Chapter 3

Fault tolerant ERTS framework

This chapter presents our proposal of a fault tolerance ERTS framework consisting of three layers. Its objective is to define a specification for the fundamental aspects from where faults can occur (network, communication, and intra-nodes operation). It is organized as follows; Section 3.1 presents an overview of the three framework layers. Section 3.2 describes the network layer; it includes the specification of the control unit and the communication controller that constitutes an ERTS node, and the specification of a model for the messages used on ERTS. Section 3.3 describes the communication layer; it includes a definition for the time-triggered, event-triggered, and mixed-triggered bus types that enable a reliable communication. Section 3.4 describes the nodes layer; it includes a definition for the AAA method with the fault tolerance heuristic. Finally, section 3.5 discusses about solutions presented in previous sections.

3.1. Framework layers overview

In software development, a framework is a defined support structure in which another software project can be organized and developed. A framework may include support programs, code libraries, a scripting language, or other software to help develop and glue together the different components of a software project. Frameworks are designed with the intent of facilitating software development, by allowing designers and programmers to spend more time on meeting software requirements rather than dealing with the more tedious low level details of providing a working system.

The framework that we propose is a set of abstract classes that covers the three principal aspects, which the previous presented state-of-the-art shows to us that can be fault focuses. For that reason the three principal layers of the fault tolerance ERTS framework are the network layer, the communication layer, and the intra-node layer (each layer is an abstract class). Figure 3.1 illustrates the general view of the principal layers of the fault tolerance ERTS framework.

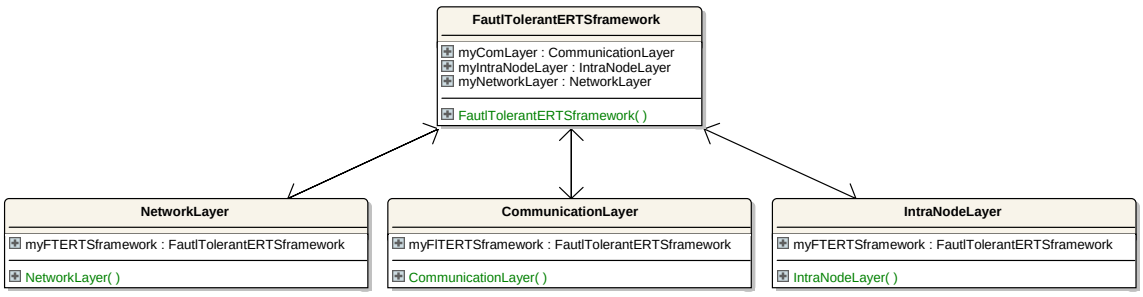

Figure 3.1 Fault tolerance ERTS framework

Each of the three principal layers has other associated sub classes that serves for the implementation of their corresponding layer into an ERTS architecture. The definition of an ERTS with the fault tolerance ERTS framework ensures that the target ERTS is fault tolerant by itself without the necessity of integration of another external mechanism. At the network layer the fault tolerance consist on the correct construction and interpretation of the transmitted frames, at the communication layer the fault tolerance consist on the concurrency control on the message passing between the ERTS component, and on the intra-node layer the fault tolerance consist on the monitoring and replacement of the fail-silent ERTS components. Next are the complete definition of the three layers.

3.2. Network layer

The objective of the network layer is to specify a network model that includes the necessary classes for defining a broadcast communication over a bus channel, considering that the communication is realized under an asynchronous model such a condition necessary for the ERTS because communication must be done in real-time. Figure 3.2 illustrates the general view of the network layer.

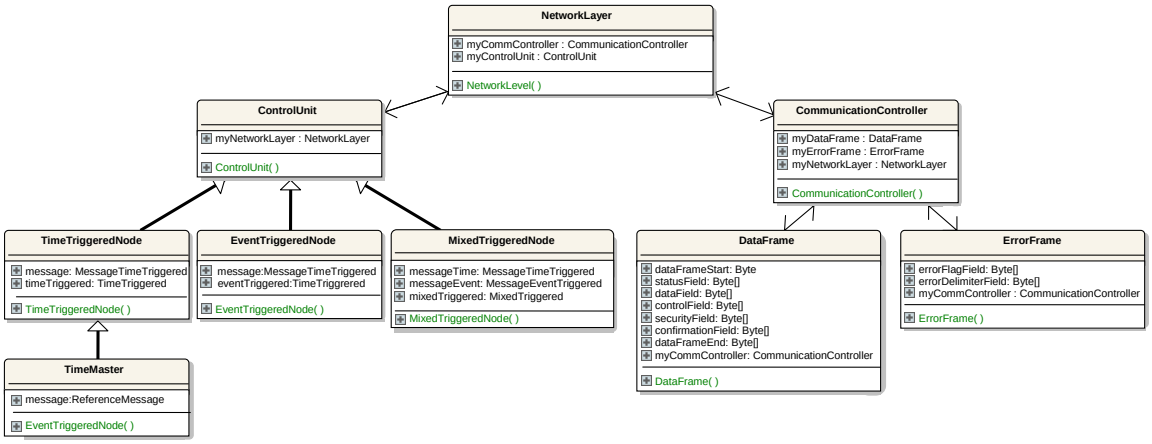

Figure 3.2 Network layer

The previous figure illustrates those aspects of the ERTS architecture that are modeled by the network layer. These aspects are the necessary elements that constitute an ERTS node in order to implement the desired communication. The elements that control the message-passing over the bus are the control unit and the communication controller. But before defining both in particular it is necessary to model the messages that can take part in an ERTS.

3.2.1. Message model

The message model represents the kind of messages that can be sent and received within an ERTS. Messages can be of time-triggered, event-triggered or reference type. Figure 3.3 illustrates the message model.

