
Kohei SAKURAI†,††, Masahiro MATSUBARA†, Nonmembers, and Tatsuhiro TSUCHIYA††, Member

1. Introduction

Time-Triggered (TT) platforms are increasingly common for real-time, life-critical systems. In this paper we focus on automotive applications. For such applications, FlexRay has been widely adopted as a major TT platform [1]. FlexRay allows timely and reliable communication between ECUs (Electronic Control Units) or simply nodes, distributed in a vehicle. However, nodes may fail, and thus, should this happen, it would be necessary to detect the failed node and isolate it from the set of active nodes. At the same time, the decision on which nodes are faulty and should be isolated must be unanimous.

Membership is the functionality that provides a consistent view of active nodes to each node. Membership for TT architectures is well studied [2]–[4]. In fact we implemented the membership functionality in our drive-by-wire prototype using the FlexRay system [5]. A performance analysis of this prototype system showed that the computation overhead incurred by the membership functionality increased quadratically in the number of nodes. It also revealed that when the number of nodes was six, the overhead consumed more than five percent of the whole computation time available at each ECU. In practice, this overhead is prohibitively large. A further analysis showed that a significant part of the computation was consumed in voting on syndromes collected from all nodes. This implies that reducing the overhead requires reducing the computation time used for the voting process. Here, a syndrome is a vector that represents the status of nodes perceived by a single node. By performing voting on syndromes, all nodes can obtain a consistent health vector which represents the nodes’ status.

To address the problem of overhead reduction, this paper proposes a method which we call voting sharing. In voting sharing, each node performs voting only on a single entity, rather than the whole vector, of syndromes. By sharing the voting result, all nodes can obtain a consistent health vector without performing the costly voting operation.

The remainder of the paper is organized as follows. Section 2 describes the model of TT systems. Section 3 describes the existing approach to membership in TT architectures. Then Sect. 4 describes voting sharing and shows the properties it guarantees. Section 5 shows some experimental results on execution time. Finally Section 6 concludes the paper with possible directions of future work.

2. Model of Time Triggered Systems

We consider a generic TT system which encompasses FlexRay. The system consists of \( n \) nodes having unique IDs \( 1, 2, \ldots, n \). The communication network is the bus type with TDMA (time division multiple access). The system runs by consecutively executing synchronous rounds, starting from round 1. A node is assigned its own sending slots where it can send a message frame. A frame sent by a node is received by all the other nodes. All sending slots are statically scheduled in the design time and thus never overlap. Every node sends a frame at least once in one round.

Faults of nodes are observed as communication errors. A fault that a node suffers is either benign or symmetric Byzantine. If a node suffers a benign fault at round \( k \), then the message frame sent by the node in the round is lost. Hence all nodes can locally detect it in round \( k \). If a node suffers a symmetric Byzantine fault at round \( k \), then, at the same round, all nodes receive from the faulty node the same erroneous message which does not conform to the protocol specification.

Nodes are correct, obedient, or symmetric Byzantine faulty. Correct nodes follow the protocol specification and suffer no faults. Obedient nodes follow the specification but they may or may not suffer benign faults. Correct nodes are thus also obedient. Symmetric Byzantine faulty nodes do not follow the specification and suffer symmetric Byzantine faults. We assume:

\[
2s + b + 1 < n
\]

(1)

where \( s \) is the number of symmetric Byzantine faulty nodes, and \( b \) is the number of benign faulty nodes.
3. Existing Approach to Membership

3.1 Overview

Some membership protocols for TT systems are proposed in, for example, [2], [5]. In this paper we consider the one shown in [5] as a basis for our new approach, though these protocols are very similar.

The existing approach to membership starts at every round \( k \) the sequence of three phases, which spans two consecutive rounds, namely, rounds \( k \) and \( k + 1 \), as follows. In round \( k \), each node evaluates other nodes’ status locally by receiving and evaluating frames sent by other nodes (EVA phase).

In the following round, round \( k + 1 \), the local syndrome, which represents the local view of other nodes’ status evaluated in round \( k \), is exchanged among all nodes (EXC phase). A local syndrome for node \( p \) is a binary \( n \)-tuple \((s_1, s_2, \ldots, s_n)\) where \( s_i = 1 \) if \( p \) evaluates node \( i \) as non-faulty and \( s_i = 0 \) otherwise.

