Assumption Coverage under Different Failure Modes in the Time-Triggered Architecture

Günther Bauer Hermann Kopetz Peter Puschner

Vienna University of Technology, Real-Time Systems Group
Treitlstr. 3/182-1, A-1040 Vienna, Austria
phone +43 1 58801 18210, fax +43 1 58 69 149
E-mail: {bauer,hk,peter@vmars.tuwien.ac.at}

Abstract

The Time-Triggered Architecture (TTA) is a distributed computer architecture for high-dependable real-time systems. The core building block of the TTA is the communications protocol TTP/C. This protocol is designed to provide non-faulty nodes with consistent data despite faulty nodes. To achieve this the protocol assumes that a fault is either a reception fault or a consistent send fault of some node. Although the communications protocol of the TTA uses this rather optimistic failure mode assumption, the TTA can isolate and tolerate a broader class of faults. This is possible by making intensive use of the static knowledge present in a TTA distributed computer system. This off-line available knowledge allows to build interconnection networks, which transform failure modes of nodes into those the communications protocol can deal with. This paper will discuss three alternative implementations of interconnection networks for the TTA, which are designed to meet different failure mode assumptions.

Key Words. fault containment, fault tolerance, distributed real-time systems, time-triggered architecture, bus guardian, Byzantine fault.

1 Introduction

The risk of a system is defined as the costs associated with a critical failure mode multiplied by the probability of occurrence of the critical failure mode. Of these factors, the costs associated with a critical failure mode are an application-specific parameter that cannot be altered by architectural measures. Conversely, the probability of occurrence of a critical failure can be reduced by making system functions tolerant to faults. This is in general achieved by replication of components and the introduction of means to provide a particular system service as long as some minimum number of components operate correctly.

A critical issue with the design of fault-tolerant systems are failure mode assumptions: the failure mode of some component defines the ways in which the component may fail. Assumption coverage is a measure of the probability that a component failure is covered by the failure mode assumption. In principle, a failure mode assumption may either be restrictive or not: any restrictive failure mode assumes that even a faulty component behaves correctly to some extent (e.g., if a component is assumed to fail silently only, then the component should never produce output any more after a fault).

To achieve high assumption coverage the system designer may choose not to restrict the failure modes of components but allow faulty components to exhibit arbitrary behaviour. Toleration of this failure mode – called Byzantine in the literature – is expensive both in terms of hardware requirements and in terms of computation and communications requirements [10]. However, assumption coverage with respect to the anticipated failure mode will be 100% if faulty components may exhibit arbitrary behaviour. Achieving high assumption coverage with restrictive failure modes requires that the respective component implements self-checking mechanisms. Nevertheless, assumption coverage will never be 1 for restrictive failure modes. While toleration of arbitrary failure modes requires a higher number of components, high assumption coverage of restrictive failure modes requires more complex components.

In any case, the designer will have to find a trade-off between the costs of the fault-tolerant system, which increase with the number of tolerated faults and decrease with restriction of anticipated faults, and the risk of the system: if the costs associated with the toleration of less restrictive failure modes are lower than the reduction of risk is, these failure modes should be tolerated.

It is the objective of this paper to present three interconnection network architectures for the TTA designed to deal with different failure modes thus providing different levels of assumption coverage. The respective failure modes are transformed by properties of the interconnection network into failure modes TTP/C – the core communications protocol of the TTA – can tolerate. The paper starts with a short introduction to the TTA and TTP/C and a discussion of the fault hypothesis of the TTP/C protocol and TTA implementation obligations. In Section 3, three variants of interconnection network architectures, which have different fault-containment capabilities, are presented. In Section 4 the fault-handling capability of the three alternatives presented in Section 3 are analyzed. Section 5 compares these three different implementations with respect to their implementation costs. The paper ends with a conclusion in Section 6.

2 Architecture Overview

In the following we will present the hardware architecture of a TTA distributed computer system and the basic features of the TTP/C protocol. We will then discuss the fault hypothesis of the TTP/C protocol and, finally, will present the implementation obligations implied by this fault hypothesis on the TTA interconnetion network.

2.1 Hardware Architecture

The time-triggered architecture (TTA) is a distributed computer architecture for the implementation of dependable real-time systems. Characteristic of the TTA is the availability of a global time of known precision in each node of the distributed computer system. A large TTA system can be decomposed into a set of nearly autonomous clusters that are linked by replicated gateway components. Fault tolerance is achieved at the cluster level by replicating the nodes within a cluster and by replicating the communications channels between the nodes. In this paper we focus on the fault-handling mechanisms at the cluster level.

