Verification of Cooperative Transient Fault Diagnosis and Recovery in Critical Embedded Systems

Zibouda Aliouat and Makhlouf Aliouat
Department of Computer Science, University of Ferhat Abbes, Algeria

Abstract: *The faults caused by ambient cosmic radiation are a growing threat to the dependability of advanced embedded computer systems. Maintaining availability and consistency in distributed applications is one of the fundamental attribute in building complex critical systems. To achieve this, a key factor is the ability to detect the fault and handle it by means of recovery. Such systems can use membership protocols designed to provide this function. The objective of membership protocol is to give all entities of every node in the cluster a consistent view of the system status, all within a pre-defined time. This paper describes a formal analysis of an extension of the group membership algorithm implemented in the time-triggered protocol. The proposed extension is to allow nodes reintegration after transient fault. We provide a detailed analysis of properties of formal model of the algorithm. The paper is intended to verify the safety and liveness properties that the protocol must satisfy. The correctness of the protocol is verified by the PVS theorem prover.*

Keywords: *Group membership protocol, formal verification, fault-tolerant distributed algorithm, node reintegration.*

Received February 25, 2010; accepted August 10, 2010

1. Introduction

Fault-tolerant computer systems are being used more and more in complex and safety critical applications particularly in sensitive areas like automotive and aircraft industry, nuclear power plants, process control, and robotics. Such critical systems claim high dependability requirements; because even a small temporal fault occurred in the embedded computer system may raise failures leading to unacceptable catastrophic situations. Therefore, it is not sufficient that such systems meet only hard real-time constraints, they must also guarantee to meet specified safety constraints despite of physical fault occurrences [1]. Since these safety critical systems include many distributed computer nodes, they must have the ability to ensure a consistent global state over all the system components.

Recently, has emerged a tendency to increase vehicles safety by introducing intelligent control systems like brake-by-wire and steer-by-wire. Such safety critical systems so-called x-by-wire may be implemented in a Time Triggered Architecture (TTA) [2]. At the core of TTA is the Time Triggered Protocol of Communication (TTP/C) for hard real-time fault-tolerant distributed systems. In TTP/C, fault tolerance abilities are implemented in two ways in both hardware and software components. Whereas the hardware relies on redundant nodes and duplicated communication channels, the software uses algorithms that control such basic services as membership agreement, clique avoidance [3], and clock synchronization. The provision of fault tolerance is based on fault hypothesis, so this paper is devoted to cope with transient faults caused by ambient cosmic radiation, which is the most important cause of transient failures in embedded systems. These kinds of faults [17] are reported to occur more frequently than permanent faults. Due to the shrinking size and reduced supply voltage of embedded systems, the transient fault rate is predicted to increase dramatically in the near future. Thus, due to the critical feature of TTP/C services, it is of great importance to get the maximum reliability of TTP/C. Therefore, a formal analysis of its behavior through its membership protocol is required. In order to increase the survivability of critical embedded systems, particularly when the number of nodes in cluster is limited, we consider that node reintegration must be the indivisible part of an overall GMP and the formal analysis must be globally performed. Therefore, in this paper, we focus on the formal verification of a node reintegration inside TTP/C's GMP, which complements the previous work in [5] and extends the work in [7].

The paper is organized as follows: Sections 2 and 3 present respectively some related work, our system model and assumption. Section 4 presents informal and formal description of our proposed algorithm. In section 5, we develop the configuration diagram used in the verification part. Verification of the protocol is presented in section 6. Section 7 shows the verification results. Finally, section 8 concludes the paper.

