# Cost-effective Dependable Communication Services for Embedded Control Systems

**Authors**: *Vilgot Claesson*, Henrik Lönn*, Rolf Snedsböl*, Jan Torin* and Martin Törngren***
* Department of Computer Engineering, Chalmers
** Mechatronics Lab, Dep. of Machine Design, Royal Institute of Technology

## 1 Introduction

Within the DICOSMOS project, solutions for dependable communication which meet the requirement of low cost are being investigated and evaluated. The targeted applications are systems where safety critical motion control functions are essential; dependability is here used primarily to refer to safety and reliability requirements.

In general, fault tolerance can be implemented by use of redundancy and replication. Massive redundancy is traditionally used in high-integrity systems such as aircraft and space-shuttles. Due to cost-constraints such techniques are not possible to use in mass produced systems, thus motivating the focus on obtaining mechanisms for "low-cost" fault tolerance.

Traditionally the design of fault-tolerant system is modularized, for example in the application, processor node, communication system etc., and fault-tolerance is often handled in each module separately. This may result in non-optimal and non cost-effective realization of fault-tolerance. In the DICOSMOS project a multidisciplinary approach is taken towards implementing fault-tolerance mechanisms. The idea is that by combining control theory and computer engineering, there should be a better chance of developing low-cost architectures and low-cost solutions for safety-critical embedded control applications.

The approach taken in this report is to introduce requirements on the communication system (section 1.1); to describe the services (and protocols) proposed by the research community (section 2), and to describe representative protocols currently used in industry (section 3). Finally, in section 4, avenues for further research to reach the aim of ‘Cost-effective Dependable Communication Services‘ are discussed.

## 1.1 Requirements on the communication system

There are several requirements that have to be met by a communication system to be applied in safety critical control applications. To provide dependable communication services, aspects on timing, synchronization, consistency, error detection and redundancy need to be considered. The support for verification through debugging services is another essential requirement. There are several trade-offs associated with the implementation of these services to be discussed in this paper.

- **Timing requirements including delays, synchronization and triggering**:
  In a real-time system there are naturally requirements on the end-to-end communication delays which need to be sufficiently low and, depending on the global scheduling approach, may not vary too much, [NWT+98]. Such delay and tolerance requirements can be subdivided into one category dealing with performance and another category concerning robustness. The latter type is often referred to as ‘outage time’ which can be defined as the time ‘a component’ is allowed to be out of service before it dramatically affects the system (before the system leaves the allowed state-space). The outage time varies from application to application, for example the allowed outage time for a communication channel between two nodes may be increased dramatically if the receiving node can estimate lost data. A highly related requirement and service is that of providing a ‘global clock’ as a fundamental part of the communication system, thereby largely facilitating most design and management aspects of distributed real-time systems, see e.g. [KO87], [Kop93]. From the communication system point of view, a control system application may be viewed as composed of a set of ‘message generators’ of periodic or aperiodic type with different connection patterns and characteristics (e.g. one to one or one to many).

Different timing, triggering and consistency requirements may typically be associated with the resulting message streams. Typically, such requirements are negotiable through a control engineering perspective, see, [NWT+98] and [TES97].

- **Consistency**:
  In a distributed system the establishment of a common view on essential state-variables is a fundamental problem. For example, errors in a communication circuit or link may lead to an inconsistent view on some essential variables. It is recognized that the consistency requirements differ depending on the nature of particular information flows; compare for example discrete mode changes with a flow of sampled data representing a continuous time variable. Consistency problems are especially apparent in mode changes, for example in a start up phase. Consistency is ensured through acknowledgements in various forms; many protocol proposals exist. One major question is whether the detection and handling of inconsistent states should be made at low levels in the communication system or be possible to tailor to application-specific requirements.
- **Error detection and handling**
  From the point of view of safety and reliability a common requirement in machinery is that the system should be able to handle one arbitrary error. Several motion control applications are thus, for safety and reliability reasons, required to be tolerant to one error. Different strategies can be adopted after the first fault, depending on the system architecture and whether safety or reliability are most important in the system. Either the system goes to a safe state immediately after the first fault, or the system continues to operate, possibly with reduced functionality. For example in a car running at 100 km/h, the control system must at least continue to give service long enough to stop the car when an error occurs. A key for suitable error handling is of course error detection with high coverage. The potential sources for errors must be identified and a strategy needs to be defined for system-wide coordinated error detection and recovery. To provide fault-tolerance, redundancy needs to be introduced. Facilities for reintegration are required in certain approaches for fault-tolerance.

