

Evaluation of Fault Handling of the Time-Triggered Architecture with Bus and Star Topology

Astrit Ademaj¹, Håkan Sivencrona², Günther Bauer¹, Jan Torin³

¹) Vienna University of Technology Real-Time Systems Group, Vienna, Austria
{ademaj,gue}@vmars.tuwien.ac.at

²) SP Swedish National Testing and Research Institute Department of Electronics and Software, Borås, Sweden
hakan.sivencrona@sp.se

³) Chalmers University of Technology Department of Computer Engineering, Gothenburg, Sweden
torin@ce.chalmers.se

Abstract

Arbitrary faults of a single node in a time-triggered architecture (TTA) bus topology system may cause error propagation to correct nodes and may lead to inconsistent system states. This has been observed in validation work using software implemented fault injection (SWIFI) and heavy-ion fault injection techniques in a TTA cluster. In a TTA system, the membership and the clique avoidance algorithms detect state inconsistencies and force the nodes that do not have the same state with the state of majority of nodes, to restart. Changing the interconnection structure of the cluster to a star topology allows the use of star couplers that will isolate faults of a node, thus guaranteeing consistency, even in the presence of arbitrary node failures. The same SWIFI and heavy-ion fault injection experiments that caused error propagation in bus-based TTA clusters, were performed in the star configuration. No error propagation was observed in a TTA system with the star topology during the execution of SWIFI and heavy-ion experiments

1 Introduction

Fault-tolerant systems are being used more and more in complex and critical applications applied in automotive and aircraft industry, nuclear power plants, process control, and robotics. The design of computer systems to be embedded in critical real-time applications is a complex task. Such critical systems impose high dependability requirements, because even a small (temporal) fault of the embedded computer system can lead to catastrophic failures. Therefore, it is not sufficient that such systems

meet hard real-time constraints imposed by the physical environment; they must guarantee to meet specified constraints despite the occurrence of physical faults. Another concern of a safety-critical system is the ability to assure the consistent state of the distributed computer nodes of the system. In the past few years there has been a tendency to increase the safety of vehicles by introducing intelligent control systems like brake-by-wire and steer-by-wire. In a brake-by-wire system [13], each wheel node calculates the braking pressure according to the computation of the local node. If one of the wheel brakes is not functioning, the system enters a degraded mode where three remaining wheel nodes will distribute the pressure on the local brakes, such that the vehicle can perform safe braking. To achieve this functionality, a consistent view of the global state in all nodes is required.

The Time-Triggered Architecture (TTA) [9] is an architecture for safety-critical applications like drive-by-wire and fly-by-wire. At the core of the TTA is the TTP/C communication protocol [10] running in a dedicated microcontroller. A prototype micro-programmable version of a TTP/C controller chip was designed and implemented in the course of the EU-funded ESPRIT OMI project TTA. The TTA based on a bus topology using the above mentioned controller was validated by multi-million fault injection experiments performed in the course of the EU IST Project FIT [6]. Different fault injection techniques were applied to validate the TTA, starting from runtime software implemented fault injection, pin-level fault injection, fault injection in the C-Sim simulation model, heavy-ion radiation, fault injection into the protocol code (permanent pre-runtime SWIFI), and VHDL fault injection.

The TTP/C fault hypothesis states that the TTP/C protocol is designed to tolerate a single fault in one of the constituent parts (nodes, buses) of the system without an impact on the operation of the whole system [10]. Each TTP node in a bus-based configuration has a bus guardian, which is claimed to be an autonomous (independent) subsystem of a node and allows the node to transmit only within the nodes scheduled transmission window [17]. The

bus guardian protects the bus from temporal transmission failures.

In the first part of this paper we present the results of SWIFI and heavy-ion experiments in the TTA with the bus topology and the dedicated controller TTP/C-C1 [15]. It was shown that in the current implementation of the TTP/C communication controller

- slightly-off-specification (SOS) failures are intricate to handle [1],
- it is possible to violate the fault hypothesis of the TTP/C protocol,
- the implementation of the bus guardian in the TTP/C-C1 communication controller is not fully independent (autonomous).

Two solutions to implement independent bus guardians were proposed [1], and the most cost efficient, with a centralized guardian and the star topology was implemented. With the centralized guardian the fault hypothesis of the TTP/C [10] can be extended beyond the restricted “single fault assumption” to tolerate arbitrary node failures [8]. Validation of the latter assumption using the SWIFI and heavy-ion fault injection experiments is presented in the last part of this paper.

The remainder of this paper is organized as follows: Section 2 describes the Time-Triggered Architecture in general. In Section 3 the results of SWIFI and heavy-ion fault injection experiments in a TTA bus system are presented. A definition of fault containment and error containment regions are given in Section 4. Section 5 describes the design requirements and the implementation of the central bus guardian. Validation of the TTA with the central bus guardian by means of fault injection experiments is given in Section 6. The paper ends with concluding remarks in Section 7.

2 Overview of the Time-Triggered Architecture

The Time-Triggered Architecture (TTA) consists of a system of n nodes (called TTP nodes) interconnected by two (replicated) communication channels denoted as channel 0 and channel 1 (Figure 1). The nodes access the replicated channels according to a Time Division Multiple Access (TDMA) scheme. Each node in the system is assigned a unique transmission slot, which is specified in a static data structure of the TTP/C controller called the Message Description List (MEDL) [10].

