurable fault tolerant simulation of HLA federations is based on the *Resource Sharing System (RSS)* presented in [5]. The RSS is designed to allow users of workstations to control the participation of their computers in an HLA-based distributed simulation. In the original RSS architecture, the *RSS manager* keeps track of participating workstations and simulation federates that are registered in the federation. *RSS clients* at the workstations allow the users to connect or disconnect their workstation to the RSS. Whenever a node connects or disconnects to the RSS, federates may be migrated. The migration operations are controlled by the manager. Migration specific communication between the manager and simulation federates is performed via a special *communication federate*. This way, non-RTI communication of the simulation federates can be avoided. A separate ftp server is used as a persistent storage system. This may not be the most efficient solution for that purpose. However, this way dependability issues of that aspect need not be considered in the FT-RSS itself. In order to provide a runtime environment for configurable fault tolerant distributed HLA simulation, the architecture of the RSS is extended in several ways (see Fig. 1).

- A configuration file generated by the configurator *FT Config* tells the FT-RSS manager the fault tolerance configuration of every federate as well as the configuration of the whole federation.
- The RSS manager is extended to control error detection, error processing, and migration mechanisms according to the configuration.
- The functionality of the clients is extended to support checks required for error detection and methods for error processing. In contrast to the RSS, this may also include non-RTI-based communication with the federates. Since no simulation specific information is exchanged this is not a violation of the corresponding HLA rule.
- An *FT monitor* is introduced to allow observation of error/failure behavior of the simulation federation and measures taken by the FT-RSS.

3 Federate Configuration

The configuration of FT options can be divided into two types: (a) aspects that can be configured for individual federates and (b) FT configuration of the whole federation. In this section, the configuration options for FT properties of single simulation federates are discussed. Three classes of options are considered: configuration of migration possibilities, configuration of error detection mechanisms, and configuration of error processing methods.

3.1 Migration Configuration

For some error processing methods, federates have to be migrated to another node. There are many situations where a configuration of migration options

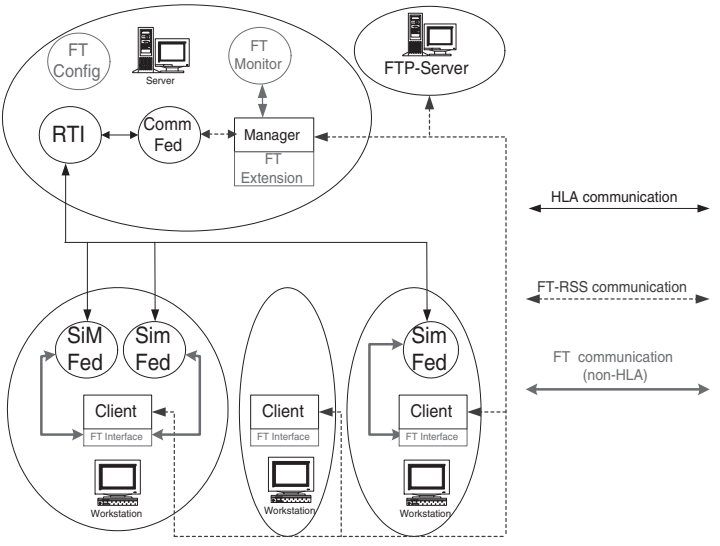

Fig. 1. Architecture of the fault tolerant resource sharing system (FT-RSS).

is required. Consider for example interactive simulation: only nodes that are within reasonable distance and equipped with appropriate I/O devices may be considered for migrating a federate with human interaction. More restrictions can be imposed by heterogeneous hardware and operating systems (OS). In the proposed FT configurator, three means to specify the migration behavior of a simulation federate are available:

- In a *node pool*, IP addresses of nodes the federate may be migrated to are listed. Also address ranges and sub-networks may be specified.
- A list of *restrictions* – again provided as addresses, host names and/or address ranges or sub-networks – can be used to prevent the federate from being migrated to certain computers.
- In a list of *OS assignments*, the operating systems the federate can be executed on can be specified.

3.2 Error Detection Configuration

In the federate configuration, only detection of errors in simulation federates are taken into account. The detection of errors in computing nodes are considered in the federation configuration (see Section 4). A choice of four basic options is considered for error detection (see Fig. 2):

- *No error detection*: if a federate is assumed to be highly dependable or some reasons make the use of error detection mechanisms for that federate not practicable, a federate can be excluded from error detection. Note that the error *processing* behavior may still be configured for that federate because the configuration may be used in the case of failures in computing nodes.

