Flexible Membership Agreement for Time-Triggered Networks

Raul Barbosa and Johan Karlsson
Department of Computer Science and Engineering
Chalmers University of Technology
Göteborg, Sweden
{rbarbosa, johan}@ce.chalmers.se

Abstract

This paper presents a processor group membership protocol appropriate for distributed fault-tolerant real-time systems that utilize time-triggered communication.

1 Introduction

Group membership is often a cornerstone of the design of fault-tolerant distributed systems. The entities, or nodes, which constitute such systems, must cooperate in providing a correct service to its user(s), even in the presence of faults. A correct behavior of the system requires that the non-faulty nodes have access to timely and consistent information about the operational state of all nodes, i.e. the *membership*. The algorithms devised to maintain this information, preferably in a decentralized manner, are usually known as processor group membership protocols.

This paper proposes a processor group membership protocol for systems relying on synchronous communication, where messages are transmitted within a known amount of time and processing nodes have a global notion of time. The protocol is appropriate for hard real-time systems which utilize network standards such as FlexRay [1] or TTCAN [2]. Such systems impose hard deadlines for achieving consensus on membership changes, while they offer limited bandwidth for the implementation of the membership services.

2 System Model

We consider a set of processing nodes linked by a time-triggered broadcast network. We assume that the network is based on a *bus topology* or a *star topology*. Each processor (node) has a single dedicated slot on the network which it uses to send messages according to a cyclic schedule.

Nodes are assumed to be *fail-silent*, i.e. either correct results or no results are broadcasted, or *fail-reporting*, i.e. either the correct result or a failure report (specifying the causes of failure) is produced.

Under fault-free conditions the node which has the right to broadcast on the network (i.e. the sending node) does so by sending a message. If, at some point, a node does not receive a valid message, due to loss or corruption, a failure has occurred. It could be a *failure of the sending node*, a *failure of the receiving node* or a *network failure*.

In our system model, we assume that failures can occur in the nodes, their incoming and outgoing links, and the network itself. We map these failure types into four different failure classes according their persistence (*permanent* or *transient*) and whether their impact on the system is *consistent* or *inconsistent*.

A transient failure only affects one message. If a fault leads to the corruption of two messages we consider this a double failure. A permanent failure will remain in the system until it is repaired. We classify a failure as consistent if all receiving nodes fail to receive a message correctly. On the other hand, an inconsistent failure occurs if only one node fails to receive a message correctly.

Table 1 shows how the different types of component failure are mapped into different failure classes.

<table>
<tr><th></th><th>Transient</th><th>Permanent</th></tr>
<tr><td>Sending Node</td><td rowspan="3">1 · Transient Consistent</td><td rowspan="3">3 · Permanent Consistent</td></tr>
<tr><td>Outgoing Link</td></tr>
<tr><td>Network (Consistent)</td></tr>
<tr><td>Network (Inconsistent)</td><td rowspan="3">2 · Transient Inconsistent</td><td rowspan="3">4 · Permanent Inconsistent</td></tr>
<tr><td>Incoming Link</td></tr>
<tr><td>Receiving Node</td></tr>
</table>

Table 1. Failure classes.

3 Protocol Specification

Let N denote the totally ordered set of processing nodes $N_1, N_2, \ldots, N_n$. We exclude systems with $n \leq 2$ processing nodes from our analysis, as membership agreement is

trivially achieved in this case.

Each node has access to a global view of time. This is supplied by clock synchronization mechanisms. In what concerns the membership problem, the required granularity of time is the *slot*. Hence, each node uses the global variable $slot$. The predicate $slot(i)$ is defined, for each synchronous step, as $true$ if and only if $i = slot \mod n$. It thus indicates which of the processing nodes N_1, N_2, ..., N_n, is the sending node in each bus slot.

Each node has a local variable $membership$, which represents its local view of the membership. The goal of the protocol is to ensure that all nodes maintain timely and consistent views of the membership.

3.1 Agreement on Departure

The membership protocol relies on the periodic messages sent by each node to piggyback a field containing a sequence of k acknowledgement bits. This ack field is used to confirm (or refute) the reception of each of the previous k messages *from the nodes in the membership*.

A node N_i is said to be *sponsoring* node N_j if, along with its message, the acknowledgement of the last message from N_j is attached. Only nodes in the membership are involved in sponsoring relations. We define the predicate $sponsor(i, j)$ as $true$ if and only if N_i is sponsoring N_j. The predicate $lastSponsor(i, j)$ is defined to be $true$ if and only if $sponsor(i, j) = true$ and $sponsor(p, j) = false$, where N_p is the immediate successor of N_i in the $membership$.

Under normal conditions each node will have k sponsors (and will be sponsoring k nodes). Thus, k stipulates how resilient will the protocol be to near-coincident failures, at the expense of communication overhead.

Initially, the $membership$ contains all processing nodes. Each node has a boolean array $present$. All the elements of $present$ are initialized to $true$. The $present$ array will be updated by each node upon message reception/loss. When a message expected from N_i is not received $present[i] \leftarrow false$. When a positive acknowledgement referring to a message from N_i is received from one of its sponsors, $present[i] \leftarrow true$.

The sending node N_i sets the ack bit which refers to its sponsored node N_a to $present[a]$. Thus $ack_i[a] = true$ if and only if N_i received the previous message from N_a or any of its acknowledgements.

When the slot of a node N_i in the $membership$ comes to an end, all nodes, including the sending node, synchronously execute a decision-making step. Node N_j, such that $lastSponsor(i, j) = true$, will be removed from the $membership$ if and only if $present[j] = false$.

3.2 Agreement on Reintegration

The reintegration procedure commences when $slot(r) = true$, $N_r \notin membership$ and a message is correctly received by at least one node. This means that N_r is up and running again and the receiving nodes will set their local variable $reintegrating \leftarrow r$.

At each step the sending node piggybacks a $r\text{-}ack$ bit to its message. The $r\text{-}ack$ bit serves the purpose of notifying the nodes which might have lost the message from the reintegrating node. Therefore, a sending node will set $r\text{-}ack \leftarrow true$ if and only if $reintegrating \neq null$.

The synchronous decision-making step about the reintegration of node N_r is executed when $reintegrating = r$ and the slot of the first predecessor of N_r contained in the membership comes to an end. This is processed by i) adding N_r to the $membership$, ii) setting $reintegrating \leftarrow null$ and iii) setting $present[r] \leftarrow true$.

Any node which receives a message reporting $r\text{-}ack = true$ when $reintegrating = null$ has certainly missed a reintegration message (from an unknown source) and will therefore be left out of the membership.

4 Discussion and Conclusions

The protocol specified in this paper is flexible in the sense that the tradeoff between reliability and communication overhead can be defined at design time, by choosing the number k of sponsors per node.

The value of k can be set to any number between 2 and $n - 1$. Agreement on departure is guaranteed if no more than $f = k - 1$ failures occur in the same communication round, while at most one node can be reintegrated into the the membership per communication round. Listening on the bus is enough for a restarted node to recover the state of the membership, before attempting reintegration.

Additionally, agreement on node failure (membership departure) and agreement on node recovery (membership reintegration) are handled by two different mechanisms. This allows decreasing the cost of fault-tolerant reintegration by implementing the reintegration procedure using the event-driven fraction of the communication schedule (provided by standards such as FlexRay and TTCAN).

References

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

[2] T. Führer et al. Time triggered communication on CAN (Time Triggered CAN - TTCAN). Technical report, Robert Bosch GmbH, 2000.