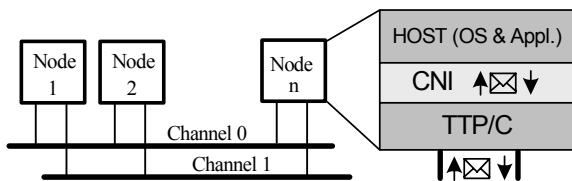


Figure 1: TTA bus topology

A TTP node consists of a TTP/C communication controller, a Communication Network Interface (CNI), and a host controller. The CNI (implemented as a Dual Ported RAM) is an interface between the application layer and protocol layer of a TTP node. A set of interconnected and synchronized TTP nodes is called a TTP cluster. The TTP/C protocol runs on the TTP/C communication controller, whereas applications run on the host subsystem. In the TTA with the bus topology (Figure 1), each node is equipped with a local bus guardian [17]. The bus guardian is an autonomous subsystem of a TTP node, which protects the communication channels from temporal transmission failures. It allows transmissions only within the slots assigned to the respective node. The bus guardian has a separate oscillator to prevent temporal coupling with the communication controller and to prevent common mode failures. The main characteristics of the TTA are common notion of time in all nodes and the provision of fully specified interfaces between these nodes. The TTP/C protocol basically provides three services to the application level:

- deterministic message sending,
- provision of a global time base and
- membership service – each correct TTP node has the actual view of which nodes are currently active (operating) in the cluster. This information is stored in the membership vector, which consists of a vector of n bits, where n is the number of nodes in the cluster.

In a TTA system, nodes transmit N-frames (containing the messages exchanged among the nodes in the cluster) and I-frames (contain the local view of global system state). I-frames are used by nodes that are reintegrating in an already synchronized cluster or during system startup. I-frames consist of: local view of global time, slot position, application mode and the membership vector. The information contained in I-frames is called the C-state (cluster state).

In the TTA star topology (Figure 2) two redundant central hubs (one for each communication channel) implement central guardians whereby no bus guardian (needs to) exist(s) at local nodes [3]. A detailed description of the star coupler implementation is given in Section 5.

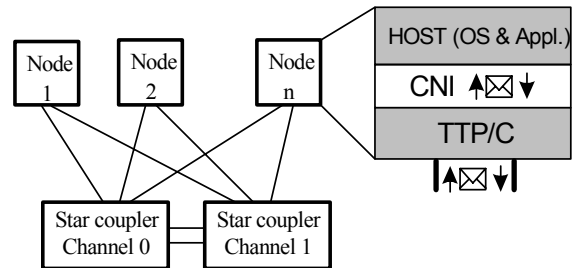


Figure 2: TTA star topology

3 TTA Bus Topology Validation

The TTP/C protocol relies on a single-fault assumption and claims that a single fault in any of its constituent parts (nodes, communication channels) should not impact the operation of the system (fault hypothesis) [10]. The fault hypothesis of the TTP/C protocol assumes that a TTP node is fail-silent. A fail-silent component either delivers correct service or does not deliver any service at all. If the delivered service is untimely (a computational result is generated too late or too early), a fail-silence violation in the time domain has occurred. If the value of the delivered service does not comply with the specification, a fail-silence violation in the value domain has occurred. It is the duty of the communication subsystem (TTP/C protocol) to assure that no fail-silence violations in the time domain occur, i.e., the messages from one node should be transmitted within the predefined time interval. Fail-silence violations in the value domain are partly handled by error detection mechanisms (EDM) of the TTP/C protocol (faults that occur during the message transmission). Any fail-silence violation in the value domain not handled by the protocol (faults that occur before message transmission or after message reception) need to be handled by the end-to-end error detection mechanism implemented in the application layer.

Consider an example when a node transmits a correct frame (protected by CRC) on the broadcast bus and during the transmission the frame is corrupted because of an EMI (electromagnetic interference) fault. In this case, receiver nodes will judge the frame as incorrect (the CRC checksum computed at the receiver does not match with the received CRC checksum because the content of the transmitted frame was changed by a fault) and will reject this frame as an incorrect frame. If nodes are considered to be fail-silent, a receiver node will report a fail-silence violation, because it has received an incorrect frame (although a correct frame was transmitted by the sender node). Therefore, in practice, we denote a TTP node operates correctly if it either delivers a correct frame, or detectable erroneous frames. Detectable erroneous frames allow the error detection mechanisms (EDM) to detect the erroneous transmission consistently at all correct receiver nodes. This will ensure that all correct receivers maintain a consistent view of the distributed system state. If the erroneous transmission is inconsistently detected by the EDMs of the (correct) receiver nodes, receivers will arrive at differing views of the distributed system state. Inconsistent states in the TTP/C are resolved by the membership mechanism and the clique avoidance algorithm [10].