A TTA cluster consists of a set of TTA nodes connected by a replicated interconnection network (Figure 1 shows a logical multi-cluster configuration abstracting from replication). A TTA node computer comprises a host computer and a TTP/C communications controller with two bi-directional communication ports. Each of these ports is connected to an independent channel of a dual-channel interconnection network. In the TTP/C protocol every receiver autonomously decides whether some message broadcast was successful or not; a broadcast is successful if a syntactically correct message is received on at least one of the two communications channels.

Communication on the interconnection network proceeds according to an *a priori* established time-division multiple access (TDMA) scheme. This TDMA scheme divides time into slots. Every slot is statically assigned to a TTA node. During its slot the node has exclusive write permission to the interconnection network. The slots are grouped into rounds: during a (TDMA) round every TTA node is granted write permission in exactly one slot. Furthermore, nodes always send in slots having the same relative position within a round; finally, the slots assigned to a particular node always have the same length. A distributed fault-tolerant clock synchronization algorithm establishes the global time, which is necessary for the distributed execution of the TDMA scheme.

A cluster cycle comprises several TDMA rounds and multiplexes the slots assigned to a node in succeeding TDMA rounds between different messages produced by the node. This is similar to the TDMA round, which multiplexes the communications channels between several nodes.

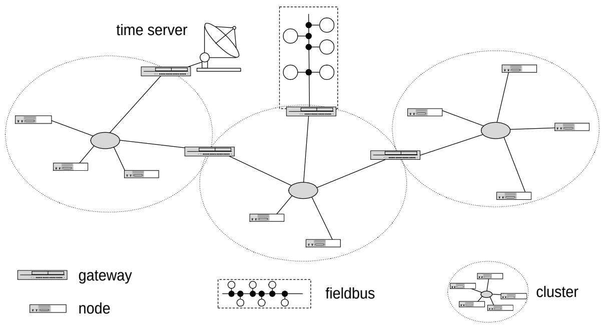

Figure 1. TTA Multi-Cluster System

Every node has knowledge of the complete TDMA scheme (and not only of the slots assigned to itself). Thus, nodes know *a priori* which messages are sent or received and there is no need for transmitting the sender ID or message IDs explicitly.

A membership service is established by the TTP/C protocol by periodic examination of message states: if a TTA node receives a syntactically correct message on either of the communications channels, it considers the respective sender correct. In the TTA, syntactical correctness is defined as follows:

Definition 1 *A message is syntactically correct, as judged by a correct receiver node, if the message passes a specified correctness criterion.*

This specified correctness criterion of the TTP/C protocol comprises four tests for each message:

1. transmission of the message starts and ends within the temporal boundaries of its TDMA slot
2. the signal constituting the message on the physical layer obeys the encoding rules
3. the received message passes a CRC check
4. sender and receiver agree on the distributed system state

It does not matter if the sender is in fact correct (as judged by an omniscient observer) or what faulty receivers conclude. If a node receives a syntactically correct message, it assumes that the contents of the message are authentic and that sender and receiver agree on the distributed system state, i.e., the controller state (C-state). The C-state consists of the membership, the global time the message broadcast was started, and the number of the current TDMA slot. To test C-state agreement, the CRC check mentioned above may be performed on the message data concatenated with the local C-state (extended CRC check [9]). If the resulting CRC checksums are identical at sender and receiver, the receiver assumes that it maintains the same C-state as the sender. Alternatively, the C-state data may be explicitly included in every message transmission (X-frame).

From the above it follows that a node can only succeed in broadcasting messages if it maintains a correct C-state. To allow for integration of nodes into an active TTA cluster, some nodes of the cluster periodically broadcast their respective C-state in I-frames (as opposed to N-frames which carry user data) if the cluster utilizes extended CRC checks. If the cluster is configured to operate with X-frames, C-state data is contained explicitly in all messages.

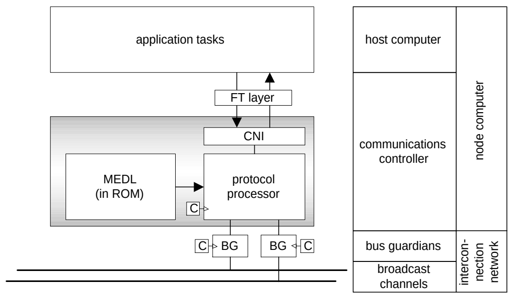

Figure 2. TTA System Node

To mask the consequences of node failures, the TTA suggests the introduction of a fault-tolerant unit layer [2]. This fault-tolerant unit (FTU) layer implements a set of well known voting strategies designed to deal with different kinds of failure modes. The FTU layer is logically placed between the TTP/C communications controller and the application tasks executed at the host computer (Figure 2). A dedicated FTU layer frees the application programmer from the burden of replication management and thus helps to make applications simpler (and less prone to design faults). Physically, the FTU layer described in [2] is located within the communications controller, so there is no need for any fault-tolerance mechanisms within the host computer subsystem of a node.

