Implementation of a Flexible Membership Protocol on a Real-Time Ethernet Prototype

Raul Barbosa[†], António Ferreira[‡], and Johan Karlsson
Department of Computer Science and Engineering
Chalmers University of Technology
SE-412 96 Göteborg, Sweden

Abstract

This paper describes the implementation of a processor-group membership protocol in an experimental real-time network. The protocol is appropriate for fault-tolerant distributed systems using TDMA for scheduling messages. It allows nodes to maintain a consensus on the operational state of all nodes, in the presence of node failures and restarts, as well as network failures. The protocol is based on the principle that, in a system of n nodes, each node must acknowledge the messages from k other nodes in the membership group, where k can assume values between 2 and $n - 1$. Membership agreement is guaranteed if $f \leq k - 1$ failures occur during n consecutive transmission slots. We have implemented the membership protocol in a time-triggered network based on COTS Ethernet hardware, programmed to schedule messages according to the TDMA method.

1 Introduction

Group membership is a building block of fault-tolerant distributed systems. In such systems, the processing nodes must cooperate in the diagnosis and isolation of faulty nodes, and in the reconfiguration that must take place to handle node failures correctly. To achieve this, working nodes must maintain a consensus on the nodes that should participate in service delivery. The algorithms designed to provide this consensus are known as processor-group membership protocols.

In this paper we describe the implementation of a flexible membership protocol intended for hard real-time systems. The protocol is appropriate for systems with static communication scheduling according to the Time Division Multiple Access (TDMA) method. This method divides the medium access into transmission slots. In each TDMA round, nodes transmit a fixed amount of traffic in their preallocated slots. The TDMA method is widely used in safety-critical applications and network standards such as FlexRay [2], TTCAN [3], or TTP [4] use this method for scheduling static real-time traffic.

The implemented protocol is a modified version of the algorithm presented in [1]. It is based on the principle that each message sent by a node in the membership group should be acknowledged by k other nodes, in a system of n nodes. The value k can be set to any number between 2 and $n - 1$. Each message contains k acknowledgement bits. The reintegration algorithm imposes an overhead of one bit per message, for the reintegration flag. This flag is used to acknowledge successful reintegrations and to signal the reintegration order. The main modifications to the original protocol are the addition of self-diagnosis for nodes that are excluded from the membership and the introduction of a reintegration order, which nodes must follow when attempting inclusion in the set of operational nodes.

We have implemented the membership protocol in a prototype of a distributed time-triggered system. The network is based on COTS Ethernet hardware, programmed to schedule messages according to the TDMA method. This prototype implementation allowed us to test the feasibility and performance of the protocol. We present the results of the communication overhead and the latencies for agreement on departure and reintegration. Moreover, we report the estimates of memory requirements and CPU overhead by the protocol implementation.

2 Membership Protocol Description

The membership protocol relies on nodes observing the transmissions of other nodes to detect failures. It allows

[†]The work of Raul Barbosa has been supported by the Portuguese Fundação para a Ciência e a Tecnologia through doctoral grant SFRH/BD/18126/2004.

[‡]This work was performed while António Ferreira was an exchange student at Chalmers University of Technology, from the University of Coimbra, Portugal.

0-7695-3054-0/07 $25.00 © 2007 IEEE
DOI 10.1109/PRDC.2007.49

fault-free nodes to maintain an agreement on the operational state of all nodes in the presence of node failures and restarts, as well as network failures (lost/corrupted messages). Failures in the system can prevent messages from reaching their intended receivers. We distinguish between consistent failures and inconsistent failures. A consistent failure occurs when a message does not reach any of its intended receivers; an inconsistent failure causes a receiver to miss a message, while others receive the message correctly.

We also distinguish between transient failures and permanent failures. We assume that a transient failure only affects one message. If several consecutive messages are lost, for instance, due to electromagnetic interference on the network, then we consider this as a case of multiple transient failures. A permanent failure will remain in the system until it is repaired, and may affect one node, its outgoing link, its incoming link or a point-to-point connection between a node and the hub, when the network has a star topology.

Independent observations are consolidated to maintain membership agreement by means of an exchange of information, piggybacked on the periodic messages. This information consists of message acknowledgements and a reintegration flag. Each node will append k acknowledgement bits to its message, confirming (or refuting) the reception of each of the previous k messages *from the nodes in the membership*. (Such a node is said to be *sponsoring* its k predecessors.) A reintegration flag (*r-flag*) is also appended to each message to allow reinitialized nodes to reintegrate the membership.

The protocol is divided into two main parts: agreement on departure and agreement on reintegration. Agreement on reintegration can be further divided into three subprotocols: recovery of the membership state, agreement on reintegration ordering and the actual procedure for agreement on reintegration.

- *Agreement on departure* – Initially, the membership set contains the active processing nodes. The startup mechanism, responsible for establishing a sufficient level of network-wide synchronization, also supplies the set of initially active nodes to the membership service. A given node N_j will be removed from the *local membership view* of N_i if and only if N_i does not receive a message from N_j and no positive acknowledgement for that message is received from any of the sponsors of N_j.