A fail-silence violation in the time domain occurs when a node (say A) sends outside the specified time window and collides with the frames sent by another node (say B), or even if no collision occurred, but the transmissions of node A and B are close to each other, and the receiver might interpret them as one transmission. In both cases,

other nodes will reject frames sent from nodes A and B, and both A and B will be removed from the membership. A fail-silence violation in the time domain may thus cause error propagation from a faulty sender (node A) to a correct sender (node B). The communication subsystem (TTP/C protocol) should assure that no error propagation occurs. If this can be guaranteed, proper operation of the whole system is provided by using replica nodes (one replicated node can fail without affecting the operation of the system).

In this paper we focus on validation of the TTA against error propagation scenarios by using two fault injection techniques. Fault injection experiments are performed in only one node. If, during the execution of fault injection experiments, an error is detected in a node other than the node under test, it is a clear indication that an error has propagated from the node under test. To clarify, we validate the failure isolation capabilities of current implementation of the local bus guardian in the TTP nodes.

3.1 TTA Bus Validation with SWIFI

A TTA system with the bus interconnection structure was validated using software implemented fault injection (SWIFI) experiments. The hardware setup consists of 4 active TTP nodes interconnected by redundant busses. Fault injection experiments are performed in only one node (node 4). The single bit-flip fault model was used, as this model corresponds with transient physical faults, which are considered to be the most common physical faults in computer systems. Fault injector is implemented using the code insertion technique. As workload application we use a distributed simulation brake-by-wire application developed by Volvo Technological Development. For a detailed description of the implementation of the SWIFI fault injector the reader is referred to [2].

SWIFI experiments were performed in the communication controller (registers, CNI, instruction memory). Every bit of the CNI, TTP/C-C1 controller registers, and the instruction memory was target of systematic software-implemented fault injection. The TTP/C protocol code is stored in flash memory and upon restart it is downloaded into the RAM denoted as instruction memory, and executed from there. Emulating transient bit flip faults in the instruction memory changes the protocol code and therefore enables the fault injector to emulate transient bit flips in the TTP/C protocol code.

During the execution of SWIFI experiments no frame collisions on the bus were observed. A detailed analysis of some experiments has shown that the local bus guardian was able to prevent the faulty controller from transmitting outside its sending window. However, we have observed several cases of *slightly-off-specification (SOS) failures with respect to the start of frame transmission* [1]. SOS failures occur at the interface between the analog and the digital world. The specification for a node requires that every correct node must accept input signals if they are within a specified range of tolerance with respect to

parameters such as frequency or voltage [11]. Each individual node will have a slightly different range of tolerance (e.g., determined by the actual speed of the local hardware clock or local power supply). If an erroneous node produces an output signal (in time or value) slightly outside the specified range of tolerance, some nodes will correctly receive this signal, while others might reject it. Such a scenario will result in an inconsistent state of the distributed system. Intermediate voltages or weak edges are causes of SOS failures in the value domain [14].

SOS failures with respect to the start of frame transmission occur when a frame is received within the time receive window of some nodes and slightly outside the time receive window of other nodes. Upon occurrence of an SOS failure with respect to the start of frame transmission, some of the nodes accept the frame as a correct one (first clique – the start of the frame was received within the time receive window), while others reject it as an incorrect frame (second clique - the start of the frame was received slightly outside the time receive window). Therefore, SOS failures lead to inconsistent state of the distributed system, and consequently to the formation of cliques. The minority clique performs restart because the TTP/C clique avoidance algorithm [10] resolves clique formations. In this way the nodes in the minority clique will perform restart and thus an SOS faulty node affects the operation of correct nodes. Even though an error has propagated, the TTA system was still able to detect the (propagated) error, and the infected (yet correct) part of the system takes recovery actions by means of restart.

A high reproducibility of SWIFI experiments allowed a detailed study of the behavior of the system during the execution of the experiments that lead to SOS failures with respect to the start of frame transmission. SOS failure scenarios were caused by single bit flips (in the registers and the instruction memory of the TTP/C controller) [1]. Therefore, the probability of occurrence of SOS failures cannot be considered negligible. No SOS failures in the value domain were observed, as faults injected by software are orthogonal to this dimension.

Table 1 summarizes the results of software implemented fault injection experiments. The table provides the total number of faults (i.e., bit flips) injected into a particular unit of the TTP/C communication controller and the number of errors (i.e., node failures) that were triggered by these faults. Furthermore, the table indicates the number of inconsistent perceptions, i.e., Slightly-Off-Specification

failures with respect to the start of the frame transmission that were caused by the fault injection experiments and the relative frequency of SOS failures with respect to the total number of errors triggered.

3.2 TTA Bus Validation with Heavy-Ion Fault Injection

The same cluster setup that was exposed to SWIFI experiments was validated by heavy-ion fault injection experiments. These experiments utilized a Californium-252 source that irradiated the communication controller (TTP/C-C1) with heavy ions, which cause bit flips in the silicon. The lid of the communication controller was removed to make the silicon accessible for the ions. For a more detailed description of the setup the reader is referred to [16]. The source was placed above the silicon, inside a vacuum chamber with adjusted pressure to reach an appropriate fault injection rate. Faults were injected in one node, denoted as fault injection node.