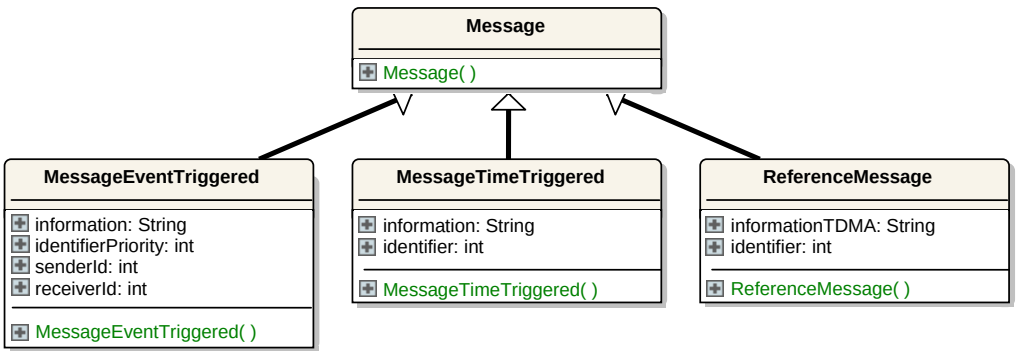

Figure 3.3 Message model

Time-triggered message type

The messages designed for a time-triggered communication are denominated time-triggered message. This type of messages contains a string with the information to transmit and a unique identifier. The length of the string depends of the predefined bits space of the data frame defined for the message-passing.

Event-triggered message type
The messages designed for an event-triggered communication are denominated event-triggered messages. This type of message contains a string with the information to transmit, a unique identifier, and information about the sender and the receiver. The identifier also represents the priority of the task associates to the message.

Reference message type
The messages designed for a reference communication is denominated reference messages. Message of this type is contains a string with the information to transmit and a unique identifier. The information corresponds to the characteristics of the TDMA round that it is referencing, in particular with the global time information of the ERTS. The identifier is a special one that indicates that it is a reference message. Reference messages are of interest for all nodes.

3.2.2. Control unit

The control unit class contains applicative code according to the functions that an ERTS node must implement. It executes operations that have as parameters the values received by sensors, and its objective is to compute responses that have to be transmitted as messages along the bus. Control unit defines two types of communication with the communication controller, one for the transmission of the message that is going to be packaged in the data frame, and other for the reception of a message with information originating from another one node. The control unit estimates the possibility of sending and re-sending (in case of error detection) messages at a predefined point in time. Estimation is done according to three approaches for concurrent control communication time-triggered, event-triggered, and mixed-triggered. Figure 3.4 illustrates the control unit aspect.

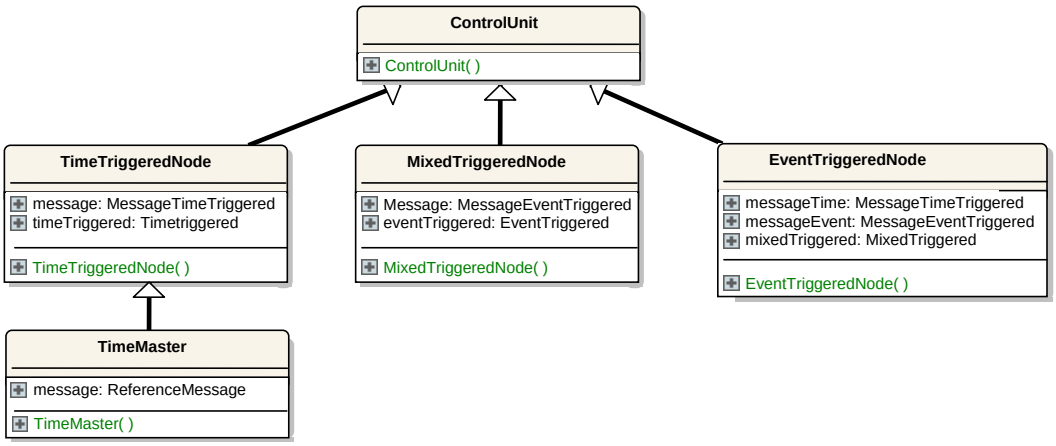

Figure 3.4 Control unit aspect

The control unit class is extended to define classes according to the particular selected communication bus. The sending and receiving messages policy depends of the particular approach:

- Time-triggered node class (see figure 3.4): It has the faculty to create time-triggered messages from processed information that it is going to transmit to other nodes. For the transmission control it maintains a schedule table with the corresponding times for sending and receiving messages. At the reception of a message a time-triggered node identifies if the message is directed to it. Such identification is based on the information of the schedule table.

- Time master node class (see figure 3.4): It is an extension of a time-triggered node with the same functional characteristics of sending time-triggered messages, but additionally, it has

the faculty to create reference messages that it sends at determined time to notify the beginning of a communication cycle. It manages two types of schedule tables, one for the normal time-triggered tasks operation and another, called reference table, which contains the predefined moments when a reference message has to be sent. It always defines the ERTS global time and helps other nodes to accurately estimate it.

- Event-triggered node class (see figure 3.4): It has the faculty to create event-triggered messages from processed information that it is going to transmit to other node. For the transmission control it maintains a dynamical schedule table with the ERTS massages (ordered by priority) that are going to be sent. At the reception of a message an event-triggered node identifies if the message is directed to it. Identification is done based on the dynamical schedule table which specifies the sender and the corresponding receiver of each message.

- Mixed-triggered node class: It has the faculty to create both time-triggered and event-triggered messages from processed information that it is going to transmit to other node, it can act like a time-triggered or like and event-triggered node at the same time. For the transmission control it maintains a schedule mixed table that defines the fixed corresponding times for sending and receiving time-triggered messages, and with event-triggered massages that are going to be send ordered by its priority.

3.2.3. Messages association

Control unit class has different associations with the message types, which constitute the message model, depending on the control unit. Figure 3.5 illustrates the relation among the message model and the control unit types.