# 2 State of the art

In the research on communication protocols a number of services, such as redundancy management and membership agreement, have been introduced over the years in order to meet the requirements introduced in section ”1.1’, see e.g. [BSJ94] for a survey. Several projects have targeted automotive applications, for example the PROMETEHUS program (in the late 80s), PRO-VIA and the X-By-Wire project [DJK+97]. At Chalmers a testbed for the DACAPO [BJO+93] protocol has been developed and used during this work.

Essential characteristics of the proposed communication services and protocols are now introduced.

## 2.1 Time-Triggered vs. Event Triggered Communication

In an Event-Triggered (ET) system, external events are used to control operation, for example by triggering an interrupt. In Time-Triggered (TT) systems only time is used for operation control.

ET systems use run-time scheduling algorithms like earliest deadline first, least slack-time first, or fixed priority scheduling. TT systems use static schedules which can be designed before run-time. Run-time scheduling makes the event-triggered systems more flexible, but the determinism obtained from a static schedule in a TT system can be advantageous in many situations.

Below we compare different aspects of ET and TT -systems, see also [Kop93]:

- **Predictability** - TT systems are inherently predictable since the scheduling is done before run-time.
- **Testability** - The predictable behaviour of the TT system makes systematic testing easier.
- **Complexity** - In run-time, TT system will be less complex than a ET system.
- **Replication** - If replication with identical replicas is employed, identical behaviour in both the value and time domain is necessary. This is easier accomplished with time-triggered behaviour.
- **Fault Tolerance** - Event-triggered systems allow re-execution if a task is affected by a transient failure. Because time-triggered systems are pre-scheduled, tasks must always be executed twice to tolerate a failure.
- **Composability** - Several time-triggered tasks can be added to a real-time system without affecting the behaviour of other task. This is because the time encapsulation prevents unwanted interactions between tasks.
- **Timing** - Since worst case execution times have to be accommodated in the end-to-end timing budget, average delay will be better in an event-triggered system, while jitter will be worse.

- **Overload** -It is more natural to handle rare events in an event triggered system, however it is hard to ensure service for rare-events in situations where bursts of interrupts happen within a short time i.e. overload situations.
- **Communication** - As apposed to TT systems - ET systems have to do clock synchronization with explicit time messages, which will be more expensive both in software and hardware.
  In TT-systems the bus arbitration can be avoided as opposed to protocols relying on collision avoidance through bitwise arbitration such as in the Controller Area Network. Therefore, bus traffic can be filtered to a lower bandwidth and the EMI (Electromagnetic interference) is reduced.

## 2.2 Atomic Broadcast

Without any strict definition one can say that an Atomic Broadcast protocol ensures that all messages are delivered in the same order to all correct processors, i.e. all correct processors receives the same set of messages in the same order. The Atomic Broadcast is an important paradigm for maintaining a consistent state in replicated nodes of a computer system. A few examples of how Atomic Broadcast can be achieved are as follows.

Cristian proposes in [Cri90] a protocol which supports synchronous atomic broadcast by using $f+1$ redundant broadcast channels to tolerate $f$ concurrent processor crash[1] failures. This ensures that at least one message will be received by each correct processor. Since he uses a broadcast channel all messages will be received in the same order by these processors.