During the execution of heavy-ion injection experiments, several cases of error propagation were observed. After a detailed analysis of the results, other additional types of error propagation other than the SOS with respect to the start of frame transmission were detected. Error propagation scenarios are classified as:

- SOS with respect to the start of frame transmission.
- Reintegration – appears when the fault injection node fails because of the injected faults, and tries to reintegrate. Since the radiation source performs continuous irradiation, faults are injected also during the reintegration phase. The faulty node has a faulty state and reintegrates by sending a frame in a wrong slot, colliding with the frame sent from other node.
- Asymmetric – a frame from faulty node is accepted as correct by some nodes, and rejected as incorrect by other nodes, and the cause of it was not an SOS failure with respect to the start of frame transmission. An analysis of some experiments has shown that one possible cause of such a scenario could be an SOS failure in the value domain (the power supply of the transmission unit of the faulty sender node can be affected such that the output signal can be “weak” – in the limits of threshold level).
- Babbling idiot – faulty node performs continuous transmission in the bus at arbitrary points in time, colliding with frames sent from other nodes. Such a failure makes impossible the communication of correct nodes within the cluster.

In heavy-ion fault injection experiments, unfortunately, there is no possibility to determine the exact origin of the fault and the number of injected faults (it is not possible to count the number of ions that penetrate in silicon and cause single upset events). Therefore, we count

Target Unit	Number of experiments	Errors triggered	SOS	SOS (%)
Instruction Memory	507744	47359	6	0.01%
Register File	14192	5081	0	0 %
HW Registers	6074	1926	8	0.4%
CNI	34112	3015	0	0 %
Total	562122	57381	14	0.02 %

Table 1: SWIFI Experiment Results with Bus Topology.

one experiment from the point in time when all nodes are running synchronized until the point in time when the node under test detects an error and performs restart (or other nodes in the cluster detect an error – error propagation). Since the node under test is exposed to continuous irradiation, the probability of having multiple faults during the duration of one experiment is relatively high. Multiple faults are outside the scope of the tolerated faults in the TTP/C fault hypothesis (with the bus topology). Table 2 summarizes the results of the heavy-ion radiation experiments.

Number of Exp.	37,036	
SOS	17	0,04%
Reintegration	49	0,13 %
Asymmetric	11	0,028%
Babbling Idiot	1	0,003%
Total	78	0,21 %

Table 2: Heavy-Ion Experiment Results with Bus Topology.

The most common failure scenarios were those that came in an already degraded system, e.g. when the node under test tried to reintegrate in the synchronized (running) cluster. The second most frequent failures were SOS failures with respect to the start of frame transmission. In all error propagation cases observed during the execution of heavy-ion experiments, there was no case where a failure resulted in a permanent disturbance of the cluster, even though there were cases when the communication controller was permanently damaged and has to be removed. Upon the occurrence of error propagation the error detection mechanisms of the TTP/C have detected the error and the system takes recovery actions by means of restart.

4 Fault Containment and Error Containment Regions

In any fault-tolerant architecture it is important to distinguish clearly between fault containment and error containment. Fault containment is concerned with limiting the immediate impact of a single fault to a defined region, while error containment tries to avoid the propagation of the consequences of a fault, the error [11]. It must be avoided that an error in one fault-containment region propagates into another fault-containment region that has not been directly affected by the original fault. A fault-containment region (FCR) is defined as the set of subsystems that share one or more common resources. A fault in any one of these shared resources may thus impact all subsystems of the FCR, i.e. the subsystems of an FCR cannot be considered to be independent of each other [7]. Subsystems that share common resources, for example computing hardware, power supply, timing source, clock

synchronization service or physical space form a fault-containment region.

The analysis of the main trends in deep submicron technology shows the multi-bit errors are more likely to occur, as both junction size and critical charge continue to decrease. Smaller features and modified wire cross-sections, as well as higher frequencies, will raise the likelihood of timing violations [5]. Because of the further miniaturization of future submicron integrated circuits, transient and intermittent faults will happen more frequently and transient and intermittent failures will in general not be contained within a region of a single die. This means that independent fault containment regions must be implemented in separate silicon dies.

In a TTA system fail silent violations in the time domain can cause error propagation, therefore, in order to avoid error propagation caused by message transmission we need timing error detection mechanisms that are in different FCRs than the message sender [11]. Otherwise, the error detection mechanism may be impacted by the same fault that caused the timing failure. In the TTA with the bus interconnection structure and the TTP/C-C1 version of the communication controller with the local bus guardian there is no clear physical separation of FCRs between the bus guardian and the TTP/C protocol unit (message sender). The local bus guardian and the TTP/C protocol unit share the same clock synchronization service, power supply and physical space (silicon die). Only a truly independent guardian allows the clear distinction between a sending FCR and the FCR performing timing failure detection. This independent guardian needs to be implemented in a separate silicon die, and needs to maintain independent clock synchronization service, and to have separate oscillators and power supply. One solution to implement a truly independent guardian is to use the star interconnection structure where two central hubs have the role of central guardians. In this manner a TTA system can handle arbitrary failures in of the components without an impact on the operation of a properly configured cluster.

5 The Central Guardian Approach

Figure 2 provides the (logical) top-level architecture of a TTA cluster utilizing a star topology network. The cluster comprises of regular nodes (TTA system nodes) and two star couplers. The regular nodes are connected to each of the replicated channels of the (star topology) interconnection network via bi-directional links. Two independent central guardians are located at the centre of each communication channel, i.e., at the star coupler. The guardian of a channel controls all the (frame) traffic at the respective channel. To achieve this, the guardian needs to be provided with the TTP/C clock synchronization service and needs to have access to C-state data.