2. Related Work

The evaluation of the reliability of a GMP is very important because group membership services are often used as building blocks in the design of fault tolerant applications. Hence, latest GMPs are always introduced together with a careful analysis of their correct functioning. The correctness of the membership algorithms, introduced by Cristian [7] for the synchronous system model, are described informally. The proofs of other algorithms, including those of Cristian and Schmuck [8] for the timed-asynchronous model and in particular those for the asynchronous system model [10, 11], are presented in a rigorous formal way. Indeed, it has been argued [12] that some of algorithm specifications proposed in the literature have flaws or suffer from other deficiencies. There are some related works in the context of GMPs. For example, Ramasamy *et al.* [14] describe a membership protocol that is part of an intrusion-tolerant group communication system. They specify the protocol in the Promela language and use the Spin model checker to verify the correctness of their protocol. Pascoe *et al.* [12], also use Promela and Spin to build an "agreement problem protocol verification environment" in the approve project. Recently Bouajjani and Merceron propose an automatic verification method based on model checking [5] to verify TTP/C's clique avoidance mechanism for an arbitrary number of processors. The protocol is abstracted in terms of unbounded counter automata and the clique avoidance property is checked by using the symbolic reachability analysis tool Alv. However, what the protocol has addressed is only cliques avoidance in GMP part, without considering node reintegration. More recently, in [14] the agreement is maintained by exchanging a configurable number of acknowledgments for each node's message. The protocol was formally verified by using Spin model checker. Closely related to our research, the protocol in [15] deals with permanent failures. Our protocol, in contrast, handles transient failures. Furthermore, the protocol in [15] requires the membership state to be periodically broadcasted to support node reintegration. In our approach, nodes recover the membership state by listening on the network. The membership protocol proposed in this paper is based on the principal that the resilience to transient failures should be adjustable to the development of embedded systems. Our previous work [5] only gives a formal specification. This paper focuses on the verification of the protocol and the properties which must be satisfied by the protocol.

3. System Model and Assumptions

We consider a distributed system composed of a set of nodes interconnected by a TDMA based real time communication subsystem. Nodes have their clocks tightly synchronized, and send messages in their pre-allocated transmission slots according to a time triggered cyclic schedule. We assume that faults occur in the interface to the network of the node, caused by radiation-induced high-energy ions or external electric disturbances. These faults lead to send/receive omission failures. According to the papers [4, 16], radiation-induced, high-energy ions are the most important cause of transient failures in future advanced computer systems. We assume that the fault occurrences are sufficiently rare [15] to guarantee that when a processor fails, it flows out an interval of time at most 2n slots before another processor becomes faulty. And after being detected faulty, it will reestablish its view during the following slots. Therefore, it has to listen at most 2n slots before being able to send a message. It depends on the type of the fault that is, sending or receiving failure.

- *Transient Sending Failure:* A failure of sending node means missing one message from this node during its slot. This message is either not received at all by the nodes or received wrong. The failure will be detected by other nodes and hence considered as faulty and excluded from the views of all nodes.
- *Transient Receiving Failure:* One node fails to receive a message correctly. This node will be detected faulty in its own slot.

4. Group Membership Protocol

In a distributed system, an adherence protocol of group membership is a fault tolerant mechanism, capable of obtaining a consensus on the identities of non failed correct processors [6]. Any failed processor must be excluded from the group at the end of limited time and will be reintegrated into the group in the following slot after the fault detection.

4.1. Informal Description of the GMP

We model a distributed computation as a finite set of processors (or nodes) labeled by 0, 1, …, n-1 and arranged in a logical ring. Every processor *p* maintains a set mem^t_p (the Membership Set (MS) of processor *p* at time *t*) containing all operational processors that *p* considers at time *t*. The broadcast is realized on the basis of TDMA strategy. In slot *t* the node with label *t* mod *n* is the broadcaster, denoted by broadcaster (*t*). The message being sent contains the broadcaster's local view mem^t_b on the membership. A broadcaster *p* is detected faulty if the view's successor (slot t+1) or next successor (slot t+2) broadcaster does not contain *p*. If a processor *p* was the previous broadcaster (in slot t) and waits for being acknowledged, the boolean state variable, $prev^t_p$ is set to true. When the view's sender in slot *t+1* doesn't contain *p,* the node *p* sets the boolean state variable, $doubt^t_p$ *to true* and waits for slot *t+2* to check whether it is faulty. In this case, the variable $succ^t_p$ holds *p*'s first successor which refused to acknowledge *p*. If the view's second successor does not contain *p*, *p* will remove itself from its own MS and fail silently. A similar mechanism could be used for diagnosing receive faults. However, each processor *p* maintains two counters, acc^t_p and rej^t_p, which count the number of messages that *p* has successfully received and rejected, respectively. A processor *p* will increase the counter acc^t_p if it agrees with the broadcaster's view. In the next round *p* checks whether it has accepted more messages in the last one than it has rejected. If so, *p* resets its counters and broadcasts; otherwise *p* removes itself from its MS without broadcasting.

