Implementation Results of a Configurable Membership Service for Active Safety Systems

Carl Bergenhem[1], Christian Archer[2], Andreas Sjöblom[2], Johan Karlsson[2]

[1]*Department of Electronics,*
SP Swedish National Testing and Research Institute
S-501 15 Borås, Sweden
carl.bergenhem@sp.se

[2]*Department of Computer Engineering,*
Chalmers University of Technology
S-412 96 Göteborg, Sweden

Abstract

This article describes a configurable membership service. The function of a membership service is to give nodes in a cluster a consistent view of the status of entities in the system. An entity is a process in a node that manifests itself with a static communications slot. The mapping of processes to entities is done off-line. The service functions by distributed voting on the opinions from all nodes. This information is broadcast by nodes that detect a change among entities. During periods of no change, nothing is sent. The proposed service is implemented in a distributed embedded system with TTCAN-communication. The protocol is tested against faults that are injected with a simple application during runtime and some results are presented here.

1. Introduction

Research into the problem of membership agreement has been going on for the last 30 years, with still much activity. A survey of membership agreement can be found in [1]. This paper describes a configurable membership service in an automotive setting. Implementation is done for a cluster with a TTCAN network [2]. Communication in TTCAN is divided into a prescheduled static segment and a dynamic segment with CAN arbitration. Each node runs a very simple RTOS for task switching and synchronises its execution to the timing of the network. TTCAN lacks a built in membership service in contrast with TTP/C [3]. Its service is tightly coupled to node communication and always concerns the entire node. A trend in automotive electronics is towards fewer and more complex nodes that contain many independent processes [4]. This motivates a more fine grain service where individual processes are regarded.

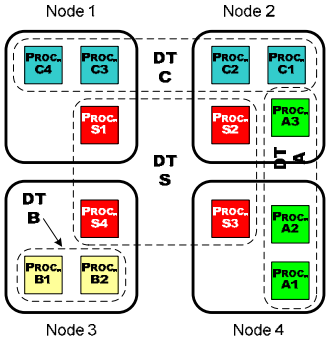

Figure 1: Several processes in different nodes constitute a distributed task (DT), e.g. DT S.

A distributed task (DT) consists of a number of possibly geographically distributed processes (throughout several nodes). It is basically a distributed application with constituent parts (processes) that communicate with each other over a network. Keeping track of the statuses of processes belonging to a DT may be done with a membership service in each node. Here the processes of the DT are the entities. An example of a DT is a distributed brake-by-wire (BBW) system which has a process running on a node situated at each wheel, as in "DT S" in

Figure **1**. The information from the membership service allows the distributed BBW system to adapt itself to the situation, i.e. use a suitable control algorithm.

In this research an entity is a process in a node that manifests itself with a static communications slot. Thus if the slot is silent, this is assumed to imply failure of the process being monitored by the membership service. A node can contain a number of processes and a selection of these will be entities. This mapping of processes to entities is done off-line. The membership service is used to achieve a consistent status of entities, locally at each node.

We assume that nodes and processes in nodes are fail-silent. Nodes may also suffer failures at incoming links and inconsistent faults may be introduced by the network. An example of this is marginal disturbances on the network causing some nodes to correctly receive a message while others do not. Therefore inconsistent opinions must be handled by the service even though no parts of the system exhibit arbitrary failure as such.

2. The membership service

Each node sends its prescheduled messages during the static segment. Messages from other nodes are noted and compared with what is expected and a local view is assembled. A deviation from the expected schedule in the form of silence is assumed to imply failure of the process that owns the slot. The service is based on the assumption that failures of processes occurs seldom. Only when a change is detected, a node uses the event-triggered communications segment to inform all other nodes of its local view. Each node runs its own instance of the membership service.

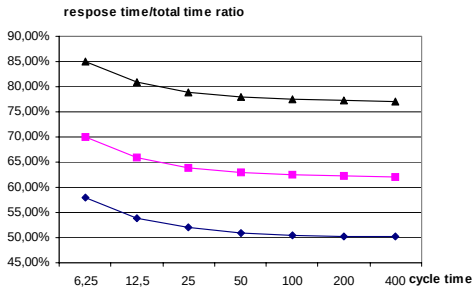

Figure 2: Ratio of response time and total time vs. the total cycle time. From top to bottom: Worst, average and best case.