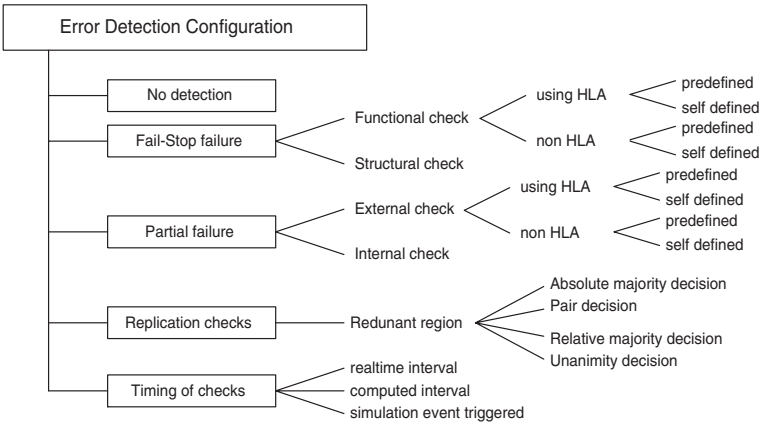

Fig. 2. Federate configuration options for error detection.

- *Fail-Stop failures*: for the detection of fail-stop failures *functional checks* or *structural checks* may be applied. In a functional check, a prespecified function of the federate is called at certain times. If the expected function results are provided by the federate, it is assumed to be running fault-free. The function used in a functional check may be an HLA interaction or it may be called by the RSS-client directly. For both cases, predefined functions may be used that are provided in the set of *FT code modules and interfaces*. As an alternative, the developer of the federate may specify the function to be used. For HLA internal function checks, the communication between the manager (triggering such a check) and the federate is performed via the communication federate.

  Structural checks are triggered by the manager and are performed by the local FT-RSS client. They use structural information about the federate (e.g., task lists provided by the OS) to check whether the process associated with the federate is still active. Since functional checks can be performed outside the federate, they can be provided by the FT-RSS. No additional implementation effort by the simulation developer is required.
- *Partial failures*: if failures in separate parts of the federate are considered in the failure assumption, such partial failures can be detected by appropriate functional or structural checks. However, since a partition of a federate into several failure regions is typically internal to that federate (for example separate threads that may fail independently), structural checks for partial failures have to be performed internally by the federate (for example by examination of a thread list). External functional checks to test for partial failures offer the same options as functional checks for fail-stop failures.
- *Replication checks*: replication can serve as a sufficient tool for both, error detection and (if more than two replicates exist) error processing. A number of federates can be grouped as a redundant region. A choice of options for

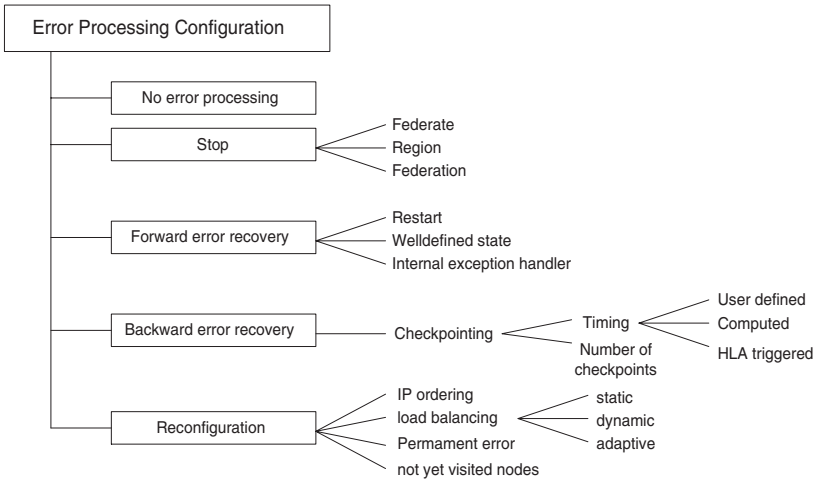

Fig. 3. Federate configuration options for error processing.

error detection is available in the configurator: absolute and relative majority decision, pair decision, and unanimity decision.

The FT-RSS manager is responsible to trigger checks for fail-stop failures and external checks for partial fail-stop failures. In these cases, timing of the checks must be specified. Three options are proposed for that purpose: A fixed realtime interval, an adaptive checkpointing interval, and checkpoints triggered by prespecified simulation events.

3.3 Error Processing Configuration

Various options are also available for error processing (see Fig. 3):

- *No error processing*: errors may be detected but no FT mechanism is applied.
- *Stop*: a failed federate is stopped without further FT mechanisms. Three variants are available: (a) only the failed federate may be stopped. This may for example be useful if the federate of just one trainee in a joined training simulation fails and it is reasonable that all others can continue the training program. (b) a sub-model consisting of several federates may be stopped. (c) the whole federation may be stopped.
- *Forward error recovery*: depending on the functionality of the federate, it may not be necessary to use a saved checkpoint to restart a failed federate. This class of methods is referred to as forward error recovery. The options in this case are: (a) restarting the federate with its initial state, (b) using a specified valid state, and (c) evoking an internal exception handler within the federate to obtain a valid state. In any case, the responsibility to consider time management conflicts with forward error recovery is left to the federate/federation developer.