A dedicated node consisting of a stand-alone TTP/C protocol controller provides these services (by providing the central guardian with a regular TTP/C protocol interface, i.e., the CNI). This controller is logically a

regular TTP/C controller that does not send any frames and whose existence is thus transparent to other nodes in the cluster. Physically, the controller is located at the star coupler and is part of the guardian itself.

This approach provides a low statistical dependency of node and guardian faults. Because of the strict “one-at-a-time” communication pattern of TDMA-based communication and the fact that a guardian protects receivers from faulty senders, it suffices to have, for all nodes, a single common guardian per channel. The central guardian is – at a particular point in time – logically assigned to the sender of the respective slot; it prohibits write access of a node outside its sending slots.

The central guardian is both physically and logically part of the interconnection network channel it is located at. Being part of the interconnection network implies that

- the guardian can transform failure modes only if and when they become manifest at the interconnection network interface of a node
- the guardian must not transform failure modes of the interconnection network itself (in particular, a faulty guardian must not spontaneously create valid frames or arbitrarily delay frames)

The fault hypothesis of the central guardian configuration assumes that only a single component of the cluster is faulty at a time, i.e., either one of the TTP nodes or one of the central guardians. If a TTP node is faulty, central bus guardians will protect the correct nodes. If one of the guardians is faulty, there will still be a correct guardian enabling communication among the (correct) nodes. The detailed fault hypothesis of the TTP/C protocol and the respective requirements for a central guardian can be found in [3]. A more general introduction to the concept of a central guardian in the TTA is provided in [4, 8].

5.1 Requirements to a Central Guardian

The central guardian needs to isolate faults of TTP nodes (when appearing at their interconnection network interface) and to transform these faults if they are not covered by the fault hypothesis of TTP/C. The result of the transformation will be a fault that can be handled by the fault tolerance mechanisms and error detection mechanisms of the TTP/C protocol. When transforming faults, the guardian will be concerned with two abstractions of the interconnection network interface of the TTP/C communication controller:

- Physical line interface
- Semantic interface

At the physical line interface the guardian needs to ensure that any frame transmission is perceived consistently at all correct receivers. By acting as a single centralized authority the guardian decides whether a frame is correct or not. The algorithms to be implemented within the guardian and the algorithms present within TTP/C communication controllers must ensure that any frame

transmission indicated to be correct by a correct guardian will be accepted by a correct receiver node. In particular, it must be guaranteed that any signal leaving a correct guardian is consistently interpreted by receiving nodes.

At the semantic interface the guardians take care that the transmission of a frame cannot possibly confuse receivers and cause them to precede along different execution paths with respect to the communication protocol. This problem arises when a faulty node sends an I-frame (initialization frame) containing an invalid C-state (local view of global state of the cluster) and some other node uses this C-state to integrate into a cluster (either in start-up or during node re-integration).

In particular, analysis and fault injection experiments have identified the following failure modes the central guardian needs to transform:

- SOS failures in the line encoding of frames at the physical layer (SOS in the value and the time domain)
- SOS failures with respect to the start of frame transmission
- transmission of any data outside the assigned sending slot (both in synchronized cluster operation and during startup)
- masquerading of nodes during the startup phase of the protocol

Additionally, to provide fault isolation to integrating nodes:

- transmission of invalid (i.e., non-agreed) C-state data

Transformation of SOS faults ensures that correct receiver nodes will perceive transmission faults consistently. Supervision of the third failure mode will guarantee that a faulty node sending outside its assigned sending slots cannot prevent a correct node from transmitting. Finally, masquerading becomes impossible to a node if the guardian checks the contents of frames carrying C-state data (startup frames contain C-state data that identify the sender of the frame).

5.2 Algorithms of the Central Guardian

The following section will discuss the algorithms implemented in the central guardian that transform failure modes as discussed in the previous section.

5.2.1 Active Reshaping

To avoid SOS failures caused by faulty encoded signals at the physical layer the central guardian performs active signal reshaping. The guardian simply decodes the incoming data stream (if the transmission meets the timing requirements discussed in the next section) received from the actual transmitter and encodes it again using the same line-encoding rule. The resulting signal is forwarded to receiving nodes. If the guardian finds that the received signal does not comply with the encoding rules it will abort forwarding the frame, thus, providing all receiving nodes

with a detectably faulty data stream (i.e., a frame that does not contain the expected number of bits).

5.2.2 Transmission Window Supervision

To avoid SOS failures with respect to the start of frame transmission the central guardian supervises the time when a transmission starts. Since the central guardian has access to the CNI of an attached TTP/C controller, it is provided with the synchronized global time base of precision ϵ . Further, the guardian knows the scheduled start time of transmissions. The guardian will let through a transmission only if it starts during a time window (the “start window”) of 2ϵ where the scheduled start time (according to the global time source of the guardian) is in the centre of the start window. Since correct nodes are synchronized to the central guardian with a precision of ϵ this algorithm will never block a correct node. Receiving TTP/C nodes will accept transmissions arriving in a start window of 4ϵ . Thus, a correct receiver node (which is synchronized to the guardian with precision ϵ) will accept any transmission that passes the guardian. Additionally, the algorithm allows (in conjunction with active reshaping) computation of the size of a transmission window that ensures that a transmission, once started during the start window, will be completed during the transmission window at all correct nodes. Finally, since a node will only be allowed to start transmission in slots that are assigned to this node, the algorithm also prevents faulty nodes from sending at arbitrary times outside their slots.