In the same round (i.e., round \( k + 1 \)), each node determines node status by voting on the exchanged local syndromes (DET phase). Specifically, a node constructs a diagnostic matrix where the \( i \)th row is the syndrome received from node \( i \) and the \( j \)th column is a vector representing the evaluation for node \( j \) from all nodes. An element \( e_{i,j} \) is either \( 1, 0, \) or \( - \). The case \( e_{i,j} = - \) occurs if node \( i \) failed to send its local syndrome in round \( k + 1 \) because of its benign fault. The opinion of a node about itself, i.e., \( e_{i,i}, 1 \leq i \leq n \), is considered unreliable and thus is assigned a special value – which specifies that it is discarded in voting.

By performing hybrid voting [6] over each of the columns, each node obtains a binary \( n \)-tuple called a health vector where the \( i \)th element represents whether node \( i \) is non-faulty or faulty. Specifically, hybrid voting is defined as follows:

\[
H_{\text{maj}}(V) = \begin{cases} 
0 & N_0(V) > N_1(V) \\
1 & N_0(V) \leq N_1(V)
\end{cases}
\]

where \( V \) is the column that is voted on, and \( N_0(V) \) and \( N_1(V) \) are the number of occurrences of 0 and 1 in \( V \).

The health vector is then updated to accuse Byzantine faulty nodes. If the current health vector \( \hat{hv} \) and the local syndrome sent from node \( i \) respectively have 0 and 1 or 1 and 0 on the same position, then node \( i \) is identified as a faulty node. The final health vector \( hv \) is obtained from \( \hat{hv} \) by setting the value on the \( i \)th position to 0 for all such nodes \( i \).

Based on the health vector, a node updates counters associated with nodes and possibly eliminates faulty nodes from the active ones. For example, one can eliminate a node by counting the number of times when the node is diagnosed as faulty and by deciding its elimination when the counter exceeds a predefined threshold. The process of updating the counters can be regarded as executing a stateful function that takes a health vector as input and produces a set of active nodes as output. We denote by \( \text{updateCounter}(hv) \) this counter updating function.

Note that multiple instances of this sequence of phases are executed concurrently as shown in Fig. 1. The node status evaluation process (EVA) can be done concurrently in the status data exchanging phase (EXC) by evaluating the exchanged status data. Due to this pipeline-like process execution, a fault detected in a round can be identified in the next round.

As an example, consider a system consisting of \( n = 5 \) nodes where node 2 is symmetric Byzantine faulty and the other nodes are obedient. Suppose that node 1 suffers benign faults in rounds \( k \) and \( k + 1 \). Also suppose that node 2 sends an erroneous local syndrome in round \( k + 1 \). Then, in round \( k + 1 \) each obedient node forms a diagnosis matrix as follows:

\[
\begin{bmatrix}
-1 & 1 & 1 & 1 & 1 \\
1 & -1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 & 1
\end{bmatrix}
\]

Upon forming the matrix, every obedient node performs hybrid voting on each column. In hybrid voting, \( - \) and \( 0 \) values are first discarded and then voting is performed on the remaining values. In this case, all obedient nodes obtain the following temporal health vector.

\[
\hat{hv} = (0, 1, 1, 1, 1)
\]

Then Byzantine faulty nodes are accused. Of the five syndromes, only the one from node 2 has different 0/1 values from \( \hat{hv} \). Specifically, the syndrome from node 2 differs from \( \hat{hv} \) in the first, third, and fifth positions. To record that node 2 has been identified as a faulty node, the health vector is updated by setting the second position to 0.
hv = ⟨0, 0, 1, 1, 1⟩

Finally, each node updates the set of active nodes by executing the following operation:

\[
active\_nodes \leftarrow updateCounter(hv)
\]

3.2 Properties

Lemma 1: Node i is diagnosed as faulty in the temporal health vector hv in round k + 1 by any obedient node if and only if i suffers a benign fault in round k.

Proof: If i suffers no benign fault in round k, then no obedient node sets the ith bit to 0 in its local syndrome in round k + 1. From (1), we have:

\[
s < n - s - b - 1
\]

Let p be any obedient node (which is possibly i) and let \( V_i \) denote the ith column of p’s diagnosis matrix. Then \( N_0(V_i) \leq s \) and \( N_1(V_i) \geq n - b - s - 1 \). By (2), \( N_0(V_i) < N_1(V_i) \); thus \( H-maj(V_i) = 1 \) and node i is diagnosed as correct in hv. If i suffers a benign fault in round k, then no obedient node sets the ith bit to 1 in its local syndrome in round k + 1. Because of the same argument as in the case where i suffers no benign fault, \( H-maj(V_i) = 0 \) and node i is diagnosed as faulty in hv.