2.2 TTP/C Protocol Fault Hypothesis

TTP/C is a synchronous communications system, i.e., within the fault hypothesis of TTP/C the only reason why two nodes cannot communicate is the failure of at least one of them. The fault hypothesis of the protocol claims that only one node of a TTA cluster becomes faulty every TDMA round. Further, a new node may become faulty only after the previously faulty node (if any) either has shut down or operates correctly again. With respect to the types of faults, the TTP/C protocol assumes that a communications fault is either a local receive fault of a single node or a consistent transmission fault of a single node. I.e., if some node fails to broadcast a message, all correct receivers will consistently perceive the fault. Furthermore, the protocol assumes that a TTP/C node does not send data outside its assigned sending slot. Consequently, a correct broadcaster will always succeed in transmitting its messages to correct receivers (if at least one of the broadcast channels operates correctly).

Since the communications channels are replicated, the protocol does not need to be concerned with the fault of one of the channels: transmission will succeed on the correct channel replica in a single fault scenario as long as only one channel happens to be faulty in a TDMA slot. However, the protocol assumes that a communications channel cannot create a syntactically correct message "out of nothing" or delay message broadcasts for an arbitrary amount of time. Thus, any syntactically correct message arriving at some receiver is a genuine message produced by the node having send permission in the respective slot. Further, a communications channel may only be faulty if the broadcaster of the current slot is correct.

The TTP/C protocol assumes that the implementation of the interconnection network justifies these failure mode assumptions and provides sufficient assumption coverage. Assumption coverage is a requirement of the application and the system designer needs to choose the interconnection network appropriate to the respective application.

2.3 TTA Implementation Obligations

Given that the failure mode assumptions presented in the previous section hold, the TTP/C protocol guarantees that all correct TTA nodes are provided with consistent data and the TTP/C protocol at the nodes proceeds along corresponding (consistent) execution paths. In the following, we will discuss the mechanisms necessary at the interconnection network level to ensure that the TTP/C protocol faces only failures it can tolerate by its intrinsic fault handling mechanisms.

In the TTP/C protocol there are four parameters, which determine the execution path of the protocol at a node during synchronized operation:

1. locally stored TTP/C configuration and control data (called MEDL)
2. host computer system
3. progression of global time
4. syntactical correctness of incoming data frames

Further, a fifth parameter needs to be considered during protocol start-up or node integration:

5. the C-state communicated in I-frames or X-frames, respectively

Finally, to ensure that all correct nodes maintain consistent user data, it must be guaranteed that the message copies at the nodes will contain identical data if any two correct nodes receive a syntactically correct message in a slot.

From the above, parameter one does not require any fault isolation mechanisms at the interconnection network level: this is a fully local parameter, which is correct *per definitionem* at a correct node. An incorrect node has no means of changing the contents of the MEDL (which is contained in read-only memory) of some remote node.

As for the second parameter, we assume that the host computer system will implement a fault-tolerance layer, which enables the host computer to base its decisions on some fault-tolerant data rather than on the information received from a singleton node. Thus, fault isolation at the host computer is achieved by fault tolerance mechanisms (which are easy to implement provided fault isolation at the communications system level can be ensured).

Parameter three, progress of global time, has two aspects. First, global time is derived from the progression of local time at a node. This aspect does not need fault isolation provided by the interconnection network, as every node is equipped with its own private oscillator. Second, global time is – at distinct points in the execution path of the TTP/C protocol – influenced by a clock synchronization algorithm. In TTP/C this clock synchronization algorithm is inherently fault tolerant and provides fault isolation as long as there exists at most one faulty clock in a cluster. Further, clock synchronization is performed at least once every TDMA round. Consequently, as a consequence of the requirement that at most one TTA node becomes faulty every TDMA round (cf. Section 2.2), the algorithm that establishes a distributed global time base provides fault isolation by itself.

To ensure consistency with respect to syntactical correctness, a correct interconnection network channel needs to guarantee that the tests specified in Section 2.1 provide the same results at all correct receivers. Thus, a correct channel will allow write access of some node only during its specified time slot. Further, if the current sender creates a correct physical signal on the communications channel, the channel will deliver the same correct physical signal to all receiver nodes. If the current sender does not provide a physically correct signal, a correct channel will deliver either an incorrect signal to all nodes or no signal at all. Consistency of the results of the CRC checks trivially follows from consistency of the received physical signal. Similarly, if any two correct nodes receive a syntactically correct message, it will contain identical data.