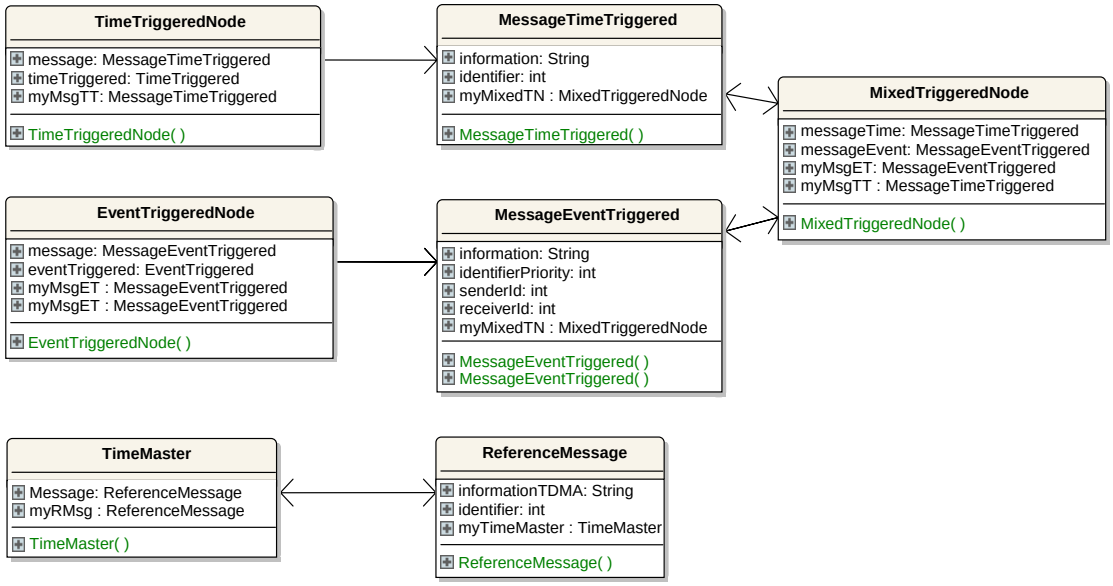

Figure 3.5 Messages relation

The figure 3.4 illustrates the relation if which nodes can send and receive the different messages types. A time-triggered node has a relation with the time-triggered message type. An event-triggered node has a relation with the event-triggered message type. A mixed-triggered node has a relation with both types of messages, time-triggered and event-triggered. Finally, a time master node has a relation with the reference message type.

3.2.4. Communication controller

The communication controller class server as interface that connects an ERTS node with the bus cable. Its objective is to produce data or error frames with the received message from the control unit. The message is seen as an atomic unit. Its inclusion on the frame is the same independently of message types (time-triggered, event-triggered, or mixed-triggered).

The transmission of information over the physical bus is realized in the form of electrical signals that represent the data or error frame produced. Analogously way, when electrical signals are detected on the physical bus, they are read and transformed into data or error frames. Figure 3.6 illustrates the communication controller aspect.

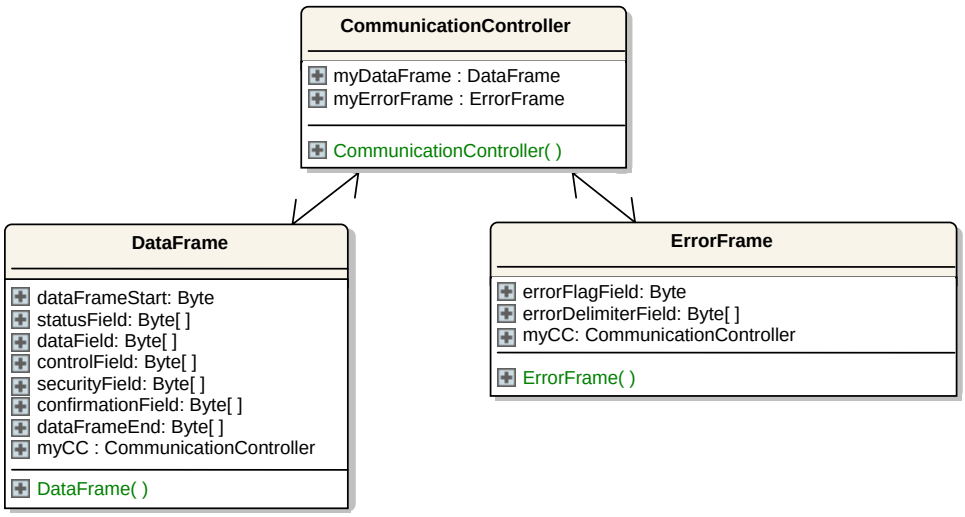

Figure 3.6 Communication controller aspect

The figure 3.5 illustrates the relation of the communication controller with both kinds of frames classes. Data frame structure includes seven fields: data frame start, status, control, data, security, confirmation, data frame end. The error frame structure includes two fields: error flag and error delimiter.

The operation when a data frame is received from another ERTS node is to verify its structure with the purpose of detect the existence of any of the five error types bit, bit stuffing, checksum, form, or acknowledgement. If there are no errors, it extracts the message included in the data frame and transmits it to the control unit. If there are errors, it produces an error frame with the detected error notification and sends it back to the data frame sender. The operation when an error frame is received from another node, with the notification of one error detection in a data frame that it previously sent, is to extract the type of error and transmit it to the control unit requesting the re-execution of the task that produces the erroneous message.

3.3. Communication layer

The objective of the communication layer is to specify a communication model that includes the necessary classes to solve the concurrency problem originated from the message-passing between ERTS nodes. It assumes that the data and error frame transmission between nodes is successfully controlled by the network layer. It does not consider the architecture physical model of the ERTS. Figure 3.7 illustrates the general view of the communication layer.