In the TDMA communication system TTP [KG94], Atomic Broadcast is accomplished in one communication round. This is achieved by using a combination of double broadcast channels and sending the message twice on each channel, i.e., no retransmission. If a single node does not receive a correctly transmitted message it will stop participating in communication, and by doing so it ensures the Atomic Broadcast property.

In CAN [CAN90] some services are provided to support Atomic Broadcast. When a node receives a corrupt message it will destroy the message tail which will prevent any node from receiving this message. The sender will also notice this and resend the message. Additional examples of Atomic Broadcast solutions can be found in [ASJS96] and [LS96].

## 2.3 Membership Agreement

Membership Agreement is used to achieve a consistent view of the current membership of a group of processors. This is important for example when a part of a distributed application loses its membership due to a failure. Via the membership agreement correct processors may then adapt themselves to the situation. Much research has been done on Membership Agreement, for example [ASJS96, Cri91, RB93, KGR91] and DACAPO which recommends a two-level membership; one for the correct nodes and one for a user defined application membership [LS96].

## 2.4 Redundancy management

### 2.4.1 Replica Determinism

One major problem with replicated processes is to ensure Replica Determinism i.e., to ensure that all replicas deliver the same output. To accomplish this all the replicated processes have to execute on the same input values and they also must execute in the same time period to deliver the same outputs. However, non-deterministic behaviour is introduced from a number of sources such as diverging sensor values, differences in execution timing and time-outs etc. Due to these sources diverging outputs may be produced even if the replicated nodes are correct. It is therefore necessary to ensure replica determinism. The problem of replica determinism is treated by S. Poledna in [Pol96].

### 2.4.2 Recovery

A process that has been off line due to a transient fault has to reintegrate to the system. If the process is replicated the off line process has to update its internal state since it is no longer valid. This can be done using checkpoints with roll-back. However this is seldom feasible in real-time control since data and messages are valid only for a short period of time. Instead forward recovery is advisable. To make this possible, the off line replica has to update its internal state with data from a correct replica so that all replicas have consistent states. To minimize communication this has to be done at a point in time where the saved internal state is as small as possible. The point in time

1. **Crash failure:** A process commits a *crash* failure if it deviates from correct behaviour and after doing so it stops executing further steps until restarted.

where these reintegration can take place (and how to minimize the internal state) has to be carefully planned during design.

## 2.5 Synchronization

Nodes in a distributed computer architecture that co-operate in a control task have to be tightly synchronized. To achieve this, each node can use a local clock which then has to be synchronized since the hardware oscillators of each clock run at slightly different rates. Even if the hardware oscillators of each clock run at approximately the same rate, the clocks must be synchronized periodically.

To avoid any masters in the system, all nodes are equals, and a distributed consensus on the clock value is reached by some clock synchronization algorithm. Ramanathan et al. surveys in [RSB90] several fault tolerant clock synchronization algorithms where a clock adjustment is computed based on the deviation of the local clock compared to each of the other clocks in the system. It is not necessary to send clock values explicitly to find the clock deviations. In [BD87] it is described how pre-scheduled activities, such as message transmissions, could be used as implicit clock readings. The method is applied in e.g. TTP [KO87] and DACAPO [LS95].

The synchronization is important for task execution. TDMA communication together with Synchronized tasks can minimize the control delay and jitter, since the process execution can be synchronized with the communication. However, process synchronization can be advantageous also for ET-systems, for example to minimize control jitter or bus collisions.

## 2.6 Fail Silence

In many traditional fault tolerant systems, systematic replication and backup are used. To differentiate between correct and faulty components they are often triplicated, and the correct result is obtained by voting. A triplicated system has high cost in hardware and communication bandwidth. One way of reducing these costs is to use fail silent components, i.e. a process detecting an error will prevent any faulty behaviour to spread in the system by for example shutting down (it becomes silent). As an example we have TTP which uses only two buses since the computer nodes are fail silent. To ensure that the components are fail silent it is important to detect errors with a very high probability to avoid dissemination of faulty values.