Finally, as for parameter five, a correct interconnection network channel will forward messages containing C-state data only if the contained C-state is correct.

3 Three Alternative TTA Implementations

In Section 2.2 we have stated that the TTP/C protocol will deliver the intended service provided at most one component (either a node or one of the communications channels) fails (in the respective failure mode) at a time. Consequently, even if we succeed in designing an interconnection network, which transforms all possible component failure modes into failure modes tolerated by the TTP/C protocol, there is a non-zero probability that two components fail at the same time. In general, every fault-tolerant system relies on the existence of a minimum number of correct components. Thus, even an optimal system architecture, which has 100% assumption coverage with respect to the tolerated failure modes, can never have 100% assumption coverage with respect to the number of coincident faults.

In the following, we will present three different implementations of interconnection network architectures for the TTA. These implementations maintain different failure mode assumptions with respect to node failures and failures of the communications channels of the interconnection network. Provided the failure mode assumptions hold, the respective implementations guarantee that the TTP/C faults will face tolerable failures only. In general, we can say that the less restrictive the failure mode assumptions are, the higher the assumption coverage will be.

3.1 Ultimate Implementation

The ultimate implementation does not make any restrictive assumptions on the failure modes of communication nodes. A faulty node may broadcast syntactically correct or incorrect messages at any time. With respect to communications channels, the ultimate implementation assumes that a channel will never create syntactically correct messages "out of nothing". I.e., if some channel delivers a syntactically correct message to some node, this message has been previously broadcast by some other node. However, a channel that receives a syntactically correct message from some node may deliver this syntactically correct message to some subset of nodes, deliver a syntactically incorrect message to some other subset, and deliver no message at all to a third subset of nodes. Further, it is assumed that a communications channel that delivers a syntactically correct message to some node has received the same syntactically correct message from some other node at most δ time units ago.

In the ultimate implementation of the TTA, every system node is provided with two independent bus guardians (BG). System nodes and bus guardians are assumed to fail statistically independently, i.e., they are assigned to distinct fault-containment units (FCUs). Each one of the BGs controls node access to one of the replicated channels of the interconnection network. Every FCU has its own independent power supply and its own independent oscillator.

The BGs reshape the incoming physical signal from the node computer (Figure 2) according to the *a priori* known coding rules. Signal reshaping must be performed both in the temporal domain (edges of the code) and in the value domain (signal level), based on the independent clock and power supply of the BG. Each BG establishes its global time-base independently of its associated communications controller, i.e., it performs the same fault-tolerant distributed clock synchronization algorithm as a regular TTP/C communications controller. According to this independently established global time, the BG grants write access to the attached channel only during slots *a priori* known to be assigned to its node. Within these slots, the BG allows sending only if the communications controller starts message transmission early enough to transmit the whole message within the boundaries of the current slot. Further, the BG will perform semantic analysis of the messages received from its attached node and will not forward any message containing invalid C-state data.

Since, in the ultimate implementation, each node consists of three independent FCUs, i.e., at least three independent chips with their own power source and their own oscillator, a system with ten nodes will comprise thirty FCUs. In order to find cheaper solutions, the fault-handling capabilities of other implementations, which require a smaller number of chips, are investigated.

3.2 Prototype Implementation

The prototype implementation was designed in the projects TTA [13] and X-by-Wire [5]. It was one objective of this implementation to design and evaluate a communications architecture for high-dependability distributed real-time systems for the automotive market. With respect to the number of chips, this architecture should be comparable to the presently widely deployed CAN bus [4], but with respect to dependability, it should be significantly superior to CAN. The design was guided by the idea that if the implementation results in a structure that is significantly more expensive to implement than a system with replicated CAN buses, it is very difficult to convince the automotive industry that such an architecture is economically viable.

During an extensive set of fault injection experiments on the MARS architecture [8], a precursor of the TTA, it was observed that failure modes causing a node to send outside its assigned sending slot (either syntactically correct messages or arbitrary other data) happen in about 1% of the observed node failures. These failure modes must therefore be isolated by the prototype implementation[1]. A failure mode where some node sends a message, which is considered syntactically correct by some receivers but incorrect by others or inconsistent omission faults have not been observed in the fault-injection experiments and these failure modes were therefore considered less relevant.