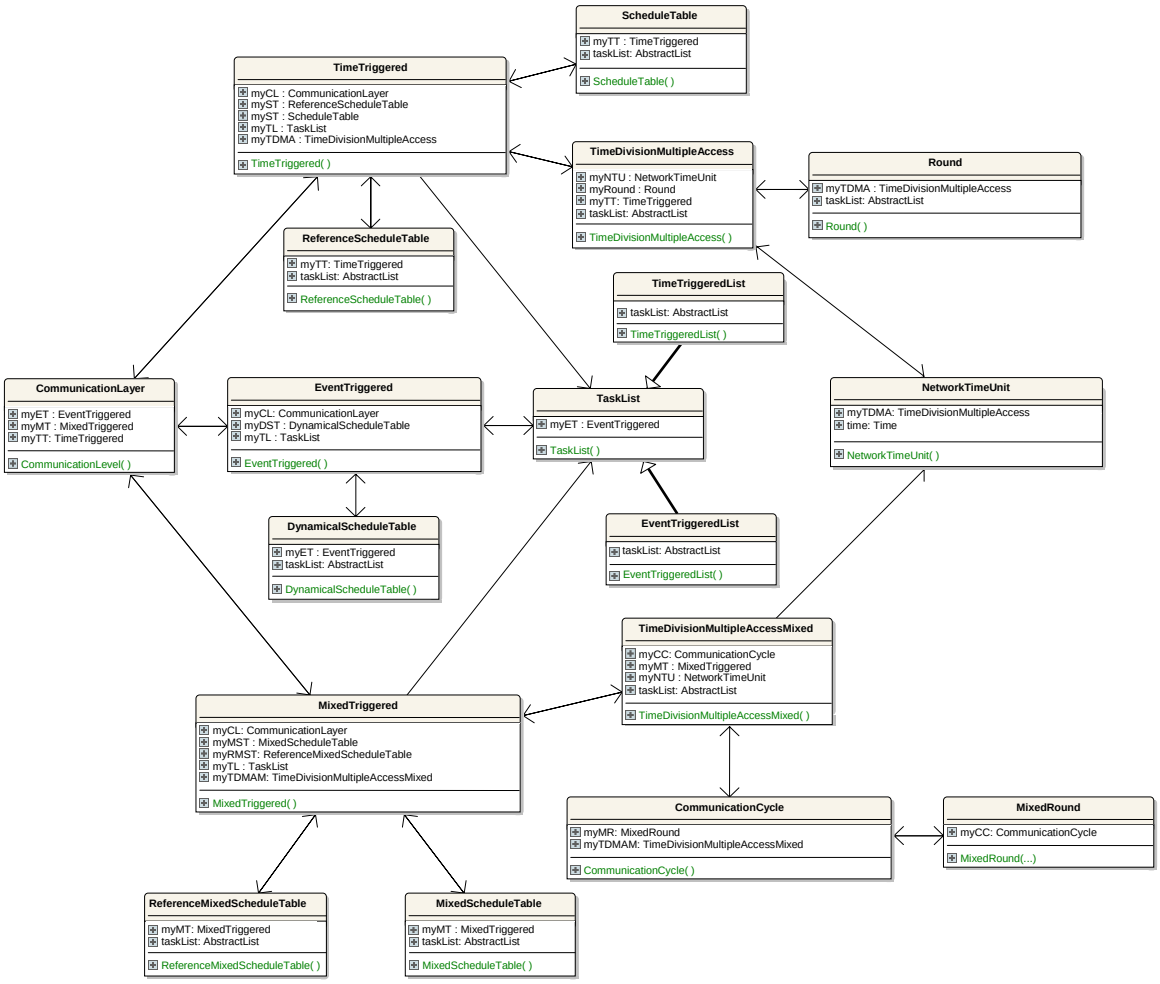

Figure 3.7 Communication layer

The figure 3.6 illustrates those classes that must be considered by an ERTS in order to implement any of the three bus types: time-triggered, event-triggered, or mixed-triggered. The aspects define the policies in which all ERTS nodes are based to send and receive messages during the ERTS execution time.

Before defining the three bus types in particular, it is necessary to consider that no matter the selected bus type, a particular scheduling mechanism must be included in order to make an order execution relation among tasks. The scheduling model includes the scheduling strategies that can be applied in ERTS.

3.3.1. Scheduling model

The scheduling model represents the kind of schedule strategy that can be considered within an ERTS. The parameters for the scheduling mechanisms are located in a task list class (scheduling table) that has the fundamental properties of the entire tasks included in the ERTS. The task list structure depends of the particular bus type. Figure 3.8 illustrates the scheduling model.

