Partitioning in the Time-Triggered Architecture

John Rushby
Computer Science Laboratory
SRI International
Menlo Park CA 94025 USA

March 2001

Deliverable 3 for SRI Project 11003; Subcontract to Honeywell Tucson under Cooperative Agreement NCC-1-377 with NASA Langley entitled *Design, Implementation, and Verification of Fault-Tolerant Modular Aerospace Controls*.

Abstract

We examine partitioning in the Time-Triggered Architecture. We undertake a failure modes and effects analysis for an ideal TTA architecture and then consider the additional issues raised in practical implementations.

Contents

Chapter 1

Partitioning

Partitioning is about fault containment: the requirement is that a fault in one component of a distributed system must not be able to propagate and interfere with the operation of non-faulty components. Partitioning depends on an appropriate notion of "component"—we cannot expect to achieve partitioning if the entire system is regarded as a single component. For partitioning, the appropriate notion of component is referred to as a *fault containment unit* (FCU) and the assumption (which must be justified) is that separate FCUs fail independently. The requirement then is to show that failure of one FCU cannot affect the operation of others. As stated, this requirement is too strong: if one FCU supplies data to another, then failure of the former necessarily affects the latter. Finding an exact characterization of partitioning that allows this type of "inevitable" propagation of the effects of failure, while excluding others (such as where failure of one FCU renders another completely inoperable), is a delicate challenge, with no completely satisfactory solution at present [Rus99a]. In this report, we examine partitioning in the Time-Triggered Architecture (TTA) with respect to specific functions and capabilities.

No system can be expected to maintain partitioning if too many faults (or too many bad faults) afflict it in too short a period, so it is necessary to specify a *fault hypothesis* that describes the kinds, numbers, and frequencies of faults that the system should be able to cope with: these elements of the fault hypothesis are referred to as the *fault model*, the *maximum faults*, and the *fault arrival rate*, respectively.

In an "ideal" implementation of TTA (which is different than any actual implementation) there would be four types of FCU.

- The *host* computers that run the applications concerned
- The TTP/C *controllers* that are the interfaces between hosts and TTA
- The broadcast *communication channels* that carry data between hosts and their associated controllers

- The *bus guardians* that are interposed between controllers and the communication channels

Actual TTA implementations compromise the failure independence of these ideal FCUs in various ways (for example, in automotive applications, the controller is on the same chip as the host and shares its clock and power supply) but it is useful to study partitioning for the ideal implementation first, and then examine the consequences of various departures from the ideal.[1]

We can also postulate an "ideal" fault model. This distinguishes two kinds of fault as follows.

- An *arbitrary active fault* is entirely unconstrained.
- An *arbitrary passive fault* is unconstrained, except that it cannot manufacture messages out of nothing (i.e., the component can only change or lose messages that it is relaying).

In the ideal TTA, hosts and controllers can exhibit active faults, while the bus guardians and communication channels can exhibit only passive faults. Actual TTA implementations may have more restrictive fault models.

One additional fault "mode" that should be considered, but that has a different character from the others, is the *spatial proximity* fault: here it is assumed that all matter in some specified volume is destroyed. Analysis of this kind of fault must consider the physical layout of the TTA implementation.

The fault arrival rate assumption for TTA is that at most one FCU may fail in any two consecutive TDMA rounds. Faults that afflict more FCUs, or that arrive more frequently than this, are considered "multiple fault scenarios"; TTA can tolerate many such scenarios, but is not guaranteed to do so. The maximum fault assumption depends on the FCUs concerned, and on the particular implementation of TTA. To tolerate n communication channel faults, there must be at least $n + 1$ communication channels; standard implementations of TTA fix $n = 1$. There must be at least one bus guardian per communication channel; in the bus-based implementation of TTA, each TTP/C controller has a bus guardian for each channel, whereas the star-coupled implementation has a single bus guardian for each channel. The algorithms executed by the TTP/C controller for clock synchronization and group membership require at least four controllers to tolerate a single fault; membership is used to exclude faulty nodes, so that an r-node system can tolerate a maximum of $r - 3$ faulty controllers. Faulty hosts should have no effect on the partitioning properties of TTA, but there may be some application-specific constraints on the number of nonfaulty hosts required to provide acceptable, or safe, service.

