A CAN-Based Architecture for Highly Reliable Communication Systems

H. Hilmer
Prof. Dr.-Ing. H.-D. Kochs
Gerhard-Mercator-Universität Duisburg, Germany
E. Dittmar
ABB Network Control and Protection, Ladenburg, Germany

In many application areas of distributed systems based on serial busses like CAN high safety and reliability are considered as major functional requirements. In addition, the communication system has to cope with periodic as well as event-driven messages, which have to be transferred under hard real-time constraints. Especially where a considerable amount of event-driven data occurs, a flexible event-oriented scheduling strategy has to be preferred. But, event-oriented data processing and communication demand enhanced fault tolerance techniques. This article introduces a distributed system concept based on a redundant CAN architecture, which is able to meet the above requirements. Beside hardware replication, extensive fault tolerance protocol enhancements are provided, comprising fault detection, notification, handling and recovery.

1 Introduction

Important technical requirements modern distributed computer control systems are subject to are:

- Deterministic real-time behavior;
- High reliability/safety.

The real-time and performance demands can be met by the use of an appropriate communication protocol, i.e. a bus access mechanism suitable for the characteristic of data occurrence. Most often, low level control systems are characterized by predominantly periodic data occurrence, since control services demand the predetermined, constant exchange of sensor and actuator data. In addition, sporadically occurring data such as process state changes, process alarms, and component faults have to be regarded, too. Due to their importance concerning system reliability and safety, the real-time constraints of event-driven information are high.

Highly reliable communication protocols and architectures often make use of periodic bus access strategies, such as time-slice [3], token, or polling methods which from the performance viewpoint are very efficient transferring periodically occurring data. But, periodic communication protocols have to reserve bandwidth of the bus medium in order to transmit spontaneous messages with minimum delay. In case the amount of sporadic data exceeds a specific limit, periodic access strategies are not appropriate since the average access delays increase due to the required amount of reserved bandwidth. As a result, real-time operation cannot be guaranteed. Especially regarding large-scale networks, bandwidth is limited because a lot of nodes have to be regarded in the pre-calculated schedule. Here, an event-triggered multi-master protocol like CAN seems to be the best choice. Event-driven protocols provide very low bus access delays on the average, unless the busload exceeds a maximum value. However, determinism of bus access is limited, because in case several nodes try to occupy the bus at the same time the access delays cannot be determined exactly. The priority controlled bus access technique of CAN allows the pre-calculation of worst-case transmission times for all messages [4]. Thus, the real-time behavior of CAN is suitable for many applications.

High reliability and safety of communication require comprehensive features concerning fault tolerance, comprising hardware redundancy and software for redundancy management. Since current protocols do not provide sufficient fault tolerance features, protocol and architectural enhancements are necessary. Designing a fault-tolerant system, a basic goal is to avoid the negative influence of many fault tolerance methods on the real-time behavior. Thus, measures have to be developed, ensuring highly reliable communication without restricting system performance.

2 Fault Tolerance Issues

Highly dependable system aspects has gone unnoticed so far in many applications. It can be realized by fault-tolerant system design including hardware redundancy and mechanisms to manage fault handling. Fault tolerance means that the functioning of the entire system is to be maintained despite faulty components, i.e. single points of failure must be avoided. Distributed systems deal with the multiplication and distribution of information to locally separated function modules. Multiplication and distribution must take place consistently, i.e. an item of source information must be present at the receivers in an identical state within a specific time. Inconsistencies of the distributed databases caused by faulty components can lead to a severe malfunction of the entire system. In conjunction with consistency, fault tolerance signifies that even when a fault occurs data consistency is to be preserved or restored before the propagation of faults affects the overall system function.