# 3 State of practice

Most applications in this area are extremely cost sensitive and solutions for fault tolerance such as ensuring fail silence are not implemented if not absolutely necessary. Therefore developers today rely to a large extent on the error detection mechanisms supported by the protocol and implemented in the communication chips e.g. Frame check errors, Parity errors, Fixed Form Field / Framing errors, etc.

A number of different communication systems ranging from proprietary to standardized ones have been developed over the years and are employed in industry. For example, MIL 1553B in military avionics, the IEC train communication network, the controller area network - CAN, and of course many proprietary ones.

## 3.1 CAN

The Controller Area Network, CAN is a CSMA/CA (Collision Sense Multiple Access / Collision Avoidance) protocol [CAN90] developed by Bosch. It was originally intended for automotive applications but is used in many types of embedded systems.

One of the strong points of the CAN protocol is its excellent error detection mechanisms. Several other aspects concerning the use of CAN in distributed real-time systems are discussed by [Tör95]. One drawback of the protocol is that the error handling policy is fixed. A CAN controller has an internal error counter that counts the number of corrupted frames on the bus. When the counters reach a certain limit the communication chip will back off and stop transmitting. Common practice is to re-initiate the chip to put it on-line again. If the chip still reaches the error limit and stops, the user application can try to re-initialize the chip after some delay.

Some transceivers for the CAN protocol have a special pin for disabling the transmit function, but it is seldom used due to the extra hardware required to control this pin. Another transceiver using two-wires has the capability to communicate (at a low speed) on only one wire in case of open or short circuits.

## 3.2 CAN Application layers

The CAN standard only defines the physical link and data link layers of the OSI model. To support systems development, various higher level layers have been developed. Their main purpose is to provide data exchange between computers while relieving the user of the details of the communication system. Two such protocols, which include some specific support for embedded real-time systems, are Volcano [CRT+98] and CAN Kingdom [Fre95].

One important prerequisite to achieve predictable timing behaviour is to ensure that each ‘message generator ’ is transmitting messages either periodically or with a maximum frequency. In order to do this the requirements of each data item to be communicated need to be defined including the data size and repetition rate. Based on this definition, data is grouped into messages and assigned a CAN identifier that is high enough to allow all data items to meet their specified requirements. In addition assumptions have to be made about the rate of CAN errors introducing error frames and retransmissions. Both Volcano and CAN Kingdom include such considerations to be able to predict the load and the corresponding delays of messages over CAN networks. CAN Kingdom in addition is designed to incorporate a global clock through clock synchronization.

An interesting property of Volcano is that an accompanying configuration tool incorporates analysis techniques developed in the research on fixed priority scheduling, see e.g. [ABD+95]. It turns out that the scheduling theory developed in the 70s is almost directly applicable to CAN - which is not so surprising since it is based on fixed priority scheduling. The configuration tool thus assists in identifier (i.e. priority) assignment and system verification.

Common for currently used protocols are that they provide little support with respect to for example redundancy management and timing determinism which affects both timing jitter and effective fault detection.

# 4 Problem Characterization and Future Work

For traditional distributed systems many solutions have been developed for basic problems such as atomic broadcast, membership services, clock synchronization etc. However, there is still a lot of work to do when taking safety and real-time requirements into consideration.

The approach taken in DICOSMOS will be to further study and investigate the requirements on the communication system; with this perspective proposed services and protocols will be further evaluated. In the following some specfic research problems in this direction are presented.

## 4.1 Fail operational communication through fail silent components

Fail silence is a very useful mechanism in building safety critical systems to ensure that no faulty data is propagated. Even though some checks can be done in hardware some faults are easier to detect in software. It is therefore important to provide strong mechanisms to ensure fail silence in the system both for the operating system and the communication system. It is of utmost concern that the communication interfaces are fail silent towards the communication media in order not to jeopardise the bus communication schedule for the rest of the nodes. Therefore different error detection mechanisms for the communication interface will be studied to obtain a fail silent communication interface.