In the next chapter we examine partitioning in TTA by means of a failure modes and effects analysis of its FCUs.

[1]This approach and much of the analysis is derived from [KB00]; an abbreviated account has been published [KBP01].

Chapter 2

Partitioning in TTA

A TTA system is operating correctly if it provides nonfaulty nodes with the services specified at its interface. These services comprise

- Clock synchronization
- Group membership
- Data communication

Partitioning in TTA must ensure that these services are provided to nonfaulty nodes without interruption, despite the occurrence of faults—provided these are within the fault hypothesis. A natural way to examine partitioning in TTA is through a failure modes and effects analysis (FMEA): we postulate all the faults that could afflict a TTA system (within the fault hypothesis) and deduce their consequences. In the following sections we first perform an FMEA for the ideal implementation, and then consider how the actual implementations differ from this.

2.1 Ideal TTA Implementation

We first consider faults in steady-state operation, and then those that could arise in startup or restart. Within each of theses phases, we perform FMEA for each type of FCU in turn. The reader is assumed to be familiar with the operation and algorithms of TTA.

2.1.1 TTP/C Controllers

A faulty controller can fail in the temporal and/or value domains: that is, it can send data at the wrong time, and/or send the wrong data.

2.1.1.1 Gross Temporal Faults

This failure mode, which includes babbling, is where a controller attempts to send data outside its TDMA slot (e.g., because it has lost synchronization). Partitioning is ensured because the bus guardians allow data to reach the communication channels only during the correct window. The single-fault assumption ensures that a controller and a bus guardian cannot both be faulty. (In a multiple fault scenario where both a controller and a guardian are faulty, partitioning should still be ensured by the guardian of the other channel). It is desirable that a controller exhibiting this failure mode should shut down; this is likely if the cause of the fault is algorithmic (e.g., loss of synchronization) because the faulty controller will fail to receive acknowledgments for its own transmissions, or fail to receive the transmissions of other nodes, and the TTA group membership algorithm will cause it to enter the *freeze* state. If the cause is low level (e.g., a faulty UART) or electrical (e.g., a faulty transistor) it is also likely that the membership algorithm will detect communications failure and enter the *freeze* state. Only if the controller has ceased to execute the correct algorithms is it likely that it will remain active and repeatedly stress the fault isolation provided by the bus guardians.

2.1.1.2 Small Temporal Faults

This failure mode arises where a controller transmits a message very close to the beginning or end of its assigned TDMA slot. Since clocks cannot be perfectly synchronized, it is possible that some receivers will consider the message to arrive within the correct window and will accept it, while others will not, and will reject it. This potentially could cause a split in the membership and the existence of several cliques of nodes that do not recognize each other's existence. The clique-avoidance component of the TTA membership algorithm is designed to exclude such a split [BP00]. This element of the algorithm has been validated by testing and simulation, but not yet by formal means. We do now have a Murϕ model that will allow exploration by model checking, and work is under way to extend the PVS verification [Pfe00] to include such asymmetric faults.

Notice that the basic scenario (transmitting close to the edge of a TDMA slot) could be caused by a fault (e.g., loss of synchronization) and it is then perfectly appropriate to tolerate it by means of the clique-avoidance element of the membership algorithm; however, legitimate clock skew also could cause nodes to transmit close to the edges of the receive windows of other nodes: this is not a fault, and should not trigger asymmetric receptions. Rather, the parameters of the TTA algorithms should be such that clock skew, message transmission delay, and so on, are correctly accounted for in setting the size and timing of the send and receive windows on TDMA slots. The calculation of parameters for a general model of time-triggered communication in the presence of imperfectly synchronized clocks is described and formally verified as part of the model described by Rushby [Rus99b]. An informal description has recently been provided [KBP01], explaining how slot timing in controllers and guardians should be handled so that transmissions by nonfaulty nodes are never rejected by other nonfaulty nodes: controller windows should be at least 4Π time units longer than the maximum message duration, bus guardian windows should start Π later and end Π earlier than controller windows, and controllers should start transmission 2Π after the start of their windows, where Π is the maximum clock skew. It would be valuable and seems quite straightforward to modify the treatment of [Rus99b] to verify this calculation. Bus guardians that terminate a transmission should do so in a way that ensures that no nonfaulty receiver will accept it (e.g., by ensuring that the CRC will fail).