Inconsistencies can be avoided or at least detected through the use of a sufficient protocol strategy. An event-oriented systems, as opposed to periodic protocols, require an acknowledgment mechanism in order to detect a message loss. With respect to the acknowledgment process, the degree of reliability increases with the number of confirming receivers. The maximum possible reliability of the transmission of data is obtained with the atomic broadcast principle: a message is either correctly received by all of the operationally capable network nodes or it is received by none of them. The CAN transmission strategy provides a very effective atomic broadcast method. Here, in the fault-free case the bandwidth of the communication medium is not loaded by any acknowledgment traffic. Instead, in case of a fault the faulty message is destroyed by the detecting node during the transmission. As a consequence, all nodes discard the faulty message and thus, data consistency is assured. If inconsistencies cannot be avoided it is necessary to detect them with a high probability (high error coverage), and with a minimal delay (low error latency) in order to be able to start fault tolerance measures.

Fault-tolerant system design requires the identification of all potential failure modes and their impact on the system services. Furthermore, a fault model has to be established, describing the faults which are to be covered by the fault tolerance mechanisms. Regarding a communication system based on a serial bus such as CAN, faults can be classified as follows:

- *Global faults* lead to a breakdown of the system-wide communication. This causes the crash of the overall system function. Sample global faults are:
    - Short failures of the transmission line,
    - Open failures of the transmission line (interruption of a bus wire),
    - Short failures at a bus-side output of a network node,
    - *Bubbling idiot* failures, i.e. a deadlock of the bus due to a permanently transmitting node.
- *Local faults* are limited to malfunctions of node components, leading to a separation of the erroneous node, but do not influence overall system communication seriously.

Additionally, faults can be classified as temporary and permanent, respectively.

Component faults are to be tolerated through the use of redundancy in order to avoid a single point of failure leading to a breakdown of the entire system. For example, bus redundancy has to be provided for in order to maintain overall communication in spite of a faulty transmission channel. However, redundancy has to be managed in an efficient and suitable way, comprising different stages:

- Fault detection
- Fault separation (to avoid impairment of the system operation by an erroneous component)
- Fault notification (to all of the network nodes)
- Redundancy switch-over (i.e. replacement of an erroneous component by an operational replica)
- Recovery of a consistent system state

The primary goal of fault-tolerant system design is that in case of a fault redundancy handling takes place without any loss, corruption, and duplication of messages. In other words, data consistency has to be maintained even in the case of component faults without exceeding the timing constraints. The consequence is that the delays of all fault management stages have to be bounded and minimized.

Concerning node loss detection, some CAN layer 7 protocols use so-called life-guarding methods. Life guarding is used to refer to the cyclical transmission of life messages by all of the operational nodes to a master node. If a life message of a node does not occur, that indicates a component fault within that node. Depending on the cycle time of the life messages, an unacceptably long time may pass until the node failure is detected so that a loss of messages occurs leading to inconsistencies. In contrast, time-driven protocols most often use time-out mechanisms to detect a node loss leading to unallowable fault latencies, too. After the detection of errors all the other network nodes have to be informed within a minimum period of time. Otherwise, lost, corrupted, or duplicated messages cause further inconsistencies demanding comprehensive recovery operations. Thus, rapid error detection and notification mechanisms minimize the design effort for recovery measures and reduce fault tolerance latencies.

Fault tolerance comprises the separation of erroneous components in order to avoid the propagation of the fault to other system components. For example, a node which blockades the communication channel permanently by a short-circuit at a bus-side output has to be disconnected. Concerning distributed systems, the prevention of error propagation is often realized through the use of the fail-silent strategy. A fail-silent node is either operational as intended or do not produce any results at all.

Regarding event-triggered protocols, due to high error detection and notification latencies, the fail-silent strategy is not sufficient. Therefore, an active error mechanism has to be provided for detecting and indicating a component fault with a minimal delay. A fault-active node is able to detect and notify a local fault autonomously.