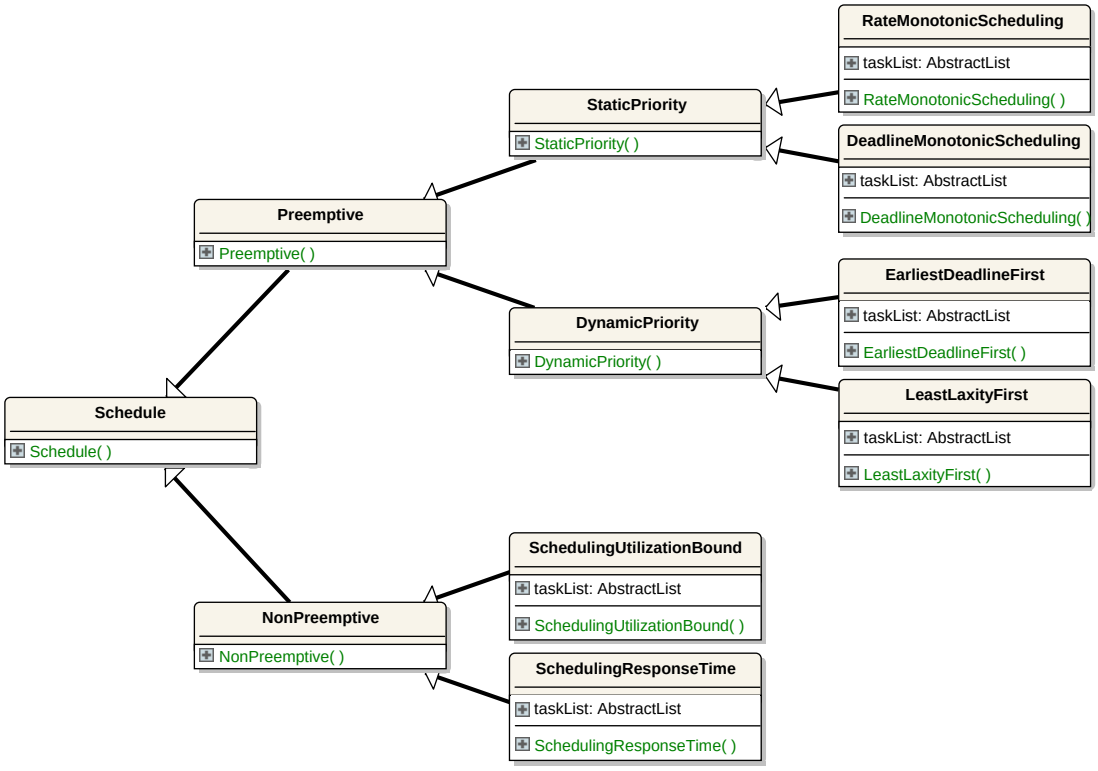

Figure 3.8 Scheduling model

In order to preserve the real-time constraints of every ERTS task, the scheduling model is divided in two major scheduling types: preemptive and non-preemptive scheduling. The first one is well suited for event-triggered buses while the second one is better suited for time-triggered ones.

Preemptive scheduling type
It is subdivided according to the tasks priorities classification into scheduling for static or dynamic priorities. Dynamic Monotonic Scheduling [ABRW1991] and Rate Monotonic Scheduling [LSD1989] are included as policies for the estimation of static priorities, while Earliest Deadline First [RR1996] and Least Laxity First [HGT1999] [RR1996] are included as policies for the estimation of dynamic priorities.

Non-preemptive scheduling type
It includes to policies for the scheduling of non-preemptive tasks, the Scheduling Utilization Bound and the Scheduling Response Time.

3.3.2. Time-triggered

The time-triggered bus class is composed by three sub classes: Time Division Multiple Access, time-triggered list, and the schedule table. Figure 3.9 illustrates the time-triggered bus class.

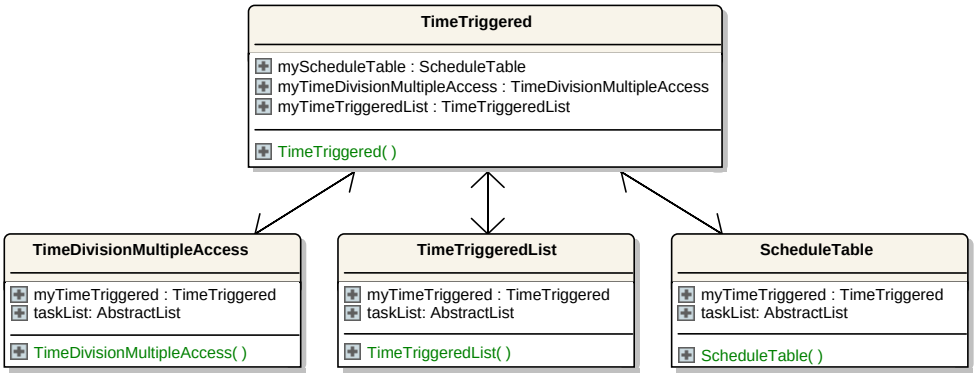

Figure 3.9 Time-triggered approach

The time-triggered bus class bases it operation on a fixed schedule table which orders the ERTS tasks according to the estimation of the time when they must be executed. In order to assign the necessary transmission time over the bus to each task, it defines a time division multiple access protocol.

Time division multiple access
It fragments the bus access time with the purpose of provide the required time to each node of an ERTS for the execution of its tasks. Two classes are necessary to define the time division multiple access: Network time unit class and round definition class. Figure 3.10 illustrates the classes for its definition.

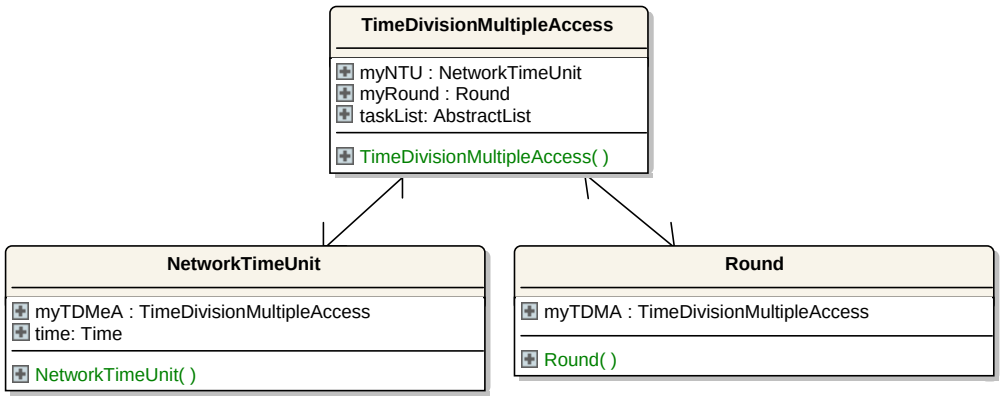

Figure 3.10 Time division multiple access elements

The network time unit represents the granularity of the network access time (for example twenty milliseconds). The network time unit is the minimal unit in which time can be fragmented, in other words, a fragment can measures from one to many network time units. Time fragments are denominated slots.