2.1.1.3 Value Faults

Controllers add a high-quality CRC to every message that they send and this is assumed to preclude modification of the contents by other (passive) components during transmission (a node may fail to receive a message, but cannot receive it incorrectly). The worst that can happen is that some nodes may fail to receive a message that others do receive. The duplicated communication channels minimize the likelihood of this happening, and the clique-avoidance element of the TTA group membership algorithm ensures that such asymmetric reception cannot split the membership.

A faulty controller can modify messages received from or transmitted to its host, and could calculate CRCs incorrectly. These faults affect only the host attached to the controller concerned, and do not threaten partitioning. However, information of importance to the operation of TTA itself is encoded into the CRCs that are appended to each message: these implicitly acknowledge receipt of messages from other nodes, and also indicate other elements of the controller's state. Faulty encoding of these elements into the CRC could potentially disrupt the membership algorithm: for example a controller that incorrectly acknowledges a previous broadcaster whose message was not received could cause that broadcaster to fail to diagnose its fault. However, this scenario presupposes a double fault and has the same manifestation as a broadcast that is received only at some nodes; similarly, the scenario that fails to acknowledge a message that was received has the same manifestation as a receive fault and is handled correctly by the group membership algorithm. Sending different, but correctly formatted messages over the two communication channels seems unlikely: we presume that the design of the controller ensures that only a single message is prepared and is sent identically to both channels. Even if it were possible, the message on each communication channel would be the same at each receiver, so any receiver that received both messages would perform the same actions (provided all receivers agree on the identity of the two channels); receivers that obtain a message only on a single channel might perform different actions, so this highly implausible scenario may be worth further study.

The messages considered so far are called "N-frames" in TTA; the other kind of message is called an "I-frame." A faulty controller that sends an I-frame when an N-frame is expected does no harm because its message will be rejected by all nonfaulty nodes; a node that sends an N-frame when an I-frame is expected similarly causes no harm. I-frames differ from N-frames in that the controllers inspect the fields of the message and may perform actions based on the contents of those fields. A faulty controller may send faulty I-frames and therefore has some potential to affect other nodes through the contents of its I-frame. It seems that this potential is limited to initiating mode changes, but further examination is needed to deduce the full potential consequences.

Reconfiguration is supported in TTA and this allows one controller to take over the role of another; a reconfiguring controller informs its bus guardians of it new role; this seems potentially to violate the assumed independence of controllers and guardians and warrants further investigation.

Value faults that can afflict other architectures, such as those that falsely indicate the originator of a message, or its type, or intended recipients, are impossible in TTA because these values are not sent in messages, but are implicit in the time at which they are sent and received: all nonfaulty controllers have synchronized clocks and identical MEDLs and therefore interpret all messages consistently.

2.1.1.4 Tiny Temporal and Value Faults

The electrical signals that appear on the communication channels must satisfy certain timing and value (voltage) specifications if they are to be recognized reliably: digital pulses must be of a certain duration, their edges must be suitably sharp, and their voltages must be of adequate stability and definition. It is possible for *slightly out of specification* (SOS) faults in oscillators or electrical driver circuits to generate poor signals that are interpreted differently by different receivers. The CRCs should ensure that, at each receiver, messages are received correctly or not at all (i.e., no incorrect message will be received)—but some receivers may accept messages that others do not, and this can lead to a split in the membership. As before, the clique-avoidance element of the group membership algorithm should avert this problem, but it is preferable to exclude it at source. (The clique-avoidance mechanism prevents complete breakdown of the system, but it may cause nonfaulty nodes to drop out and reintegrate, and it may—if there is no majority clique—cause the complete system to do a restart.) This can be done by causing the bus guardians to reshape and retime the signals transmitted to the communication channels. If the controller is generating flawed signals, the retiming and reshaping will ensure that the signals sent on each communication channel are properly timed and formatted (though the messages they encode will possibly be garbled—the CRCs will detect this); there is then no danger that nonfaulty receivers will interpret them differently.