3 A Fault-Tolerant CAN Architecture

A standard CAN system is able to tolerate a subset of the mentioned fault modes. For example, CAN ensures the continuation of system operation in case of transient global faults, such as message violation caused by EMI by the use of message repetition, as well as of permanent local fault (e.g. node losses) by switching off the faulty node. In contrast, permanent global faults lead to a system breakdown. These faults can only be tolerated by the use of replicated components. But, the redundant realization of system components, in

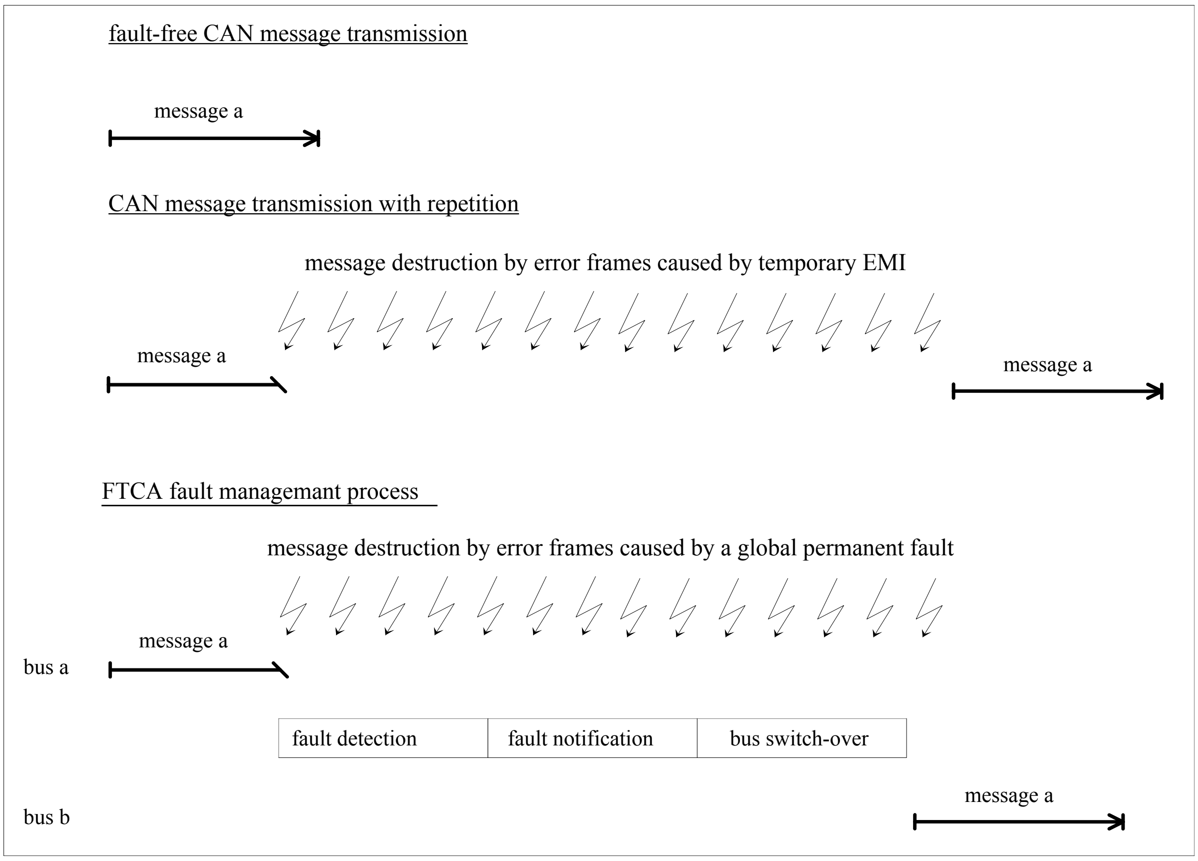
fault-free CAN message transmission
message a
CAN message transmission with repetition
message destruction by error frames caused by temporary EMI
message a
message a
FTCA fault managemant process
message destruction by error frames caused by a global permanent fault
message a
bus a
fault detection
fault notification
bus switch-over
message a
bus b