For a double (FO) bus the nodes have to be connected to both buses. Since the buses should fail independently, they should also operate as independently as possible. A minimum requirement is that they are electrically isolated, but their logical connections should be as low as possible. For example, they should be as loosely synchronized as the real-time requirements allow. If the traffic on the two buses are slightly offset, transient failures will not hit both buses in the same way, and errors are more easily detected. Therefore to obtain the FO bus, synchronization mechanisms between the two fail silent buses will be studied and furthermore the internal cross coupling problems between the buses and the interfaces will be examined. At CTH a double communication bus is studied. This type of bus will tolerate one permanent failure and is called an FO bus (Fail Operational: After one failure it is still operational). To achieve an FO-bus, fail silence is used.

## 4.2 Time- vs. Event-Triggering

The big advantage with a TT triggered system is the determinism that is achieved since the scheduling of communication and all executable processes is done off-line. However one drawback with this is the low flexibility for handling sporadic events, such as for example the triggering of an airbag system. Another problem arises in complex TT-systems where we have different operational modes e.g., the start-up, flight and landing modes in an airplane. In such system it might be impossible to schedule all the messages for the different modes on the bus. One approach to handle this is to use similar modes on the bus i.e., different bus modes. How to handle sporadic events and mode changes in a safety critical system should be further investigated.

## 4.3 Consistency

Atomic Broadcast (AB) and Membership Agreement are costly even in a synchronous system. For example to ensure AB in one TDMA round, a node may have to stop to communicate to preserve the AB property if one received message is determined to be incorrect. To restore the normal operation of the node it will need to update its internal state and reintegrate with the communication system.

Using the knowledge of the control application it might be possible to relax the properties of Atomic Broadcast in the communication system. This will be investigated in a DICOSMOS case study. There also exists a lot of other interesting issues such as what type of information a control application needs from a Membership Service and how Atomic Broadcast affects the overall robustness of the communication.

In systems where we have two replicas we have no possibility to vote on values. This can be solved by enforcing replica determinism. However, Replica Determinism can be hard and costly to implement, therefore we will investigate the possibilities to handle this in the control layer i.e., by the control application, where necessary.

Finally all these types of support need an appropriate interface between the application and communication layers. The requirement of this type of interface will be studied within the DACAPO architecture.

## 5 References

[ASJS96] T. Abdelzaher, A. Shaikh, F. Jahanian, and K. Shin. RTCAST: lightweight multicast for real-time process groups. In *Proc. of IEEE Real-time technology and Applications Symposium*, Boston, 1996.

[ABD+95] Audsley N.C., Burns A., Davis R.I., Tindell K.W., Wellings A.J. (1995). Fixed priority pre-emptive scheduling: An Historical Perspective. *J. Real-Time Systems*, 8, 173-198. Kluwer. ISSN 0922-6443.

[BD87] Ö. Babaglou and R. Drummond. (Almost) no cost clock synchronisation. *Proc. 17th Annual IEEE Int. Symposium on Fault-Tolerant Computing*, FTCS-17, pp. 42-47, Pittsburgh, PA, USA, June 1987.

[BJO+93] O. Bridal, L. Johansson, J. Ohlsson, M. Rimen, B. Rostamzadeh, R. Snedsböl, J. Torin. *DACAPO: A Dependable Distributed Computer Architecture for Control of Applications with Periodic Operation.* Technical Report, No. 165, Dept. Computer Engineering, Chalmers University of Technology, Gothenburg, Sweden, 1993.

[BSJ94] Bridal O., R. Snedsböl, L. Johansson. On the Design of Communication Protocols for Safety-Critical Automotive Applications. In Proc. of *IEEE Vehicular Technology Conference*, Stockholm, Sweden, June, 1994.

[CAN90] Controller Area Network CAN, an In-Vehicle Serial Communication Protocol. In *SAE Handbook* 1992. SAE Press, pp. 20.341-20.355.