5.2.3 Semantic Analysis

To prevent masquerading and to protect integrating nodes from faulty nodes that transmit invalid C-state data, the central guardian performs semantic analysis of the frames being forwarded. Semantic analysis checks – on a bit-by-bit basis – the contents of I-frames by comparing the received data to local data. During startup the local data is the sender ID of the node transmitting a *coldstart* frame (the central guardian knows the intended ID of a node connected to a particular port). During synchronized protocol operation the central guardian checks the received C-state data against the (expected) C-state data contained in the CNI of the attached TTP/C controller. If the guardian finds that any bit does not match the expected value, it will immediately abort forwarding of the respective frame. Thus, receiving nodes will be provided with a corrupt frame (i.e., a frame that carries less bits than expected). In particular, if an integrating node receives the aborted frame, it will not use it for integration but wait for a correct frame.

6 TTA Star Topology Validation

A TTA cluster based on the star topology was validated with the heavy-ion and SWIFI experiments. In analogy to the experimental setup with the bus topology, the hardware setup consists of 4 regular TTP nodes and two star nodes.

The same regular nodes that were used in the bus topology experiments are used for fault injection experiments with the star topology, whereby the functionality of local bus guardian in regular nodes is disabled. The software setup and the application workload remain the same as for the fault injection setup with the TTA bus topology. Only the MEDLs (static configuration files for each node) have to be converted to account for the additional transmission delay introduced by the active reshaping mechanism of the central guardian. Faults were injected into one node. As in the bus topology the objective of these experiments is the validation of the TTA against error propagation. Moreover, the failure isolation capabilities of the central guardian will be validated.

6.1 TTA Star Validation with SWIFI

A TTA with the star topology was validated using SWIFI experiments. The same set of experiments performed in the bus topology is repeated. Additional SWIFI experiments are executed targeting hardware registers of TTP/C-C1 controller and especially timer registers because the results of experiments affecting timer registers are not always fully deterministic. The monitoring and fault injection setup remains the same as for the bus topology. In this campaign of experiments target locations were TTP/C registers and the instruction memory, as SOS failures (in the bus system) were observed during the execution of experiments in these locations.

Target unit	TTP/C-C1 HW Registers	Instruction Memory
Experiments	34000	507744
Error propagations	0	2
SOS	0	0

Table 3: SWIFI experiment results with star topology

No SOS failure was observed during the execution of SWIFI experiments with the star topology. However, in this set of experiments two similar cases of error propagation were observed during the execution of the experiments with the TTP/C-C1 instruction memory (Table 3). Since the fault injection time, type, and location are known, we were able to reproduce the experiment. A detailed analysis by reviewing TTP/C code has shown that there was an error in the specification of the TTP/C protocol acknowledgment algorithm.

6.2 A Note on the Design of the TTP/C Protocol Acknowledgment Algorithm

Acknowledgment of the receipt of a frame by the successor of the sender is performed implicitly through the membership service. During a sending slot a node sends

two frames, one on each replicated channel. Transmitted messages are protected by a CRC checksum calculated over the user data to be sent concatenated with local C-State data of the sender node. The C-State comprises the local view of global time, MEDL position (slot), application mode and the membership vector. At receiver nodes, the CRC is computed over the received message data and the local C-State of the receiver node (including membership vector). If the CRC check succeeds on either channel, the receiver considers the sender to be active. For the following, let A be the sender node and B be the successor of A.

When node A receives the frames of B it will calculate two CRC checksums for each frame: one checksum, say *i*, assuming that B has A contained in its local membership (thus indicating that B correctly received at least one of the frames sent by A); another checksum, say *ii*, assuming that B does not have A contained in its membership. If the first CRC check succeeds, then the previous transmission of A is acknowledged. If check *ii* succeeds, then B did not receive any correct frame from A either because A suffered a transmission fault or B suffered a reception fault. To clarify, A will wait for another node to either acknowledge the transmission of A (thus indicating that B suffered a reception fault) or to confirm the view of B (thus indicating that A in fact suffered a transmission fault).

However, if node B becomes faulty and sends frames with different content in redundant channels, such that the frame sent in one channel acknowledges the transmission of node A (membership of A is set – check *i*) and the frame sent in the other channel does not acknowledge node A (membership of A is not set – check *ii*), node A terminates its operation by raising an error interrupt [10]. Two such scenarios were observed during the execution of SWIFI experiments with the instruction memory (Table 3). In the first case a frame sent from node B in channel 0 acknowledged node A, and the frame sent in channel 1 did not acknowledge node A. In the second scenario there was a reverse order, a frame in channel 0 did not acknowledge node A, while the frame in channel 1 acknowledged node A. In these two cases, an error in faulty node B affects the non-faulty node A by causing the termination of the operation of the node A. Therefore the fault hypothesis of the TTP/C is violated.