According to these observations, the prototype implementation assumes that the failure modes to be isolated are benign violations of the TDMA access pattern. It is assumed, that even a faulty node will never produce a message, which is considered syntactically correct by some receivers but incorrect by others. As with the ultimate implementation, it is assumed that a broadcast channel will never create a syntactically correct message.

In order to reduce the chip count and the silicon real-estate of the TTP/C controller chip, it was agreed to provide

[1]A CAN system cannot handle these failure modes - it causes a failure of the complete communications system (commonly called *babbling-idiot* failure).

only a single BG for both channels and to implement this single BG on the same chip as the communications controller. Furthermore, it was decided that the silicon real-estate of the BG should not need more than 5% of the silicon area of the communications controller. This decision precludes the provision of a fully independent clock synchronization mechanism in the BG. However, the BG was provided with an independent time source (oscillator). Signal reshaping by the BG was excluded, since the bus guardian and the communications controller were connected to the same power supply. A failure of the power supply thus would have affected both the bus guardian and the node computer consequently rendering signal reshaping useless. In order to ensure fault isolation capabilities despite these economically driven compromises, a far-reaching set of error-detection mechanisms and plausibility checks was included in the prototype implementation. Whether these error detection mechanisms are sufficient, is currently investigated on the basis of experimental observations in extensive fault-injection experiments.

3.3 Star-Coupler Implementation

From the conceptual point of view, the prototype implementation has two shortcomings:

1. The use of a replicated bus system makes a system vulnerable to spatial proximity faults. The two channels of the bus form two independent fault-containment regions that are in close physical vicinity at least at each node, where they must interface to the communications controller. If, e.g., a physical failure causes catastrophic damage to a node and its vicinity, it is difficult to argue that at most a single channel of the bus will be affected. This shortcoming is of concern to any implementation that relies on replicated buses.

2. The single BG, which resides on the same chip as the communications controller, is not fully independent of the communications controller. Although it has its own oscillator, it relies on control signals from the communications controller and uses the same power supply as the communications controller.

To overcome these weaknesses, a TTA implementation with two independent star couplers has been designed [1, 7]. This implementation integrates the functions of the BG into the star couplers. Further, it uses the same controller chips as the prototype implementation. Again a "logical node" consists of three independent FCUs: the node computer and the two central star couplers with the integrated BGs with their own independent distributed fault-tolerant clock synchronization, independent oscillators, and independent power supplies. The main difference between the star coupler implementation and the ultimate implementation is that the central BGs of the star coupler implementation supervise all nodes. Additional failure modes that may be introduced by this sharing of the BGs must thus be carefully analyzed. This physical configuration has the advantage that nodes can be physically separated and the critical parts of the communications system are duplicated in two physically separated and encapsulated areas, the two star couplers.

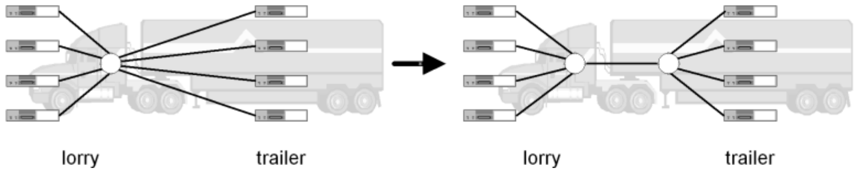

Figure 3. Distributed Star Coupler in a Brake-By-Wire Application

In order to economize on cabling costs the (logically) central BG can be implemented as a physically distributed BG. Consider, for example, a lorry and a trailer both equipped with a brake-by-wire system consisting of four nodes (Figure 3 abstracts from replication of the interconnection network). If a single star coupler with a central BG located, e.g., at the lorry were used, (replicated) stubs to all four nodes of the trailer must be provided. However, if the star coupler were physically distributed and one physical coupler were located at the lorry while the other resided at the trailer, only a single line would be needed to connect the two semi-star couplers. All traffic originating at any node connected to one coupler would be forwarded to the other.

4 Fault Analysis

In this section, the fault-handling capabilities of the three alternative implementations are investigated.

4.1 Single Faults

Ultimate Implementation. We claim that in the ultimate implementation a single FCU (node computer, one of the two BGs, or one of the two broadcast transport system channels) can fail according to the respective failure mode without an effect on the correct operation of a properly configured cluster. A failure effect would be considered catastrophic if the failure of an FCU caused a fault in any other FCU.

The reasoning is as follows: With this implementation a single FCU failure will either affect the node computer, one of the BGs, or one of the broadcast transport system channels. In case the node computer fails and tries to write to the broadcast transport system outside its assigned slots, the BGs will prevent it from doing so. In case one of the BGs fails by starting to babble, only a single channel of the broadcast transport system will be affected; the same is true if a broadcast transport system channel starts to babble itself. Thus, a correct node can always broadcast its messages on the remaining correct channel in a single fault scenario.