- *Recovery of the membership state* – Assuming that fundamental data such as the communication schedule will survive a crash, a node is able to recover the membership state by listening to the incoming messages. Starting with its membership set empty, the node adds other nodes to the local view when their message is correctly received. The node is required to listen for one complete round of communication.

- *Agreement on reintegration ordering* – The protocol establishes a cyclic order that nodes must follow during reintegration. Each node has a single communication round where it can send a reintegration request. Agreement on the round number is kept by the membership nodes as the communication schedule progresses. Such nodes must signal the current round number explicitly in order for a failed node to become synchronized. The protocol supplies the round number to restarting nodes by signaling a delimiter pattern with the *r-flags* of nodes in the membership (one *r-flag* from each node in every round). Once a restarting node listens to the delimiter pattern, a new reintegration cycle has started.

- *Agreement on reintegration* – After restarting, a node obtains both the current round number and the membership state. This node then sends a reintegration request in its dedicated reintegration round. The reintegration request contains its view of the membership state. Upon the reception of such a request, any receiving node (in the membership) will verify if the view is correct. If it is, the receiving nodes set their *r-flag* to *true* in their following message. Once the communication round is completed the node is added to the membership.

2.1 The Membership Component

We have built a software component that implements the membership protocol. It consists of a C program that provides a set of functions for updating the membership set in response to events. The module is independent from any other software and provides a strictly reactive service, *i.e.*, executes only when a function is called in the context of an existing process. Thus, it is easily portable to any time-triggered system that meets the protocol's assumptions. The membership interface contains the following functions:

- `init(membershipView)` – Initializes the membership component with the given membership state.

- `messageReceived(membershipData)` – Updates the membership view upon message reception.

- `messageLost()` – Updates the local membership view in response to a lost message.

- `newRound()` – Updates the local membership view at the end of a communication round.

- `getMembershipVector()` – Returns the local membership view.

- `getAcknowledgements()` – Returns the acknowledgements and the *r-flag* for the next message.

3 The Real-Time Ethernet Prototype

We have implemented the membership protocol in a prototype of a distributed real-time system. The experimental real-time network is based on COTS Ethernet hardware. Using only software, the system is programmed to schedule messages according to the TDMA method. Figure 1 depicts our experimental setup.

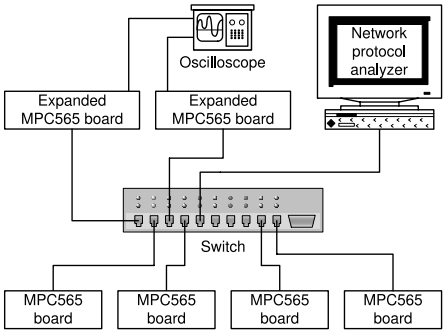

Figure 1. The experimental real-time Ethernet network.

The computer nodes in Figure 1 are Phytec's phyCORE-MPC565 development boards [6]. Each contains a Freescale MPC565 microcontroller, based on the PowerPC architecture. The boards include an RJ45 socket and an Ethernet controller.

The two boards depicted on the upper-left corner of Figure 1 are expanded with a custom board. We developed these expansion boards in order to output the internal clock of the nodes and to have a 7-segment display (for showing the number of active nodes in the membership). We connect the clock outputs to an oscilloscope that measures their synchronization. The expansion boards can be used with any processor board (to test for slight differences among nodes). Moreover, we connect a regular PC running Wireshark – a protocol analyzer – to the network, in order to verify the execution of the protocol. Each board executes a small software module that allows failure scenarios to be configured and tested.

The experimental network is based on a star topology with a central switch – HP's ProCurve Switch 2324. The Ethernet controller included in the boards runs at 10 Mbit/s (10Base-T standard). To maintain the TDMA schedule we implemented the daisy-chain clock synchronization algorithm [5]. This algorithm adjusts the clock of each node every time a new message is received. The adjustment is a fraction of the difference between the *expected* and the *actual* arrival time of a message.

3.1 Network Configuration

The length of the Ethernet frames can vary between 64 and 1518 bytes. We used 64-byte packets in our experiments – 46 bytes of payload data, 4 bytes for the CRC checksum and 14 bytes for the MAC header. The MAC header identifies the source address (*i.e.*, the message sender) and the destination address, which is set to *broadcast*. With this configuration the estimated propagation delay for the Ethernet frames was 215 μs.

The duration of a transmission slot was configured to 400 μs. Under these conditions the daisy-chain algorithm maintained the processor nodes synchronized within 3 μs. Table 1 summarizes the most important network parameters.

Parameter	*Value*
Number of nodes	6
Packet size	64 bytes
Transmission slot	400 μs
Reintegration cycle	36 ms
Communication round	2.4 ms
Clock skew (measured)	< 3 μs

Table 1. Configuration of the real-time Ethernet network and resulting clock skew.

The membership protocol was configured to 4 sponsors per node. Each 64-byte packet therefore included 5 bits of membership information (4 acknowledgements and 1 *r-flag*).