This development of the protocol acknowledgment algorithm was a wrong design decision. It allows propagation of an error caused by a single bit flip (single bit flip belongs to the class of faults which should be tolerated by a TTP/C with either bus or star configuration). In general, there should be no design decisions in the TTP/C protocol, where the operation of a (correct) node depends on information received from only one node (which may be faulty or not-faulty), if the goal is to guarantee that the operation of the system cannot be

affected by a faulty node. *This design rule should be valid not only for TTP/C but for all fault tolerant systems that aim to tolerate single node (component) faults.*

A detailed analysis of these experiments shows that the scenario described above occurred in the bus topology but the monitoring system in the bus system was unable to detect it (these two particular experiments were repeated with the bus topology). Monitoring tasks reside in the host controller of each node, and they rely on error interrupts triggered by communication controller, in order to monitor state changes. In this error propagation scenario the node that has the faulty (node under test) node as a successor, terminates the operation, but it does not generate any error interrupt (as specified), because it was not correctly implemented. To clarify, a wrong design was implemented incorrectly. Therefore, this case was not detected during the execution of SWIFI experiments in the bus topology. This case of error propagation was detected by using a special monitoring node connected to the central guardian. This node records all state transitions and the frame status of all nodes in each slot.

Moreover, the SWIFI experiments presented in this paper show that SWIFI is an effective technique not only for validation but can be successfully applied also for verification purposes.

6.3 TTA Star Validation with Heavy-Ion Fault Injection

The same TTA cluster configuration using star topology that was used in SWIFI experiments was irradiated with heavy ions. Experiments were performed in one regular node. The node subjected to heavy-ion radiation performs software reset upon self-detection of an error, reintegrates, and continues operation. If the node under test starts consuming more current from the power supply unit than specified (because of a single event latch-up), the current guard in the power supply unit disconnects the node from the power supply. We should mention that these experiments could be destructive to the target system. Statistically we have observed one latch-up in 6 to 10 errors.

In this series of experiments three cases of error propagation were observed (Table 4). The analysis of the log files obtained from the monitoring system in these three error propagation cases has shown that the experiments scenarios were similar to scenarios described in Section 6.2. Since it is impossible to know the location of the fault injection we were not able to precisely detect the cause of this error propagation. However, the number of error propagations with the star topology compared to bus topology has significantly decreased because no SOS, reintegration or babbling idiot failures were observed.

Number of Exp.	9800	
SOS	0	0.00 %
Reintegration	0	0.00 %
Asymmetric	3	0.03 %
Babbling Idiot	0	0.00%

Table 4: Heavy-Ion Experiments in the Star Topology

6.4 Experiments with modified version of the TTP/C protocol

The TTP/C-C1 protocol code was modified such that in the scenario when a node is acknowledged in only one channel, it acts as if the transmission was acknowledged (self-confidence). This modification (in order to avoid the error propagation scenario described in Section 6.2) of the TTP/C protocol does not affect the properties of the TTP/C protocol. The exact set of SWIFI experiments conducted in the previous campaign (Section 6.1) was repeated. As can be seen from Table 5, no error propagation was observed during the execution of SWIFI experiments.

Target unit	TTP/C-C1 HW Registers	Instruction Memory
Experiments	34.000	507.744
Error propagations	0	0
SOS	0	0

Table 5: SWIFI experiment results with star topology.

Since we relate the error propagations observed in the heavy-ion experiments with the star topology, the experiments with heavy-ions have been repeated, and to reach a certain level of confidence the number of experiments has been increased.

As can be seen from Table 6, no error propagation was observed during the execution of heavy-ion experiments. In the bus topology, an arbitrary node failure causes error propagation, but in the star configuration, arbitrary node failures are isolated (or corrected – SOS cases) by the central bus guardian. Therefore, the extension of the fault hypothesis of the TTA to tolerate arbitrary failures, when the configuration with the star topology is used, is justified.

Number of Exp.	26600
SOS	0
Reintegration	0
Asymmetric	0
Babbling Idiot	0

Table 6: Heavy-Ion Experiments in the Star Topology

The fault injection experiments have shown that error propagations caused by message failures in time-triggered systems can be avoided by implementing error detection mechanisms in a separate FCR than the message sender. The design error in the TTP/C microcode, which was detected by SWIFI experiments, has shown that the above mentioned claim is valid only if the principle of the self-confidence is taken into consideration during the design of the system.

Fault injection experiments are performed in a regular node of the TTA system with the star topology. The experiments are not performed in the star nodes. Since the star nodes are replicated, it is assumed that the fault in one star node does not affect the operation of the second star node. Therefore, faults in one of the star nodes would not affect the operation of the system, since the replica star node should maintain the communication among the normal nodes. Of course this assumption should be validated by means of faults injection, which is a part of future work.

7 Conclusion

In the TTA system with the bus topology and the TTP/C-C1 version of the communication controller with local bus guardian, the sender and the timing failure detector subsystems are not clearly separated into two fault containment regions (FCR). The sender subsystem and the timing failure detection subsystem share the same clock synchronization algorithm, same power supply and reside on the same silicon die. Therefore, a single fault can affect a shared resource and affect the operation of both subsystems (message sender and timing error detector). Because of this implementation, several cases of error propagation were observed during the execution of software implemented and heavy-ion fault injection experiments. Even though error propagation in the TTA bus system has occurred, error detection mechanisms of TTP/C have detected the error and the system takes recovery actions by means of restart.