Of particular concern are faulty nodes which send slightly-off-specification (SOS) faulty messages, which are considered syntactically correct by some receivers but incorrect by others. There are two causes for SOS faults in the TTP/C protocol:

(a) Nodes will accept messages only if transmission both starts and ends within the assigned TDMA slot. Since it is impossible to perfectly synchronize local clocks [11], TTA nodes will have slightly different notions of the global time base. Consequently, if message transmission starts or ends slightly outside the borders of a slot, some nodes may consider it correct while others do not.

(b) The line encoding rules used by TTP/C require edges to happen at distinct distances exclusively and signal levels to be within specified boundaries; nodes will accept only messages whose encoding follows these rules. An SOS failure may cause a message to have marginally incorrect encoding and/or signal levels thus causing receiving nodes to judge message correctness differently.

The signal reshaping functionality of a correct BG as stated in Section 3.1 ensures that type (b) SOS faults of the node computer cannot propagate to receiving nodes. To prevent type (a) SOS faults from propagating, the BGs simply have to shorten the sending slot of the respective node: The TTP/C protocol provides a global time base of known precision Π. A BG that allows sending only Π time units after the nominal start of the slot for a duration that is shortened by 2Π time units with respect to the nominal duration of the slot will assure that all correct nodes receive the respective message within their local view of the slot. Consequently, a timing-SOS fault of the broadcaster will always be isolated by the BGs. If nodes start their broadcasts only 2Π time units after the start of the respective slot and slots are at least 4Π time units longer than the node will need to transmit its data, the transmissions of correct nodes will never be truncated by a correct BG [12].

An SOS fault of a single BG or a channel of the interconnectin network in general can be tolerated: Membership information in the TTP/C protocol is derived from message correctness (Section 2.1) and a node will be considered correct if a syntactically correct message is received during its TDMA slot on at least one channel. If an implementation assures that any SOS failure that happens to cross the borders of a TTA system node will always be accompanied by a correct message on one of the replicated channels, receiving nodes will maintain a consistent membership view. The ultimate implementation meets this requirement: if any one of the BGs or one of the channels is SOS faulty, the other BG or broadcast channel and the node computer will be correct. A correct node computer in combination with a correct communications channel is guaranteed to transmit a correct message on the respective channel.

	Scenarios				
Bus Guardian Channel 1	faulty	ok	ok	ok	ok
Bus Guardian Channel 2	ok	faulty	ok	ok	ok
Broadcast Channel 1	ok	ok	faulty	ok	ok
Broadcast Channel 2	ok	ok	ok	faulty	ok
Node Computer	ok	ok	ok	ok	faulty
Consequence	one of the replicated communications channels operates correctly; a syntactcally correct message will be broadcast on this channel				both channels operate correctly; a SOS failure of the node computer cannot propagate

Figure 4. Single Failure Scenarios

Figure 4 summarizes all possible single failure scenarios, i.e., cases where one of the five FCUs fails. Whenever there is a failure on one of the replicated communications channels (because of a failure either of the bus guardian or of the broadcast channel), a correct message will proceed along the other communications channel. If a node computer fails, the bus guardians will isolate the failure.

The ultimate implementation avoids spatial proximity faults by placing the two BGs and the node, which are connected by point to point transmission lines, into three different physical fault-containment regions. A single fault that annihilates all matter within the physical space of one fault-containment region thus cannot affect the proper behavior of a cluster.

Prototype Implementation. The prototype implementation assumes that the nodes are fail-silent or – if faulty – send syntactically incorrect messages consistently to all receivers. If this assumption is violated, then a system global failure may occur in the worst case. Such a worst case occurs if a node generates the same SOS fault on both channels. This kind of asymmetric fault can cause an inconsistent view of the membership of the faulty node at the correct nodes of the cluster. The ensuing inconsistency will be resolved within at most two TDMA rounds [3]. At the end of this interval, the subset of nodes, which forms the minority view will be shut down by the clique avoidance algorithm of TTP/C. If the SOS fault is transient, then the shut-down nodes will reintegrate into the cluster within the application specific recovery time. If no minority configuration survives (in case of multiple or a permanent SOS fault), the whole cluster will restart within the cluster recovery time, which is in the millisecond range. If this worst-case SOS fault is permanent, then the cluster may not be able to perform a restart. Therefore, the deployment of the prototype implementation is not recommended for fail-operational applications that must tolerate any arbitrary single fault. For fail-safe applications, the occurrence of the worst-case permanent SOS fault is of no safety concern. The fault is detected by the TTP/C clique avoidance and the application can switch to the safe state. Since the probability of the occurrence of such an SOS fault is very low, the effect of the fault on the availability of the system is negligible.