The set of slots that match one by one with every ERTS node is denominated a round. Rounds are repeated continuously with the purpose of provide a periodic execution time assignation to the nodes of an ERTS.

Time-triggered list
It is a model for the specification of the entire tasks that are included on an ERTS. It allows the

definition of a task list, when the description of every task includes the same subset of the fundamental real-time tasks properties that are of importance in the time-triggered bus approach. The useful real-time tasks properties are worst-case computation time, period, dead-line, and worst-case response time.

The order in which tasks are located into the task list does not represent their execution order. To define an execution order that satisfy the particular real-time constraints of the entire tasks it is necessary to process the task list with a schedule strategy that computes a fixed schedule table.

Schedule table
It is a class that contains a table with a fixed structure that represents a schedule for all tasks included in an ERTS. It can be generated from any of the two scheduling policies included in the non-preemptive scheduling type (scheduling utilization bound and scheduling response time), which comprise the schedule model. The real-time task properties that are defined in the time-triggered schedule table are the necessary parameters for the application of any of the scheduling policies defined for the time-triggered bus approach.

Reference schedule table
It is class that contains a table with a fixed structure that represents a schedule for tasks that correspond to the sending of reference messages. A reference message is sent at the beginning of every round, in a time division multiple access protocol, to indicate to each node that a cycle has begun. There is no need to use a complex scheduling policy, reference tasks are executed according to predefined time intervals and its execution does not interfere with the execution of other reference tasks.

3.3.3. Event-triggered

The event-triggered bus class is composed by two sub classes: Event-triggered list and the dynamical schedule table. Figure 3.11 illustrates the time-triggered approach.

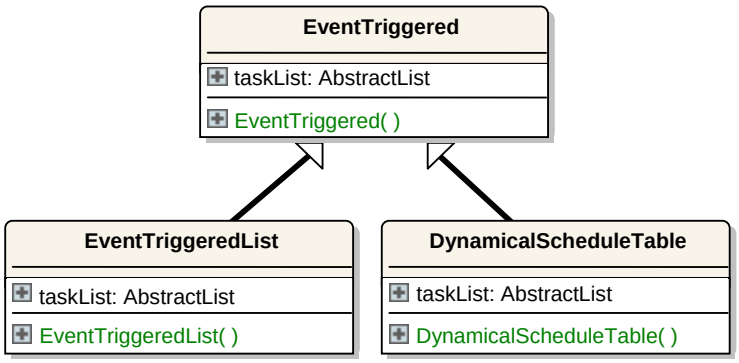

Figure 3.11 Event-triggered approach

The event-triggered bus bases it operation on a dynamical schedule table which orders the ERTS tasks according to the estimation of the time when they must be executed, the estimation is based on the assignment of priorities to tasks.

Event-triggered list
It is a model for the specification of the tasks of an ERTS. It allows the definition of a task list, when the description of every task includes the same subset of the fundamental real-time tasks properties that are of importance in the event-triggered bus approach. The useful real-time tasks properties are, worst-case computation time, period, dead-line, worst-case response time, and remaining time from completion time to dead-line.

The order in which tasks are inserted the task list does not represent their execution order. To define an execution order that satisfies the particular real-time constraints of the whole system

if is necessary to process the task list with a schedule mechanism that gives a dynamical schedule table as result.

Dynamical schedule table
It is a class that contains a table with a fixed structure that represents a schedule for all tasks included in an ERTS. The table schedules the tasks according with the estimation of their associated priorities, leaving first the task with higher priority of execution.

The priorities are calculated from any of the four scheduling policies included in the preemptive scheduling type which is part of the schedule model. Two scheduling policies are well suited for tasks that require static priorities (Rate Monotonic Scheduling and Dynamic Monotonic Scheduling) and the other two are well suited for tasks that require dynamic priorities (Earliest Deadline First and Least Laxity First).

There is no a priori knowledge of the moment in which a task is going to be executed for a node, for that reason the schedule table is considered dynamical because to allow the inclusion of new tasks without affecting the calculated execution time from the already scheduled ones. The real-time task properties that are defined in the event-triggered list are the necessary parameters for the application of any of the scheduling policies defined for the event-triggered bus approach.

3.3.4. Mixed-triggered

The mixed-triggered bus is composed by four classes: TDMA mixed, time-triggered list, event-triggered list, and the mixed schedule table. Figure 3.12 illustrates the mixed-triggered class.

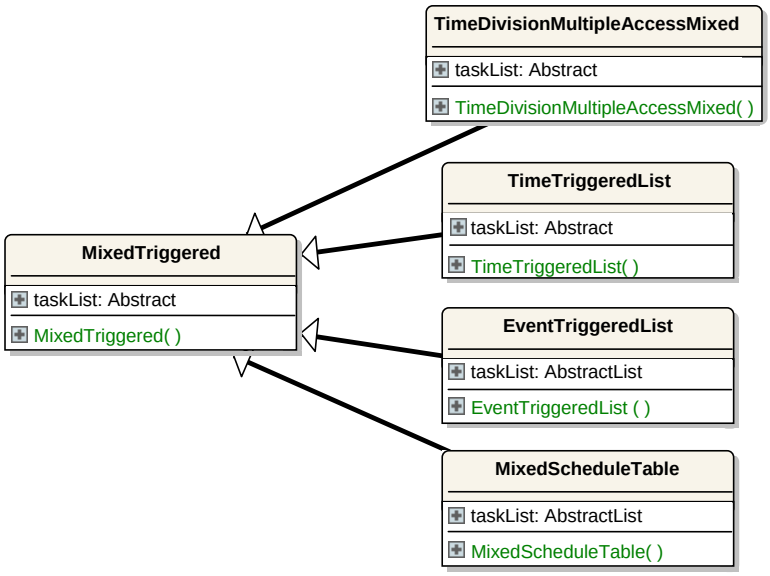

Figure 3.12 Mixed-triggered approach

The mixed-triggered bus bases it operation on a mixed schedule table which orders the ERTS tasks according to the estimation of the time when they must be executed; it is mixed because it includes static and dynamic tasks. In order to assign the necessary transmission time over the bus to each task, it defines a time division multiple access mixed protocol.