In the configuration with the central guardian, which performs signal reshaping in the time and the value domain, the sender subsystem and the timing failure detector are in separate FCRs. Thus, the central guardian presents a general solution, which resolves error propagation cases observed in the bus topology. A prototype FPGA implementation of a central guardian is being validated using heavy-ion and software implemented fault injection experiments. Fault injection experiment results show that the assumptions made about the fault isolation capabilities of the central guardian are justified. In a series of stressful fault injection experiments we have observed no error propagation in a TTA system with the star topology. This shows that a star topology is very feasible solution when a high dependability is required.

In addition, a design error is detected and corrected in the TTP/C protocol microcode for the TTP/C-C1 version of the communication controller. From this case, a design

rule is defined that states that the operation of a correct computer node in a distributed fault-tolerant system should not depend on information received from only one node, if the goal is to guarantee that the operation of the system should not be affected by an arbitrarily faulty node. In general, to avoid error propagation in distributed fault-tolerant system, the message sender unit and the error detection mechanism unit must be implemented in separate fault containment regions. The fault injection experiments have shown that this is true but only if the above mentioned design rule is taken into consideration during the design of distributed fault-tolerant systems.

8 Acknowledgments

This paper is partly supported by the EU IST project FIT and partly by the EU IST project Next TTA. The authors would like to thank Mattias Persson for supervising heavy-ion fault injection experiments. We furthermore wish to thank Prof. Hermann Kopetz for his valuable comments on earlier versions of this paper.

9 References

- [1] A. Ademaj, "Slightly-Off-Specification Failures in the Time-Triggered Architecture", Seventh Annual IEEE International Workshop on High Level Design Validation and Test (HLDVT'02), Cannes, France, October, 2002
- [2] A. Ademaj, "Assessment of Error Detection Mechanisms of the Time-Triggered Architecture Using Software Implemented Fault Injection", Fourth European Dependable Computing Conference (EDCC-4), Toulouse, France, October, 2002.
- [3] G. Bauer, H. Kopetz and W. Steiner, "The Central Guardian Approach to Enforce Fault Isolation in the Time-Triggered Architecture", In Proceedings of the 6th International Symposium on Autonomous Decentralized Systems, to appear. Pisa, Italy, April 2003.
- [4] G. Bauer, H. Kopetz and P. Puschner, "Assumption Coverage under Different Failure Modes in the Time-Triggered Architecture", In Proceedings 7th IEEE International Conference on Emerging Technologies and Factory Automation, pages 333-341. Antibes – Juan les Pins, France, October 2001.
- [5] C. Constatinescu, "Impact of Deep Submicron Technology on Dependability of VLSI Circuits", In Proceedings of the International Conference of Dependable Systems and Networks (DSN02), June 2002, Washington D.C, USA.
- [6] FIT official web page: <http://www.cti.ac.at/fit> , 2002
- [7] L.M. Kaufmann, S. Bhide, B.W. Johnson, "Modeling of Common-Mode Failures in Digital Embedded Systems", Proc. of the Reliability and Maintainability Symposium 2000, Los Angeles, USA.
- [8] H. Kopetz, G. Bauer and S. Poledna, "Tolerating Arbitrary Node Failures in the Time-Triggered Architecture", Document number 2001-01-0677, SAE 2001 World Congress, Detroit, MI, March 2001.
- [9] H. Kopetz, "The Time-Triggered Architecture", Proceedings of First International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 98) pages 22-29, Kyoto, Japan , April 1998.
- [10] H. Kopetz, et al. TTP/C Protocol. TTTech Computertechnik AG, 1999. <http://www.Ttpforum.org>.
- [11] H. Kopetz. "Fault Containment and Error Detection in TTP/C and FlexRay", Technical Report 39/2002 Vienna University of Technology, Real-Time System Group. www.vmars.tuwien.ac.at.
- [12] Lamport, R. Shostak, M. Pease, "The Byzantine Generals Problem", ACM Transactions on Programming Languages and Systems, vol. 4 issue 3, pp. 382-401, 1982.
- [13] T. Ringler, J. Steiner, "Increasing System Safety for by-wire Applications in Vehicles by using a Time Triggered Architecture", SAFECOMP, 17th Int. Conf. on Computer Safety, Reliability and Security 1998.
- [14] J. Rushby, "Systematic Formal Verification for Fault-Tolerant Time-Triggered Algorithms", IEEE Transactions on Software Engineering, 25(5) 651-660, September 1999.
- [15] Austria Micro Systems (AMS), TTP/C-C1 Communication Controller Data Sheet, Available at <http://www.tttech.com>.
- [16] H. Sivencrona, P. Johannessen, J. Torin, "Protocol Membership Agreement in Distributed Communication System- A Question of Brittleness", (2002-01-0108), SAE World Congress, Cobo Center, Detroit, Michigan USA, March 2003.
- [17] C. Temple, "Avoiding the Babbling-Idiot Failure in a Time-Triggered Communication System", In Proceedings of the 28th Annual International Symposium on Fault-Tolerant Computing (FTCS-28), June 1998.