4.2. Formal Analysis of the GMP

Our formal model is described by a set of guarded commands. In every slot *t*, every processor executes exactly one of these commands. The guards are evaluated in a top-down order. Let NF^t be a set of non-faulty processors at time *t*, $sends^t_b$ denotes that the current broadcaster *b* sends a message; $arrives^t_p$ indicates that a message arrives at the receiver *p*, and $integrat^t_p$ means that a processor *p* will integrate to the group. The PVS specification is given as follows:

NF^t: set[proc]; $Sends^t_b$:bool; $Arrives^t_p$:bool; $Integrat^t_p$:bool
Sending: Axiom Let b=broadcaster(t) In $b \in mem^t_b \Rightarrow sends^t_b$
A message sent by the broadcaster b will arrive at a processor p ($p \in NF^t \cup$ integrator). These axioms also imply that broadcasts are consistent:
arrival: Axiom Let b=broadcaster(t) In $sends^t_b \wedge p \in NF^t \Rightarrow arrives^t_p$
arrival_int: Axiom Let b=broadcaster(t) In $sends^t_b \wedge integrat^t_p \Rightarrow arrives^t_p$
nonarrival: Axiom Let b=broadcaster(t) In $\neg sends^t_b \Rightarrow \neg arrives^t_p$
The processor that has been detected failed and has emptied its MS will be integrated:
Integrating: Axiom Let b=broadcaster(t) In mem^t_p=empty $\wedge \neg integrat^t_p \Rightarrow integrat^{t+1}_p$

4.3. Group Membership Algorithm

The algorithm is specified by a set of guarded commands. We explain only those relating to the node reintegration. The clause 3 describes the behavior of a processor that has already emptied its MS, of course a processor detected faulty. Such a processor will be reintegrated to the group but not immediately. It resets the integrat flag to true and the counters acc and rej to 2 and 0, respectively. Its MS will then contain only itself and the current broadcaster. The clause 4 describes the situation of an integrator receiving a correct message and the current broadcaster has accepted the previous broadcaster's message. Therefore, the processor finish the reintegration process, resets the *prev* and *integrat* flags to false and increases the acc counter. The clause 8 describes the behavior when the current broadcaster is integrator and the receiver is a previous broadcaster and has correctly received a message. The clause 17 is evaluated to true when the processor receives a message and agrees with the integrator broadcaster's view on the membership.

Broadcaster

1. $acc_p^t > rej_p^t \wedge acc_p^t \geq 2 \rightarrow mem_p^{t+1} = mem_p^t \wedge prev_p^{t+1} = T \wedge acc_p^{t+1} = 1 \wedge rej_p^{t+1} = 0;$
2. $otherwise \rightarrow mem_p^{t+1} = emptyset \wedge prev_p^{t+1} = F \wedge acc_p^{t+1} = 0 \wedge rej_p^{t+1} = 0;$