Time division multiple access mixed
It fragments the bus access time with the purpose of provide the required time to each node of an ERTS for the execution of its tasks. It is an extension of the time division multiple access protocol used for the time-triggered bus approach; with the difference that it is adapted to handle produced by event-triggered nodes. Two elements are necessary to define the time

division multiple access mixed: Network time unit and communication cycle definition. Figure 3.13 illustrates the elements for its definition.

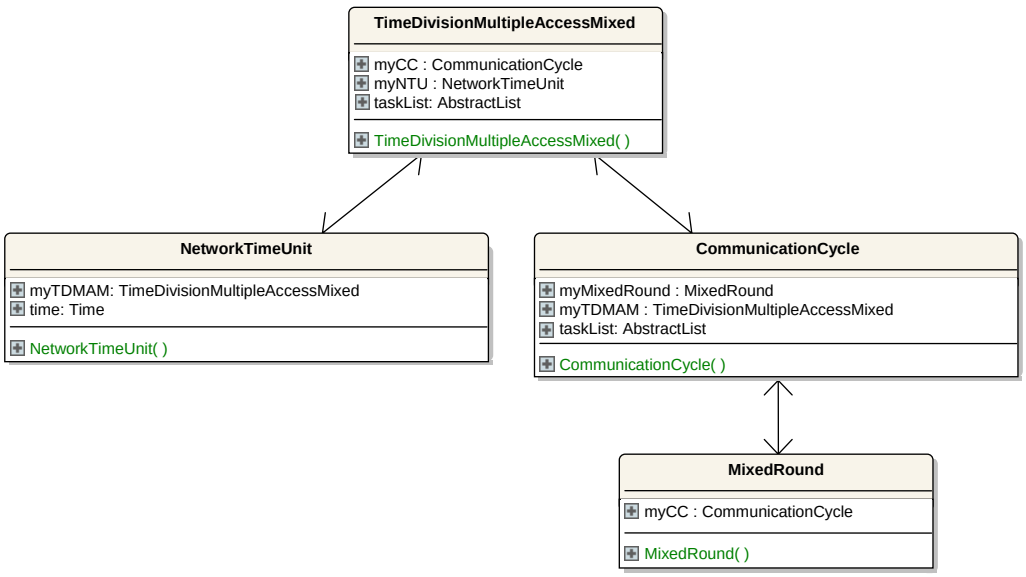

Figure 3.13 Time division multiple access mixed elements

The network time unit represents the granularity of the network access time. The network time unit is the minimal unit in which time can be fragmented, in other words, a fragment can measures from one to many network time units. Time fragments are denominated slots.

ERTS nodes that take part in the operation of a mixed-triggered bus are mixed ERTS nodes; they can act sometimes like time-triggered ERTS nodes and others like event-triggered ERTS nodes. When they act like time-triggered ERTS nodes they have a corresponding time association similar to the presented by the single time division multiple access, that means, that slots are associated to them in a static manner. But when they act like event-triggered ERTS nodes it is impossible to know in advance when they need to use the bus. In this case a set of slots, denominated slots for dynamic messages, is dispersed throughout the time to be associated to them when needed.

The set of slots that match one by one with every mixed ERTS node is denominated a mixed round. Mixed rounds are repeated continuously with the purpose of provide a periodic execution time assignation to the mixed nodes of an ERTS. Mixed round with predefined slot associations are static rounds and round with slots for dynamic messages are dynamic rounds.

Both, static and dynamic round are grouped to be repeated continuously in order to provide the required access to the bus to mixed ERTS nodes in their two facets (time-triggered and event-triggered). A communication cycle is when all mixed triggered ERTS nodes (in their two facets) are considered. The communication cycle is repeated continuously during the operation time of an ERTS.

Time-triggered and event-triggered lists

Tasks classified as time-triggered go in the time-triggered list class and the rest in the event-triggered list class. The description of every task includes the same subset of the fundamental real-time tasks properties that according to the type of bus, and the order in which tasks are located into the tasks list does not represent their execution ordered. To define an execution order that satisfies the particular real-time constraints of the entire tasks is necessary to process the task list class with schedule strategies. As result the schedule strategies produces a mixed schedule table.

Mixed schedule table

It is a class that contains a table with a fixed structure that represents a schedule for all tasks included in an ERTS. It can be generated from the application of the six scheduling policies included in the preemptive and non-preemptive scheduling types), which comprise the schedule model.

It has a priori the time-triggered tasks that can be known in advance the beginning of the ERTS execution. They cannot be altered but the rest of them can be updated dynamically. This rest of the tasks are rich with the passage of time and its occurrence, always respecting the time constraint of the task already scheduled. The mixed schedule table is considered as a dynamical table. The real-time task properties that are defined in the time-triggered and the event-triggered list classes are the necessary parameters for the application of any of the scheduling policies defined for the mixed-triggered bus.

Reference mixed schedule table
It is a class that contains a table with a fixed structure that represents a schedule for tasks that correspond to the sending of reference messages. In time division multiple access there are two types of reference messages, one that indicates the beginning of mixed rounds and another that indicates the beginning of communications cycles. A reference message is sent at the beginning of every round or every communication cycle. There is no need to use a complex scheduling polity, reference tasks are executed in predefined time intervals and its execution does not interfere with the execution of other reference tasks.

3.4. Node layer

The objective of the node layer is to specify a node model that includes the necessary classes for defining a mechanism for producing distributed and fault tolerant code, applicable to ERTS that has the capacity to detect processor failures via fail-stop behavior. It assumes that the data and error frame transmission between nodes is successfully controlled by the network layer and the concurrency control over the bus is well controlled by the communication layer. Figure 3.14 illustrates the general view of the node layer.