Lemma 2: Node i is accused as faulty in the Byzantine node accusation by any obedient node in round k + 1 if and only if i sends an erroneous local syndrome in round k + 1.

Proof: Let p be any obedient node. If i sends a correct local syndrome in round k + 1, then by Lemma 1, p’s temporal health vector hv is the same as the local syndrome received from i, and thus i is not accused. If the local syndrome sent by i is lost, then i is not accused in the Byzantine node accusation by p. If i sends an erroneous local syndrome in round k + 1, then the local syndrome differs from the correct one with respect to evaluation of some node j(\( \neq i \)). Let \( V_j \) be the jth column in p’s diagnosis matrix and \( v \in \{0, 1\} \) and \( 1 - v \in \{0, 1\} \) be the correct and erroneous values for the evaluation of node j. Because of the same argument as in Lemma 1, \( N_h(V_j) > N_{1-v}(V_j) \). Therefore i is accused as faulty by p in the Byzantine node accusation.

Lemma 3: The following two properties hold:

- Correctness: a correct node is never diagnosed as faulty in the health vector of any obedient nodes.
- Completeness: a faulty node that suffers a benign fault in round k or sends an erroneous local syndrome in round k + 1 is always diagnosed as faulty in the health vector of all obedient nodes in round k + 1.

Proof: The lemma directly follows from Lemmas 1 and 2.

Lemma 4: Consistency: the health vector is agreed by all obedient nodes in each round.

Proof: Because of Lemma 3, the set of nodes identified as faulty in the health vector is agreed by all obedient nodes. Hence the lemma follows.

Lemma 5: Consistent Isolation: the set of active nodes is agreed by all obedient nodes in each round.

Proof: Let hv denote the health vector obtained by an obedient node. Because of Lemma 4, hv is agreed by all obedient nodes. Since the updating function updateCounter is deterministic, its output updateCounter(hv) is also agreed by all obedient nodes.

3.3 Problem

The execution time needed to compute the health vector by each node increases at least quadratically with respect to the number of nodes. The performance analysis using our earlier prototype implementation revealed that when the round length was 5ms and the number of nodes was six, the execution time of the membership protocol consumed more than five percent of the whole available CPU time. This ratio might sound small but in practice this is prohibitively large, because CPU time is already a very scarce resource. For automotive systems, it is ideal to assign as much CPU time as possible to controlling applications in order to achieve better driving performance and safety. Thus it is not practically acceptable to spend that much amount of CPU time only for membership service.

The earlier system was equipped with 40MHz CPU at each node. Using faster CPUs could mitigate the problem to some extent. However, the demand on high responsiveness of driving control is increasing at the same time. For example, today’s automotive controlling systems often require a much shorter round time than the one in our prototype system. Such increasing demand on shorter round time easily offsets the increase of CPU performance.

4. Voting Sharing

To address the above mentioned problem, we propose voting sharing. The idea of this technique is to have each node perform voting to detect a fault in a single node and to share the voting result with all the nodes. Each node has a responsibility to vote on the exchanged local syndromes with respect to one specific node and to broadcast the result in the following round. As a result, the computation cost required for voting is reduced compared to the existing approach which performs voting for each of the n nodes.

4.1 Protocol

The execution of the protocol now involves three rounds, instead of two rounds (Fig. 2).

In round k, a node locally evaluates other nodes’ status, as in the existing approach (EVA phase).

In round k + 1, the local syndrome representing errors
observed at round $k$ are broadcast, just as in the existing approach (EXC phase). Unlike in the existing approach, each node $p$ performs voting only for a single node: Node $p$ extracts, from each of the local syndromes received, one element that is associated with a node that $p$ is responsible for. We denote by $\text{target}(p,k)$ the node for which $p$ is responsible for diagnosis in the protocol execution that starts from round $k$. Thus node $p$ collects $n - 1$ values (excluding $\text{target}(p,k)$’s own evaluation), each of which is either 1, 0, or $e$. Then node $p$ performs hybrid voting on these $n - 1$ values. The result of the voting is either 1 or 0. Following the hybrid voting, node $p$ executes the Byzantine node accusation described in Sect. 3.1 by comparing the voting result with the $n - 1$ values. Finally, the result of the diagnosis is encoded as an $n$-bit vector where 0 on the $i$th bit represents that node $i$ is diagnosed as benign faulty ($\text{target}(p,k) = i$) or Byzantine faulty ($\text{target}(p,k) \neq i$) (VT phase).