Receiver

3. $mem_p^t = emptyset \rightarrow integrat_p^{t+1} = T \wedge mem_p^{t+1} = \{p,b\} \wedge acc_p^{t+1} = 2 \wedge rej_p^{t+1} = 0;$
4. $prev_p^t \wedge arrive_p^t \wedge mem_b^t = mem_p^t \cup \{p\} \wedge integrat_p^t \rightarrow mem_p^{t+1} = mem_p^t \wedge prev_p^{t+1} = F \wedge acc_p^{t+1} = acc_p^t + 1 \wedge integrat_p^{t+1} = F;$
5. $prev_p^t \wedge arrive_p^t \wedge integrat_p^t \rightarrow mem_p^{t+1} = mem_p^t \cup \{b\} \wedge prev_p^{t+1} = F \wedge acc_p^{t+1} = acc_p^t + 1;$
6. $prev_p^t \wedge arrive_p^t \wedge mem_b^t = mem_p^t \cup \{p\} \rightarrow mem_p^{t+1} = mem_p^t \wedge prev_p^{t+1} = F \wedge acc_p^{t+1} = acc_p^t + 1;$
7. $prev_p^t \wedge arrive_p^t \wedge mem_b^t = mem_p^t \setminus \{p\} \rightarrow mem_p^{t+1} = mem_p^t \setminus \{b\} \wedge prev_p^{t+1} = F \wedge doubt_p^{t+1} = T \wedge rej_p^{t+1} = rej_p^t + 1 \wedge succ_p^{t+1} = b;$
8. $prev_p^t \wedge arrive_p^t \wedge p \in mem_b^t \wedge b \notin mem_p^t \rightarrow mem_p^{t+1} = mem_p^t \cup \{b\} \wedge acc_p^{t+1} = acc_p^t + 1 \wedge prev_p^{t+1} = F;$
9. $arrive_p^t \wedge p \notin mem_b^t \wedge b \notin mem_p^t \rightarrow mem_p^{t+1} = mem_p^t \cup \{b\} \wedge acc_p^{t+1} = acc_p^t + 1;$
10. $prev_p^t \wedge null_p^t \rightarrow mem_p^{t+1} = mem_p^t \setminus \{b\};$
11. $doubt_p^t \wedge arrive_p^t \wedge mem_b^t = mem_p^t \cup \{p\} \setminus \{succ_p^t\} \rightarrow mem_p^{t+1} = mem_p^t \wedge acc_p^{t+1} = acc_p^t + 1 \wedge doubt_p^{t+1} = F;$
12. $doubt_p^t \wedge arrive_p^t \wedge mem_b^t = mem_p^t \cup \{succ_p^t, b\} \setminus \{p\} \rightarrow mem_p^{t+1} = emptyset \wedge doubt_p^{t+1} = F \wedge acc_p^{t+1} = acc_p^t + 1;$
13. $doubt_p^t \wedge null_p^t \rightarrow mem_p^{t+1} = mem_p^t \setminus \{b\};$
14. $doubt_p^t \rightarrow mem_p^{t+1} = mem_p^t \setminus \{b\} \wedge rej_p^{t+1} = rej_p^t + 1;$
15. $arrive_p^t \wedge integrat_p^t \wedge (mem_p^t = mem_b^t) \rightarrow mem_p^{t+1} = mem_p^t \wedge acc_p^{t+1} = acc_p^t + 1 \wedge integrat_p^{t+1} = F;$
16. $arrive_p^t \wedge integrat_p^t \rightarrow mem_p^{t+1} = mem_p^t \cup \{b\} \wedge acc_p^{t+1} = acc_p^t + 1;$
17. $arrive_p^t \wedge p \notin mem_b^t \wedge b \notin mem_p^t \rightarrow mem_p^{t+1} = mem_p^t \cup \{b\} \wedge acc_p^{t+1} = acc_p^t + 1;$
18. $arrive_p^t \wedge (mem_p^t = mem_b^t) \rightarrow mem_p^{t+1} = mem_p^t \wedge acc_p^{t+1} = acc_p^t + 1;$
19. $null_p^t \rightarrow mem_p^{t+1} = mem_p^t \setminus \{b\};$
20. $otherwise \rightarrow mem_p^{t+1} = mem_p^t \setminus \{b\} \wedge rej_p^{t+1} = rej_p^t + 1;$

4.4. Initial State

We suppose that all the processors are initially not faulty. Their MSs contain all the processors and so the properties of validity and agreement are verified in the initial state. For others state variables, the mechanism of clique avoidance which is integrated into the algorithm imposes constraints on the values of acc and rej of the processor which is going to broadcast in the following slot and have to be equal to 2 and 0 respectively. As in the GMP, the last broadcaster is distinguished from the other receivers, because it waits for its acknowledgment by its successor, and hence its variable prev is set true. We suppose that the previous broadcaster is the processor labeled (n-1); the counters acc and rej are set to 1 and 0 respectively (no other message has been accepted except its own). For the doubt and integrat variables, we assume that no processor is in doubt or is integrator respectively.