Figure 1: Atomic Actions in CAN and FTCA

particular of the bus line, requires additional mechanisms for fault detection, notification and the consistent changeover to the replicated components in order to avoid the violation of the real-time constraints and of data consistency. CAN does not provide techniques for redundancy management. Moreover, the negative confirmation strategy of CAN data transmission causes high error latencies in case of a node loss: the transmitter of a message does not detect the total outage of a network node but rather assumes that if error frames do not occur all of the receivers have received its message without faults. Thus, additional detection mechanisms have to be designed. Especially the lack of redundancy management of CAN and the lack of an efficient node loss detection gave the motivation for the fault-tolerant CAN architecture (FTCA). Our system concept provides the following features:

- Redundancy of bus line and node links
- Fault tolerance management by an active fault handling strategy (fault-active nodes), providing low fault latencies
- Enhanced fault detection capabilities (high fault detection coverage)

The major objectives and prerequisites of FTCA are:

- to keep the consistency of data in the case of a single fault (including common-mode failures) without violating the timing constraints
    - minimize fault management latencies
    - maximize fault coverage
- to satisfy an extensive fault model in order to tolerate all local and global faults;
- to minimize hardware costs (a lot of techniques, such as majority voting based on n-of-m structures are unsuitable from the economic point of view);

- to use off-the-shelf components;
- to minimize software complexity in order to prevent the software enhancements from inducing new fault sources;

In order to meet the consistency and real-time constraints in the case of a component fault, the fault management process has to be integrated into the atomic action strategy of CAN data transmission. That means, the fault management stages in the case of a global fault resulting in a bus switch-over have to be finished until a message loss violates data consistency. Only when all messages transmitted during the fault management process are rejected by the operational nodes, the fault tolerance process can be regarded as an atomic action like a standard CAN transmission (Fig. 1).

3.1 Spacial Redundancy

As to be seen in the Fig. 2, a distributed communication system using the proposed fault tolerance concept comprises fault-tolerant communication nodes. A node provides two fully redundant bus links, each link comprising a micro-controller (MC), a CAN communication controller (CAN), and a transceiver (TC).

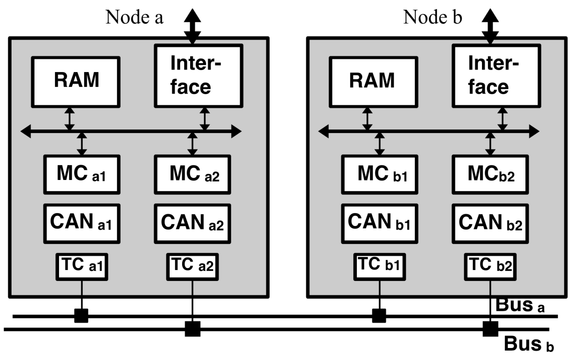

Figure 2: FTCA Communication Node

3.2 Fail-Active Fault Handling

The fault-active mechanism allows each communication link to be monitored by the other link of its node. For this purpose each micro-controller serves as a watchdog processor for the other link. In addition, in case of a component fault a node is able to become active autonomously, i.e. it informs all network nodes about the fault by transmitting a notification message through the operational link and communication channel. Thus, the second bus system fulfills the function of a watchdog bus. During normal operation the entire process data traffic is executed through one communication channel.

A sample fault reaction process due to the occurrence of a fault of CAN controller a1, takes place as follows: CAN controller a1 disrupts all network traffic on bus 1 by sending error frames until its error counter reaches 128; reaching the error counter value 96 CAN controller a1 transmits an error interrupt to micro-controller a1; micro-controller a1 informs micro-controller a2 of the loss of the CAN controller; micro-controller a2 starts the transmission of an error notification message through bus 2; all of the network nodes receive the error message through the links 2; all nodes switch off bus 1 and continue the transmission of process data through bus 2. The advantage of this method lies in the fact that the fault tolerance process is executed while the faulty CAN controller is in the active error state (error counter 127). This controller therefore continuously destroys the message which is detected as being faulty up to the switch-over process. As a result, no message is lost and no faulty message is processed until the bus switch-over has finished. Regarding redundant systems, message losses may occur because the faulty node is not able to receive any message until it is replaced by its replica component. Lost messages have to be retransmitted through the use of time-consuming recovery processes. Regarding the proposed concept, in case of a CAN controller fault no message loss occurs. But, specific malfunctions of a micro-controller may cause the loss of messages. This situation can occur, for example, if the fault detection latency is greater than the duration of the transmission of a message. Nevertheless,