A node's opinion about an entity can be either OK or silent. The dynamic segment is configured such that the messages that are part of the membership service are guaranteed to be sent. Thus opinion messages are sent only if the node notices change in the cluster to conserves network capacity which may be reused by other best-effort dynamic data traffic.

At the end of the dynamic segment all correct nodes have received opinions from those nodes in the cluster that have noticed change. Each opinion consists of a single bit for the perceived status of each entity in the cluster. All nodes will thus have a matrix with opinions from all nodes in the cluster. This is the object of further processing.

Each node performs a vote on each entity status in the collected opinions. The consistent result is the membership view during the last cycle and may be presented, after processing, to the applications in the node. Further processing may consist of quarantining (delayed notification) a failed or reintegrating entity for some cycles to be certain of the status. The voting process may be individually configured for each entity depending on the application. Critical applications may e.g. require that an entity is expelled from the group with no tolerance. Strict majority voting is used for all entities in the experiments described here.

The basic steps of the service are:

During the static segment:

1. Observe each entity (slot): OK or silence.
2. Compare with what is expected.
3. Build local opinion on each entity status: OK or Silence.

During the dynamic segment:

4. If there is change in status of any entity during last cycle: Broadcast local opinion.
5. If no change detected but other opinions are sent, force sending my opinion next cycle.
6. Receive local opinions from nodes or handle opinion absence (= no change noticed).
7. Distributed vote on outcome.
8. Process result and present to application.

Assisting functions:

9. Detect and handle possible crashed nodes (opinions were not sent even when forced).
10. Detect and handle possible incoming link failure of nodes.
11. Remove "bad" opinions from crashed or incoming link failure nodes.

The basic service also has assisting functions to detect incoming link failure and crashed nodes and is described in [5]. The goal is higher resilience to failure.

3. Results

The response-time of the service is measured and depicted in Figure 2. Here there are six entities in the cluster of five nodes. The total cycle length is increased by increasing the idle time between messages. The response time is measured in percentage of a complete cycle (which is different in each experiment). It is defined from the time that a change occurs until distributed consensus is reached. Response time depends on when in the cycle that the fault occurred. As expected, the response time increases linearly with an increasing cycle length. With very small cycle lengths, the response time is affected by the execution time of the voting procedure and the effect is more noticeable. More details and results from further tests can be found in [5].

4. Conclusions

A membership service has been implemented and tested in a time triggered distributed system. The service utilises the event and time-triggered communications mode that is available in TT-CAN. The service includes features for detection of node crash and incoming link failure.

Response time has been investigated for different number of entities, nodes and cycle length and is always less than one communication cycle. The performance of the membership service is affected by how the communication in the system is scheduled. Ideally this must be considered during design.

Tests where performed to evaluate the service under the influence of errors. These showed that the membership can handle errors occurring in the membership entity slots as long as not more than half of the nodes are affected. Crash detection and incoming link failure detection algorithms can handle faults as long as there are enough correct nodes to perform a vote.

5. Acknowledgements

This research will be conducted within the project CEDES, which is funded by the Swedish industry and government joint research programme IVSS - Intelligent Vehicle Safety Systems.

6. References

[1] C. Bergenhem "Survey of membership agreement protocols", Chalmers technical report 2005:19, Department of Computer Engineering, Chalmers University of Technology, Sweden.

[2] T. Führer, B. Müller, W. Dieterle, F. Hartwich, R. Hugel, M. Walther, "Time Triggered Communication on CAN (Time Triggered CAN- TTCAN)", Robert Bosch GmbH

[3] G. Bauer, M. Paulitsch, "An investigation of membership and clique aviodance in TTP/C" IEEE Reliable Distributed Systems 2000.

[4] N. Navet, Y. Song, F. Simonot-Lion, C. Wilwert, "Trends in automotive communication systems," Proc. of the IEEE, vol. 93, pp. 1204-1223, 2005.

[5] A. Sjöblom, C. Archer "Membership implementations on time triggered architectures", 2006, Department of Computer Engineering, Chalmers University of Technology, Sweden.