[CRT+98] Casparsson L., Rajnak A., Tindell K., Malmberg P. Volcano - a revolution in on-board communications. *Volvo Technology report*, 1998.

[Cri90] F. Cristian. Synchronous atomic broadcast for redundant broadcast channels. *J. of Real Time Systems*, vol. 2, pp. 195-212, 1990.

[Cri91] F. Cristian. Reaching agreement on processor-group membership in synchronous distributed systems. *Distributed Computing*, vol. 4, pp. 175-87, 1991.

[DJK+97] E. Dilger, L.A. Johanson, H. Kopetz, M. Krug, P. Liden, G. McCall, P. Mortara, B. Muller, U. Panizza, S. Poledna, A. Schedl, J. Söderberg, N. Strömberg, T. Thurner. Towards an Architecture for Safety-Related Fault-Tolerant Systems in Vehicles. *ESREL-97, International Conference on Safety and Reliability*, June 1997, (pp. 1021-1030).

[Fre95] Lars-Berno Fredriksson. *A CAN Kingdom*. Rev. 3.01, ©KVASER AB, 1990,91,92,95.

[GUT83] Guth R., Lalive d'Epinay T., The Distributed Data Flow Aspect of Industrial Computer Systems. *Proc. of the 5:th IFAC Workshop on Distributed Computer Control Systems*, Sabi-Sabi, South Africa, 1983.

[KG94] H. Kopetz and G. Grunsteidl. TTP - a protocol for fault-tolerant real-time systems. *IEEE Computer*, vol. 27, 1994, pp. 14-23.

[KGR91] H. Kopetz, G. Grunsteidl and J. Reisinger. *Fault-Tolerant Membership Service in a Synchronous Distributed Real-Time System*. Technical Report, Technical University Vienna.

[KO87] Kopetz, H. and W. Ochsenreiter. Clock Synchronization in Distributed Real-Time Svstems. *IEEE Trans. Computers*, Vol. 36, No. 8, Aug. 1987, pp. 933-940.

[Kop93] H. Kopetz. Should Responsive Systems be Event-Triggered or Time Triggered? *IEICE-Transactions on Information and Systems*, vol. E76-D, No 11, pp. 1325-32, 1993.

[LS96] H. Lönn and R. Snedsböl. Efficient synchronisation, atomic broadcast and membership agreement in a

TDMA protocol. In *Proc. of the 9th International Conference on Parallel and Distributed Computing Systems* (PDCS '96), Dijon, France, 1996.

[LS95] Lönn, H. and R. Snedsböl. Synchronisation in Safety-Critical Distributed Control Systems. *Proc. of IEEE International Conference on Architectures and Algorithms for Parallel Processing* 1995, Brisbane, Australia, 1995.

[NWT+98] Nilsson J., Wittenmark B., Törngren M., Sanfridson M. Timing Problems in Real-time Control Systems. In *Research problem formulations in the DICOSMOS project*. Technical Report, Dept. of Machine Design, The Royal Institute of Technology, Stockholm. TRITA-MMK 1998:20, ISSN 1400-1179, ISRN KTH/MMK/R--98/20--SE.

[Pol96] S. Poledna. *Fault-Tolerant Real-Time Systems: The Problem of Replica Determinism*. Kluwer Academic Publishers. 1996.

[RSB90] P. Ramanathan, K.G. Shin, R.W. Butler. Fault-Tolerant Clock Synchronization in Distributed Systems. *IEEE Computer*, Vol. 23, No. 10, October 1991, pp. 33-42.

[RB93] A. M. Ricciardi and K. P. Birman. *Process membership in asynchronous environments*. Technical Report TR93-1328, Dept. of Computer Science, Cornell University, February 1993.

[TES97] Törngren M., Eriksson C., Sandström K. (1997). Real-time issues in the design and implementation of multirate sampled data systems. In Preprints of SNART 97 *-Swedish National Association on Real-Time Systems Conference,* Lund, 21-22 August 1997.