In round $k + 2$, the results of the voting and Byzantine node accusation are broadcast (EXC$^V$ phase). This phase can be combined with the EXC phase of the next protocol execution by adding the $n$-bit diagnosis data to the local syndrome. Each node collects diagnosis results from $n$ nodes, including itself. Then, in the DET phase, the node forms a health vector $\langle s_1, s_2, \ldots, s_n \rangle$ by computing a bitwise AND of all received $n$-bit diagnosis vectors. Upon obtaining a health vector $hv$, then the set of active nodes is updated to the return value of function $\text{updateCounter}(hv)$. The function $\text{updateCounter}(hv)$ is deterministic.

For any round $k$, $\text{target}(p,k)$ must meet the following properties:

$\forall p, q : p \neq q \Rightarrow \text{target}(p,k) \neq \text{target}(q,k)$

$\forall p : p \neq \text{target}(p,k)$

The first property states that two different nodes are responsible for voting for two different nodes. This property ensures that in every round, every node has its unique node that is responsible for voting for it. The second property signifies that such a voter node is different from the node diagnosed by that voter node.

Note that in an execution of the proposed protocol that starts from round $k$, node $i$ is diagnosed twice: in round $k + 1$ and in round $k + 2$. When necessary, in the latter case we say that a node is diagnosed (as faulty or non-faulty) in the health vector.

4.2 Properties

Here, we prove some key properties of voting sharing. Consistency and consistent isolation hold straightforwardly.

**Theorem 1:** (consistency) The health vector is agreed by all obedient nodes in each round.

*Proof:* When a node sends its diagnosis result, the message is either received correctly by all nodes, lost due to a benign fault, or received incorrectly by all nodes due to a symmetric Byzantine fault. Hence the health vector finally obtained is identical among all obedient nodes.

**Theorem 2:** (consistent isolation) The set of active nodes is agreed by all obedient nodes in each round.

*Proof:* The counter updating function $\text{counterUpdate}(hv)$ is deterministic (see Sect. 4.1). Because of Theorem 1, the health vector $hv$ is identical for all obedient nodes. Hence the active nodes, which are updated to the output of the function in every round, are always identical for all obedient nodes.

Correctness and completeness hold in a somewhat weaker form than the existing method.

**Theorem 3:** (correctness) Correct node $q$ is never diagnosed as faulty in the health vector in round $k + 2$ if all nodes are obedient.

*Proof:* Suppose that $q$ is a correct node and all nodes are obedient. From Lemmas 1 and 2, $q$ is always diagnosed as correct by any node in round $k + 1$. In round $k + 2$ the diagnosis result sent by each node is either correctly broadcast or simply lost, because the node is obedient. Hence $q$ is never diagnosed as faulty in round $k + 2$ by any nodes in their health vector.

**Theorem 4:** (completeness w.r.t benign faults) If node $q$ suffers a benign fault in round $k$, then $q$ is diagnosed as faulty in round $k + 2$ by all obedient nodes if $q$’s voter node $p$ (i.e., the node such that $\text{target}(p,k) = q$) is obedient and suffers no fault in round $k + 2$.

*Proof:* From Lemma 1, if node $q$ suffers a benign fault in round $k$, then $q$ is always diagnosed as faulty by voter node $p$ in round $k + 1$ if $p$ is obedient. If no fault occurs in the voter node in round $k + 2$, then the voting result 0 is correctly broadcast and occurs in the health vector of all obedient nodes in the round.

**Theorem 5:** (completeness w.r.t symmetric Byzantine faults) Suppose that a symmetric Byzantine faulty node $q$ sends an erroneous local syndrome in round $k + 1$ and that the erroneous syndrome differs from the correct one with respect to evaluation of a node $i$. Then $q$ is diagnosed as faulty in round $k + 2$ by all obedient nodes in their health vector if the node $p$ such that $\text{target}(p,k) = i$ is obedient and suffers no fault in round $k + 2$.

*Proof:* By the argument in Lemma 2, if node $p$ such that $\text{target}(p,k) = i$ is obedient, then $q$ is always diagnosed as faulty by $p$ in the Byzantine node accusation in round $k + 1$. If no fault occurs on $p$ in round $k + 2$, then the diagnosis result is correctly broadcast and reflected in the health vector of all obedient nodes in the round.