– *Backward error recovery*: restarting a federate using a previously saved state is referred to as backward recovery. For backward recovery the checkpointing behavior has to be specified. Timing for checkpoints can be chosen to be (a) a realtime interval, (b) automatically computed using an underlying failure model, or (c) triggered by certain simulation events. Solving or avoiding time management conflicts is a crucial problem when using backward recovery. In strictly optimistic distributed simulation, backward recovery can be implemented by deploying timewarp and its anti-message mechanism [3]. However, locally decreasing simulation time is not an available HLA time management service.

If forward or backward recovery is chosen for error processing, the federate has to be restarted after a failure. As a default, every failure is assumed to be transient. Thus, a failed federate is restarted at the same node. However, in the error processing configuration it can be specified that every failure is assumed to be permanent. This means that in a failure situation the federate is migrated to another computing node before restart. As an additional option it can be specified that in the case of failure-driven migration, a federate has to be migrated to a node it has not yet been executed on in the current run.

4 Federation Configuration

In addition to configuration options for HLA federates, some aspects concerning FT behavior of the whole federation have also to be configured. The HLA definitions do not specify the behavior of federates and of the RTI in failure situations. For example, some RTI implementations (e.g., DMSO's RTI NG [6]) fail to work properly after a fail-stop failure of one of the federates whereas other RTI implementations (e.g., portable RTI [7]) are able to treat a failed federate as if it had resigned from the federation. Such differences have to be considered in the mechanisms used by the FT-RSS. Thus, an RTI in a list of versions available for the FT-RSS has to be specified in the federation configuration.

Also hardware failures of participating computing resources should be detected and processed (here, only fail-stop failures are considered). In the federation configuration, options for functional and/or structural checks for hardware failures as well as associated timing behavior can be specified. When an error of a computing node is processed, every federate running on that node is treated as if a failure of the federate had occurred.

Detection and processing of RTI fail-stop failures is also included in the federation configuration. Structural as well as functional checks can be applied for error detection. Stopping and backward recovery may be used for error processing of RTI failures. A consistent internal state of the RTI after a failure can only be reached with the HLA service *federation restore*.

There are several reasons why error processing measures required in a fault tolerant distributed simulation using the FT-RSS may fail. For example a node that conforms to a necessary migration operation or resources (e.g. files) required for an error processing measure may not be available. For such situations it

is important to include an *exception handling* mechanism in the FT-RSS. A basic feature of FT-RSS exception handling is to report the situation to the simulation user and to stop the simulation run to avoid propagation of incorrect or misleading simulation results.

5 Summary and Conclusions

Although failed simulation runs may be safety critical and costly, currently most distributed simulation systems restrict fault tolerance measures to the use of dependable hardware. The requirements with respect to dependability, interactivity, repeatability, and timing behavior varies not only from one federation to another but even among the federates of a single federation. An appropriate selection of mechanisms to support fault tolerance has to be implemented. In order to achieve a flexible FT configuration of HLA-based simulations we propose a framework including a *fault tolerant resource sharing system* (FT-RSS). The FT-RSS allows the configuration of migration, error detection, and error processing for individual federates and for the federation. Prototypes of the configuration tool and the FT-RSS runtime environment have been implemented. Future work includes the implementation of FT federations, extension of the FT-RSS implementation with respect to the number of FT methods, and research in the direction of replication approaches for HLA federates.

References

1. Dahmann, J.S.: The high level architecture and beyond: Technology challenges. In: Proc. 13^{th} Workshop on Parallel and Distributed Simulation (PADS '99, Atlanta, GA, USA), IEEE CS Press (1999) 64–70
2. Agrawal, D., Agre, J.R.: Replicated objects in time warp simulations. In: Proc. 1992 Winter Simulation Conference, SCS (1992) 657–664
3. Damani, O.P., Garg, V.K.: Fault-tolerant distributed simulation. In: Proc. 12^{th} Workshop on Parallel and Distributed Simulation (PADS '98, Banff, Alberta, Canada), IEEE CS Press (1998) 38–45.
4. Lüthi, J., Berchtold, C.: Concepts for dependable distributed discrete event simulation. In: Proc. Int. European Simulation Multi-Conference (ESM 2000, Ghent, Belgium), SCS (2000) 59–66
5. Lüthi, J., Großmann, S.: The resource sharing system: Dynamic federate mapping for HLA-based distributed simulation. In: Proc. 15^{th} Workshop on Parallel and Distributed Simulation (PADS 2001, Lake Arrowhead, CA, USA). IEEE CS Press (2001) 91–98
6. DMSO: RTI Next Generation Programmer's Guide Version 3.2. Defense Modeling and Simulation Office (DMSO), Alexandria, VA, USA (2000)
7. Pitch Kunskapsutveckling AB. http://www.pitch.se. validated: December 2003.