4 Network and Membership Performance

The nominal bandwidth of the network is 10 Mbit/s. However, real-time communication using TDMA must take into account the propagation delays and the clock skews to ensure that there are never two messages being transmitted at the same time. This is achieved by inserting guard times between messages. Due to these guard times, we estimate that our experimental network can achieve a maximum bandwidth of 3.3 Mbit/s using 1518-byte packets.

In our experiments, we used 64-byte packets and transmission slots of 400 μs, which results in a network bandwidth of 1.3 Mbit/s. This way, we can calculate the resource usage when the protocol executes at nearly the highest possible frequency for our setup. Since each frame reserves 18 bytes for the header and the CRC checksum, we have, for this configuration, 920 Kbit/s of *useful bandwidth* available for payload data (which includes the membership information).

In our experiments, each message had 5 bits of piggybacked membership information. Messages were sent once

every 400 μs. The bandwidth required by the membership service is therefore 12.5 Kbit/s. Since we have 920 Kbit/s of useful bandwidth available, the membership service imposes a 1.4% communication overhead. If we consider the network's nominal bandwidth of 10 Mbit/s, the membership's overhead is less than 0.2%. We emphasize that these values were obtained for 64-byte packet sizes, which provide the lowest useful bandwidth. Increasing the packet size would reduce the membership's communication overhead significantly.

A departure is detected by the membership when the node's last sponsor transmits its message. In the worst case, this may occur $n-1$ slots after the message is lost. Since a node may fail immediately after broadcasting a message, it may take n slots until a message is missed by the other nodes. The latency for agreement on departure is therefore $(6+6-1) \times 400\ \mu\text{s} = 4400\ \mu\text{s}$. This and other important latency values are shown in Table 2. It should be noted that these are calculated (not measured) values. The worst case latencies for reintegration occur when node 6 wishes to be reintegrated and starts listening on round 3; the node has to wait $2 \times 6 + 4 = 16$ rounds for the next delimiter pattern and then 12 (2×6) rounds to be reintegrated.

Activity	*Latency*
Agreement on departure of a crashed node	4.4 ms
Recovery of the round number	38.4 ms
Fault-free reintegration from restart	67.2 ms

Table 2. Node departure and node reintegration latencies (worst case).

A direct implementation of our protocol requires nodes to acknowledge their immediate predecessors. An important concern is therefore to ensure that nodes have enough time to react to received/lost messages. In our experimental setup, we have verified through extensive testing that the nodes were able to send their acknowledgements on time. However, for systems where nodes have a long reaction time, the order of the acknowledgements can be set in such a way that node N_i sponsors the nodes starting at N_{i-2}, instead of sponsoring its immediate predecessor.

Another important aspect of the implementation of membership protocols is that the processing capacity of nodes may be very limited. For our experimental setup, we estimate that the size of the code related to the membership service is less than 4KB; the data structures occupy 42 bytes in memory. We measured the CPU usage with and without the membership service enabled and an early estimation shows that the CPU overhead of the membership service is negligible.

5 Conclusion

This paper described the implementation of a processor-group membership protocol appropriate for hard real-time systems with TDMA communication scheduling. The protocol allows a trade-off between reliability and communication overhead to be made at design time. This flexibility is well suited for hard real-time systems – such systems impose hard deadlines for achieving consensus on membership changes, while they offer limited bandwidth for the implementation of the membership services.

The membership protocol was implemented on an experimental time-triggered network based on COTS Ethernet hardware. Through extensive testing, we verified that the protocol maintains membership agreement under multiple failures. This implementation allowed us to calculate the departure/reintegration latencies and the bandwidth consumption in a realistic configuration. Configuring the network for the smallest packet size (64 bytes) the useful bandwidth is 920 Kbit/s. In this case the membership protocol has a 1.4% communication overhead. Increasing the packet size would reduce this overhead significantly (the useful bandwidth would increase and the protocol would execute at a lower frequency).

Acknowledgements

The authors would like to thank Mikael Hedén for his contribution to the development of the real-time Ethernet prototype.

References

[1] R. Barbosa and J. Karlsson. Flexible, cost-effective membership agreement in synchronous systems. In *Proceedings of the 12th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC'06), Riverside, California, USA*, pages 105–112, Dec. 2006.

[2] J. Berwanger et al. FlexRay: The communication system for advanced automotive control systems. *SAE transactions*, 110(7):303–314, 2001.

[3] T. Führer, B. Müller, W. Dieterle, F. Hartwich, R. Hugel, and M. Walther. Time triggered communication on CAN (Time Triggered CAN - TTCAN). Technical report, Robert Bosch GmbH, 2000.

[4] H. Kopetz and G. Bauer. The time-triggered architecture. *Proceedings of the IEEE*, 91(1):112–126, Jan. 2003.

[5] H. Lönn. A fault tolerant clock synchronization algorithm for systems with low-precision oscillators. In *Proceedings of the 3rd European Dependable Computing Conference (EDCC-3), Prague, Czech Republic*, volume 1667 of *Lecture Notes in Computer Science*, pages 88–105. Springer, Sept. 1999.

[6] PHYTEC Meßtechnik GmbH. *phyCORE-MPC565 Hardware Manual (available from www.phytec.de)*, 2004.