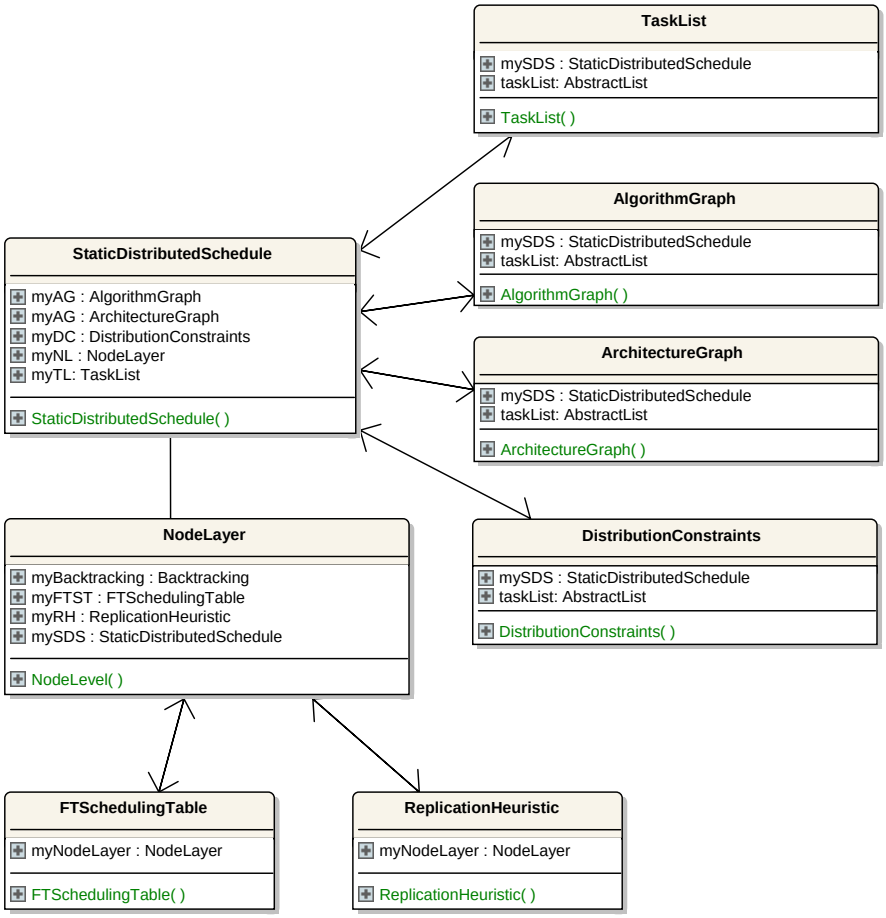

Figure 3.14 Node layer

The previous figure illustrates those classes that must be considered by an ERTS in order to generate a set of fault tolerance scheduling tables; the approach is based on the one presented in chapter 2 for the intra-nodes operation. Fault tolerance scheduling tables are calculated from the particular descriptions of the ERTS application algorithm and the description of the topology of the ERTS physical architecture, considering the particular real-time tasks fundamental properties and the ERTS distribution constraints. The classes that compose the node layer are the static distributed schedule, the fault tolerance scheduling table, the replication heuristic, and the backtracking technique.

3.4.1. Static distributed schedule

Its objective is to provide a specification for the definition of the AAA method input parameters and to generate a set of possible schedule tables. It is composed by four sub classes.

Task list model
The first class needed for the estimation of a static distributed schedule is the task list model. It is a model for the specification of the entire tasks that are included into an ERTS. It allows the definition of a task list, when the description of every task includes its fundamental real-time properties. The order in which tasks are located into the task list does not represent some class of relationship.

Algorithm and architecture graph models
The second and third classes are the algorithm graph model and the architecture graph model respectively. The second one is a model for the ERTS algorithm specification that defines the manner in which operations and data-dependencies must be represented; it establishes the

policies to represent the information as a directed non cyclic graph, where each vertex is an operation and each edge is a data-flow channel. The third one is a model for the ERTS architecture specification that defines the manner in which nodes and communication links among them must be represented; it establishes the policies to represent the information as a non oriented hyper-graph graph, where each vertex is a node and each hyper-edge is a communication link.

Distribution constraints model
The last class is a distribution constraints model. It is a model for the specification of the distribution constraints originated in the assignation a set of nodes from the architecture graph to each operation of the algorithm graph. It establishes the policies to represent the characteristics of each operation relatively to the hardware component on which it can be executed. The characteristics representation is done with a couple of tables (one for operations and another for data-dependency).

3.4.2. Replication heuristic

The replication heuristic is a claass for providing the definition of a particular fault tolerance heuristic for the AAA method that can be substituted instead the original heuristic in the static distributed schedule element. The fault tolerance replication heuristic class calculates the possibility of each node to be executed in more than one ERTS node, instead the normal one by one association.

3.4.3. Fault tolerance scheduling table

The execution of the static distributed schedule technique with the fault tolerance heuristic produces a set of fault tolerance scheduling, that are located in a scheduling table class. The fault tolerance scheduling table provides an adapted scheduling according to the functional nodes of an ERTS

3.5. Discussion

This chapter presented a fault tolerance model for ERTS. The model abstracts the analyzed concepts presented in the studied start-of-the-art, and based on them defines three models whose objectives was to be focused on the principal fault focuses, which are network, communication, and nodes.

Every presented algorithm on the start-of-the-art is represented as one model element that is classified for a particular fault tolerance layer. This classification allows to the designer to select only a useful fraction of the model adapted to the ERTS implementation necessities. This selection capacity can be done thanks to the characteristic of independence of every of the layers that compose the model. The independence capacity means that there are no common elements among the fault tolerance layers that imply that the decision of adapt one layer of the model implies that another adaptation of another element must be done.