	Fault Detection	Fault Separation	Fault Notification	Fault Handling	Fault Recovery
CAN (Layer 1+2)	CRC, form, stuff, bit, acknowledge error	Error-passive mode, Bus-off mode	local fault detection		message repetition, error counter decrement
FTCA	Watchdog processor, enhanced reception and transmission monitoring	Bus and link switch-off	Watchdog bus notification	bus and link replica switch-over	message repetition

Table 1: CAN and FTCA Fault Tolerance Techniques

only a few messages are to be retransmitted causing a minimal recovery effort. Thus, the preconditions of maintaining the consistency of data in the case of a fault are fulfilled. Table 1 gives a summary of the fault tolerance techniques of a FTCA system compared to the basic CAN protocol [1] features.

3.3 Enhanced Error Detection

Concerning the micro-controller and the CAN chip, there are a few malfunctions, which can be detected neither by the CAN error detection mechanism nor by watchdog mechanism, especially in an adequate amount of time. Enhancing the monitoring ability of a watchdog-CPU allows both the detection of omission faults of a transmitter and the detection of reception faults immediately. In the following, two methods are introduced improving fault detection capabilities, such that error latencies are reduced and fault coverage is increased.

Transmission Monitoring

The method to be introduced provides the monitoring of each sending process of a bus link by the watchdog-processor of its neighbor link. This takes place as follows: When a node has to transmit a process message, both micro-controllers of this node get informed about this task. While the micro-controller of the process link performs the transmission, the watchdog micro-controller is waiting for a signal of the process micro-controller indicating the successful transmission completion. As a result, in the fault-free case, a watchdog-processor gets a confirmation of each successful sending process. In case the confirmation does not arrive, a transmission loss, caused by a faulty sending component, is assumed. The watchdog processor may start a fault recovery process by transmitting the lost message through bus a. Thus, fault latency is minimized, such that a message loss is detected and re-transmitted at once.

Reception Monitoring

The total outage of a link component may cause the omission of messages. Message omissions have to be detected in time in order to start the fault tolerance mechanisms immediately. The following method provides a mechanism in which faults can be detected at the point of their occurrence:

- A master node transmits a low priority test message permanently;
- Receiving a test message, the CAN-Controller of each network node sends a reception interrupt to its micro-controller;
- The process micro-controller informs the watchdog-processor of the node about the reception of the test message.

Thus, a watchdog-processor gets reception interrupts periodically; caused either by test or process messages. In case a reception confirmation does not occur, a faulty receiver component is assumed. Now, the watchdog-processor starts the fault tolerance process by transmitting a fault notification message through the watchdog bus, as mentioned above. Using this method, the bus is occupied permanently by process and test messages, respectively. Due to higher priorities, the process messages win the arbitration against the test messages. But, the medium access delays for process data increases slightly, since a node trying to transmit a process message has to wait until a test message just being transmitted has finished. In addition, the load of the link components increases due to the permanent reception processing. In order to reduce this load, the frequency of test message transmission may be reduced.

4 References

[1] *CAN Specification 2.0*. Philips Semiconductor. 1994.

[2] Hilmer H., Kochs H.-D., Dittmar E.: *A Fault-Tolerant Architecture for Large-Scale Distributed Control Systems*. Proceedings of 14th IFAC Workshop on Distributed Computer Control Systems (DCCS'97), Seoul, Korea. 1997.

[3] Kopetz H.: *Distributed Fault-Tolerant Real-Time Systems - The MARS Approach*. IEEE Micro, pp. 25-41, Feb. 1989.

[4] Tindell K., Burns A.: *Guaranteed Message Latencies for Distributed Safety Critical Hard Real-Time Networks*. Ycs 229. Dept. Computer Science, University of York. 1993.