2.1.2 Bus Guardians

Bus guardians prevent mistimed messages from reaching the communication channels; they also reshape and retime the messages that they do pass through. In the ideal implementation, bus guardians are separate FCUs, which requires that they have their own power supply and clocks, and are able to synchronize independently. The equivalent components of the

Honeywell SAFEbus™ [HD92, ARI93] (they are called BIUs) do have this independent character.

An improperly synchronized guardian will prevent those controllers attached to it from sending messages on the corresponding communication channel. By the single fault hypothesis, the other channel and associated bus guardians must be nonfaulty, and so the affected controllers will transmit and receive correctly on the other channel and will not enter the freeze state.

A bus guardian whose reshaping and retiming circuits are faulty could generate asymmetric receptions on the communication channel to which it is attached. As with improper synchronization, the other channel and its associated bus guardians must be nonfaulty in this case, and there will be no loss of partitioning.

Bus guardians do not have access to the CRC computation that is necessary to create properly formatted messages, so they cannot manufacture messages from nothing (hence the *passive* fault assumption). It is possible that a babbling guardian will, by the laws of probability, generate correctly formatted messages with some frequency, and could cause faults similar to those hypothesized in Section 2.1.1.3, where different messages are received on the different communication channels. This is another reason for examining the consequences of this fault manifestation, and also suggests that guardians should contain self-test circuitry that attempts to identify and curtail babbling.

2.1.3 Communication Channels

These are assumed to behave as passive wires; their only fault modes are complete failure (so that no nodes are able to communicate) and cuts (in which subsets of nodes are able to communicate). The communication channels are duplicated, so if one is afflicted in this way, the other must be nonfaulty and service will continue. In a multiple-fault scenario in which both channels are faulty, the clique-avoidance element of the TTA membership algorithm ensures that all but at most one (majority) subset of nodes will shut down, and that subset will be able to communicate correctly (else it, too, will shut down).

2.1.4 Partitioning at Startup

At startup, or on restart, controllers and bus guardians are unsynchronized. Controllers that detect no bus activity may attempt to start the system by sending a coldstart I-frame on one channel; controllers that receive this message then execute their startup algorithm. Failure modes have been identified in which some subset of nodes fails to receive the coldstart frame and establishes an independent clique. Proposed remedies include changes to the startup algorithm, and sending coldstart frames on both channels. Another failure mode has been identified in which a faulty node sends bad data in the coldstart frame. Proposed solutions include a more “distributed” algorithm in which little or no data is transmitted in the coldstart frame, and moving more responsibility for coldstart to the bus guardians. Further investigation of these issues is warranted.

2.2 Bus-Based Implementation

The prototype, bus-based implementation of TTA differs from the ideal in several respects. In this implementation, the communication channels are buses, and each controller has its own pair of bus guardians. However, the bus guardians are not fully independent of the controller: they have their own oscillator but depend on the controller for synchronization (and may also share its power supply), and they do not perform signal reshaping and retiming. A consequence of the first design compromise is that loss of synchronization may cause the transmit window of a faulty controller to impinge on that of another node (or possibly two other nodes). The controller should detect this error and enter the freeze state (since its messages will not be acknowledged and it will not receive messages from other nodes), but a faulty controller cannot be guaranteed to do this. On the other hand, the nonfaulty nodes into whose transmit windows it intrudes will detect errors and enter their freeze states. In the next TDMA round, the faulty controller could impinge on the transmit window of another good node, causing it to freeze, and so on. A consequence of the second design compromise is that SOS signals can propagate from a faulty controller to both buses and can lead to asymmetric receptions.