Star-Coupler Implementation. Considering a single-fault scenario, the star-coupler implementation is a realization of the ultimate implementation. However, the ultimate implementation will permanently lose the services of a channel only if a broadcast channel itself suffers a permanent failure. Opposed to this, in the star-coupler implementation also the permanent failure of a bus guardian may cause the total loss of a channel (to prevent spare exhaustion in the star-coupler implementation it is, however, possible to make use of warm stand-by spare channels).

The design of the central BG prevents a broadcast channel from creating syntactically correct messages by

1. not providing the knowledge (algorithms) of how to generate a syntactically correct CRC field of a message to the central BG

2. not giving the central BG the capability to store messages and transmit copies of the stored messages at a later time

Considering the criterion for syntactical correctness (Section 2.1) it is extremely unlikely that a signal randomly generated by one of the replicated channels of the broadcast transport system happens to form a syntactically correct message.

4.2 Multiple Faults

In general, it is not justified to assume that the environment will only produce "single faults" whose type is covered by the fault hypothesis. An architecture for high-dependability applications must also demonstrate a defined behavior if multiple near coincident failures or unanticipated failures occur. In the TTA, we consider a set of SOS faults as "near-coincident" if the faults occur within two TDMA rounds. Faults tolerated by the communications system are near-coincident if they occur within a single TDMA round.

Fault injection experiments have demonstrated that the majority of the observed failures are fail-silent failures of nodes [6]. All three implementations of the interconnection network tolerate any number of such fail-silent node failures (either transient or permanent) as long as a minimum of four nodes survive. A particular application configuration may, however, be much less resilient with respect to multiple fail-silent node failures. This depends on the functions of the nodes in the particular application.

If two non-fail-silent near-coincident failures occur, then a TTA cluster may enter an inconsistent (semantic) state. Such an inconsistent state is resolved within two TDMA rounds by the clique avoidance algorithm. If no viable configuration survives a multiple-fault scenario, a cluster restart is initiated. We assume that many of the envisaged control applications can tolerate such a cluster restart by freezing the actuators for some tens of milliseconds until the restart is successfully completed.

In the prototype implementation, a single transient worst-case SOS fault on both channels has the same effect as a set of near-coincident non-fail-silent faults: it possibly causes the restart of the complete cluster. However, designing a system that is able to isolate two failures, where one is a fail-silent failure and the other an SOS-failure is quite straight forward on basis of the ultimate implementation: it suffices to introduce a third channel. We are currently investigating the necessary changes to the TTA for cases where two FCUs failing in a respective failure mode are to be tolerated.

5 Comparative Evaluation

In this section we compare the costs of the three different implementations outlined in Section 3. The significant economic advantage of the star-coupler implementation over the ultimate implementation is a consequence of the reduction in chip count. A ten-node system with a star coupler requires only 12 chips, compared to 30 chips for the ultimate implementation. The number of independent crystal oscillators is reduced by the same amount. The cabling effort of the star-coupler implementation is somewhat higher than that of the prototype implementation, although in

	Cost/Unit	Ultimate	Prototype	Star-Coupler
Node w/o BG (one oscillator)	15	10/150		10/150
Node with BG (tow oscillators)	18		10/180	
Standalone BG	12	30/360		
Star with BG, 10 nodes	25			2/50
Wire incl. termination	1	60/60	20/20	20/20
Total cost absolute		570	200	220
Total cost relative		285	100	110

Table 1. Cost Comparison of the Three Alternative Implementations on a Ten Nodes Cluster. The cost of a single wiring connection is assumed unity.

the automotive environment, where connection count but not cable length is an issue, the additional cost of a star coupler wiring over a daisy chain bus wiring is not significant.

Table 1 compares the costs of the three alternative implementation approaches. The figures provided are estimated values, where the costs of a single wire connection are assumed unity. Whenever an implementation needs several units of the same type both the number of units required and the costs of all units are displayed (separated by a slash mark). The last row of the table displays the costs of an implementation relative to the existing (cost-optimized) prototype implementation in percent. From this it can be seen that the fault-tolerance-optimized star-coupler implementation is only marginally (10%) more expensive than the prototype implementation while the ultimate implementation is about three times as expensive as the prototype implementation.

Table 2 summarizes the main properties of these three different implementations.