These weaker guarantees pose the following problems.
• False negatives: diagnosing faulty nodes as non-faulty.
• False positives: diagnosing correct nodes as faulty.

These problems are mainly caused by Byzantine faulty vot-
ers. In the next section, we show how one can address these
problems.

4.3 Rotating Voters

To mitigate the above problems, we propose the use of rotating voters. The idea is to change the voter node for each
node in every round so that a node becomes the voter
node for any other node in any consecutive \( n - 1 \) rounds.
More concretely, node \( p \) becomes the voter of node \( p + 1 \)
in round 1 and then changes the node it is responsible for
(i.e., \( \text{target}(p, k) \)) as \( p + 2, p + 3, \ldots, n, 1, 2, \ldots, p - 1 \). This rotation is repeated every \( n - 1 \) rounds. Note that the rotating voter scheme ensures the two conditions on \( \text{target}(p, k) \)
 stated in Sect. 4.1.

By means of rotating voters and the algorithm design of the
counter updating function \( \text{counterUpdate}(hv) \), the two
problems can be mitigated as follows.

A false negative with respect to node \( q \) occurs if all
nodes that should be able to diagnose \( q \) as faulty happen to
be faulty simultaneously in the same round. Rotating voters
ensure that any node always becomes a voter for any other
node in every consecutive \( n - 1 \) rounds. Thus if \( q \) suffers
faults intermittently or permanently, then it is safely diag-
nosed as faulty by correct nodes.

The case of false positives is trickier than the other two
cases, because a Byzantine voter can repeatedly produce in-
correct diagnosis results for any correct nodes. Therefore
if the counter updating function simply counted the times
when each node is diagnosed as faulty, then it would lead to
a rapid and undesirable shrink of the set of active nodes. A
possible solution to this is to offset the effects of incorrect
diagnosis with those of correct ones. This solution can be
implemented by, for example, decreasing the counter if the
node is diagnosed as correct. Another approach could be to
use two counters for each node that represent penalty and
reward, as has been done in the p-r algorithm proposed in
[2].

5. Experiment Results

This section presents the results of our experiment. We de-
veloped a prototype system that was equipped with four to
eight nodes. A node was equipped with a 60MHz CPU.

The results are summarized in Table 1. The row “vot-
ing” shows the time required for 1) calculating health vec-
tors (DET phase) in the existing membership, and 2) hybrid
voting of the target node (VT phase) and calculating health
vectors (DET phase) in the voting sharing. The row “all”
shows the total time used for the membership service includ-
ing the process time to receive and send message frames.

Though various processing overheads are added to the
pure voting calculation, which is reduced by a factor of the
number of nodes, the calculation time for the voting pro-
cess in the voting sharing is reduced to 42% compared with
the existing membership for eight nodes. The total time for
the membership service is reduced by 11%, which is sub-
stantially effective for industrial embedded systems with ex-
tremely limited hardware resources.

6. Conclusion

To reduce the overhead incurred by fault diagnosis, we pro-
posed voting sharing. We showed the properties ensured by
the proposed protocol. Although the proposed protocol has a
cost of small degradation of diagnosis accuracy, the over-
head can be substantially reduced. A possible future direc-
tion is to mitigate the accuracy degradation by, for example,
the design of the counter updating algorithm.

References

[2] M. Serafini, P. Bokor, N. Suri, J. Vinter, A. Ademaj, W. Brandstätter,
F. Tagliabù, and J. Koch, “Application-level diagnostic and member-
ship protocols for generic time-triggered systems,” IEEE Trans. De-
for safety related systems using TT/ET communication scheduling,”
Proc. 14th IEEE Pacific Rim International Symposium on Dependable
[4] H. Pfeifer, “Formal verification of the TTP group membership algo-
rithm,” Proc. Joint International Conference on Formal Description
Techniques for Distributed Systems and Communication Protocols
and Protocol Specification, Testing and Verification (FORTE/PSTV
[5] K. Sakurai, M. Matsubara, and M. Hoshino, “Membership middle-
ware for dependable and cost-effective X-by-wire systems,” SAE Int.
2009.
tive consistency under hybrid fault models,” Proc. 23rd Int’l Symp. on
tolerance, from theory to reality,” Proc. 22nd International Conference
on Computer Safety, Reliability, and Security (SAFECOMP), pp.235–