The bus-based implementation is vulnerable to spatial proximity faults. Both buses necessarily come into close proximity at each node, and general destruction in that space could sever or disrupt both buses.

2.3 Star-Coupler Implementation

In the proposed star-coupler implementation, each communication channel takes the form of separate lines (probably fiber-optic) connecting each controller to a single bus guardian. There are two (or in some proposals three) separate bus guardians and their associated collections of lines.

This architecture is less susceptible to spatial proximity faults because the bus guardians can be placed in different locations. Destruction in the space around one guardian will eliminate one communication channel; destruction in the space around one controller will eliminate that controller and its lines to both bus guardians, but the guardians themselves, and communications with other nodes, should remain unaffected.

The central bus guardians can incorporate the circuitry (essentially, a stripped-down controller) to perform independent clock synchronization, can be provided with their own power supply and high-quality oscillator, and can perform signal reshaping and retiming. The principal difference between this and the ideal implementation is that a single fault in a guardian can render an entire communication channel faulty. This does not affect the logical argument for partitioning, but does affect (reduce) the fault arrival rate that is consistent with any given reliability goal.

Chapter 3

Discussion and Future Work

We have outlined, informally, the arguments for partitioning in TTA, and examined it through failure modes and effects analysis for the ideal architecture and for both practical architectures.

In would be interesting to see if the informal examination reported here can be undertaken in a more formal manner that can be checked mechanically.

We have identified some areas where further investigation, both formal and informal, seems warranted.

- Formal verification of the controller and guardian window timing parameters
- Characterization of the behavior of TTA in the highly improbable scenarios where different messages are sent on the two communications channels and some receivers obtain only one of the messages
- Exploration of the consequences of misuse of I-frame data by faulty controllers
- Exploration of the consequences of reconfiguration requests by faulty controllers
- Exploration of the consequences of faulty behavior at startup

Some of the candidate MAC architectures use unconventional configurations of TTA components and it is desirable to repeat the analysis performed here for those architectures.

Bibliography

[ARI93] Aeronautical Radio, Inc, Annapolis, MD. *ARINC Specification 659: Backplane Data Bus*, December 1993. Prepared by the Airlines Electronic Engineering Committee.

[BP00] Günther Bauer and Michael Paulitsch. An investigation of membership and clique avoidance in TTP/C. In *19th Symposium on Reliable Distributed Systems*, Nuremberg, Germany, October 2000.

[HD92] Kenneth Hoyme and Kevin Driscoll. SAFEbus™. In *11th AIAA/IEEE Digital Avionics Systems Conference*, pages 68–73, Seattle, WA, October 1992.

[KB00] Hermann Kopetz and Günther Bauer. Tolerating arbitrary node failures in the Time-Triggered Architecture. Unpublished draft, Technische Universität Wien, Real-Time Systems Group, Vienna, Austria, January 2000.

[KBP01] Hermann Kopetz, Günther Bauer, and Stefan Poledna. Tolerating arbitrary node failures in the Time-Triggered Architecture. In *SAE 2001 World Congress*, Detroit, MI, March 2001. Society of Automotive Engineers. SAE paper number 2001-01-0677.

[Pfe00] Holger Pfeifer. Formal verification of the TTA group membership algorithm. In Tommaso Bolognesi and Diego Latella, editors, *Formal Description Techniques and Protocol Specification, Testing and Verification FORTE XIII/PSTV XX 2000*, pages 3–18, Pisa, Italy, October 2000. Kluwer Academic Publishers.

[Rus99a] John Rushby. Partitioning for safety and security: Requirements, mechanisms, and assurance. NASA Contractor Report CR-1999-209347, NASA Langley Research Center, June 1999. Available at `http://techreports.larc.nasa.gov/ltrs/PDF/1999/cr/NASA-99-cr209347.pdf`; also to be issued by the FAA.

[Rus99b] John Rushby. Systematic formal verification for fault-tolerant time-triggered algorithms. *IEEE Transactions on Software Engineering*, 25(5):651–660, September/October 1999.