6 Conclusion

In this paper we have shown how appropriately designed interconnection network architectures enable the time-triggered architecture (TTA) to tolerate different types of failure modes.

To have bus guardians protect the replicated communications channels against faulty node computers in both the time and the value domain, bus guardians must be fully independent of node computers. This requires the BGs to have independent clocking and power sources as well as not to accept any control data or control signals from the associated node computer.

However, introducing two independent bus guardians for every node computer in a TTA cluster is quite expensive and economic arguments render this solution useless for large volume applications. As a second major contribution of this paper we have shown that node computers may share their bus guardians without compromising the fault-isolation capabilities of the BGs. The introduction of such central bus guardians (one for each of the replicated communications channels) is particularly cost efficient in a star-network topology. This combination of central bus guardian and star-network topology is, from the point of view of single-fault tolerance, equivalent to an ultimate setup where every node computer is equipped with its own distinct bus guardians.

	Advantages	Disadvantages	Application Domain
Ultimate Implementation	spatial proximity faults considered, single arbitrary failure of FCU tolerated	too many chips, too much cabling effort	theoretical reference implementation
Prototype Implementation	smallest number of chips, lowest cabling costs	fail-silence difficult to validate, no spatial proximity faults considered, bus connections difficult for fiber optics	spatial proximity faults are not an issue and application is fail-silent
Star-Coupler Implementation	spatial proximity faults considered, fault isolation by “design arguments”, single arbitrary failure of FCU tolerated, point-to-point connections advantageous for fiber optics	slightly more chips and more expensive cabling than prototype implementation	spatial separation important or application fail-operational

Table 2. Advatages and Disadvatages of the Three Implementations

Acknowledgements

This work has been supported, in part, by the European IST research projects DSoS (project No IST-1999-11585) and FIT (project No IST-1999-10748).

References

[1] G. Bauer, T. Frenning, A.-K. Jonsson, H. Kopetz, and C. Temple. A centralized approach for avoiding the babbling-idiot failure in the time-triggered architecture. In *Workshops and Abstracts of The International Conference on Dependable Systems and Networks (DSN 2000)*, pages B6–B7, New York, USA, June 2000.

[2] G. Bauer and H. Kopetz. Transparent redundancy in the time-triggered architecture. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN 2000)*, pages 5–13, New York, USA, June 2000.

[3] G. Bauer and M. Paulitsch. An investigation of membership and clique avoidance in TTP/C. In *Proceedings 19th IEEE Symposium on Reliable Distributed Systems (SRDS'00)*, pages 118–124, Nürnberg, Germany, Oct. 2000.

[4] CAN. Controller area network CAN, an in-vehicle serial communication protocol. In SAE, editor, *SAE Handbook 1992*, pages SAE J1583:20.341–20.355. SAE Press, 1992.

[5] E. Dilger. Towards an architecture for safety related fault tolerant systems in vehicles. In *Proceedings of the International Conference on Safety and Reliability (ESREL '97)*, Lisbon, Portugal, 1997.

[6] R. Hexel. *Validation of Fault-Tolerance Mechanisms in a Time-Triggered Communication Protocol using Fault Injection*. PhD Thesis, Vienna University of Technology, Institut für Technische Informatik, 1999.

[7] A.-K. Jonnson and T. Frenning. Design of a centralized bus guardian for the time-triggered protocol class c. Technical Report 9/2000, Real-Time Systems Group, Vienna University of Technology, Aug. 2000.

[8] J. Karlsson. Integration and comparison of three physical fault injection techniques. In B. Randell, J.-C. Laprie, H. Kopetz, and B.Litlewood, editors, *Predictably Dependable Computing Systems*, pages 309–327. Springer, 1995.

[9] H. Kopetz. *TTP/C Protocol*. TTTech Computertechnik AG, 1999. Available at http://www.ttpforum.org.

[10] J. Lala and R. Harper. Architectural principles for safety-critical real-time applications. *Proceedings of the IEEE*, 82(1):25–40, Jan. 1994.

[11] J. Lundelius and N. Lynch. An upper and lower bound for clock synchronization. *Information and Control*, 62(2/3):190–204, Aug./Sept. 1984.

[12] J. Rushby. Formal verification of transmission window timing for the time-triggered architecture. Technical Report Deliverable 24b, SRI Project 11003, Computer Science Labaratory, SRI International, Menlo Park CA 94025, Mar. 2001.

[13] C. Scheidler, G. Heiner, R. Sasse, E. Fuchs, H. Kopetz, and C. Temple. Time-Triggered Architecture (TTA). In *Advances in Information Technologies: The Business Challenge*, Nov. 1997. IOS Press.