- *Definition 1:* 1. $\forall$ p: $mem_p^0 = \{p/T\}$; 2. $\forall$ p: $(p \neq n-1) \Rightarrow acc_p^0 = 2$; 3. $acc_{n-1}^0 = 1$; 4. $\forall$ p: $rej_p^0 = 0$ 5. $\forall$ p: $prev_p^0 \Leftrightarrow p=(n-1)$; 6. $\forall$ p: $doubt_p^0 = F$; 7. $\forall$ p: $integrat_p^0 = F$

5. Developing the Configuration Diagram

The diagram for the GMP is shown in Figure 1. The nodes of the diagram represent the configurations, and arrows denote transitions from one configuration to others and are labeled with transition conditions. Configurations are parameterized by the time *t* and describe the global state the system is in. Configurations can have additional parameters such as processors (*x*, *y*, .) that behave differently from the rest of system, or additional entities necessary to describe the system state. The labels of transitions express the preconditions for the system to move from one configuration to another. The system is said to be in a stable configuration if the MS of all non-faulty processors *p* is equal to NF^t, and a faulty processor has already diagnosed its fault and thus removed itself from its own MS.

- *Definition 2:* stable (t, z): bool = recent (t, z) $\wedge$ ($\forall$p: $p \in NF^t \Rightarrow mem_p^t = NF^t$) $\wedge$ ($p = z \Leftrightarrow acc_p^t > rej_p^t$) $\wedge$ ($p \neq z \Rightarrow acc_p^t > rej_p^t+1$) $\wedge$ ($prev_p^t = T \Leftrightarrow p = z$) $\wedge$ $doubt_p^t = F \wedge integrat_p^t = F$

In the configuration stable (*t, z*) the counters of non-faulty processors are set such that $acc_p^t > rej_p^t+1$. This is to allow for a non-faulty processor *p* to cope with a send fault of other broadcaster in the next round, in which case, the counter rej_p^t will be increased; this should not lead to *p* removing itself from its own MS in its next sending slot, for which $acc_p^t > rej_p^t$ must hold. However, the most recent non-faulty broadcaster, say *z*, cannot satisfy this condition as in its sending slot *z,* sets the counters: $acc_z^t=1$ *and* $rej_z^t=0$.

The expression recent *(t, z)* denotes that at time *t*, processor *z* is the recent non-faulty broadcaster.

- *Assumption 1:* At the start time of the algorithm all processors are non-faulty: $\forall p: p \in NF^0$.
- *Lemma 1:* There is a processor *z* such that the GMP is in the *stable* configuration at time 0 with respect to *z*: $\exists z$*: stable (0,z).*
- *Proof:* Let *z* the processor labeled n-1; the conditions for the stable configuration follow immediately from definition 1 and assumption 1.

In order to determine the transition conditions from the stable configuration, we consider whether or not a new fault occurs in the next step. For the first, if no processor becomes faulty, that is $NF^{t+1}=NF^t$ holds, the system remains in stable, because neither the broadcaster nor the receivers change their MSs: the broadcaster will execute command 1, while the receivers will execute command 18, except for *z* which will execute command 6. The broadcaster *b* now becomes the most recent non-faulty broadcaster and the new configuration is stable (*t+1, b*).

- *Lemma 2:* Let the system be in the stable configuration at time *t* with respect to *z* and let *b* denotes the broadcaster at time *t*. If *b* is already faulty and no new fault occurs in the next step then the system will be in the stable configuration at time *t +1* with respect to *z*: *stable (t, z)* $\wedge b \in NF^t \wedge NF^{t+1}=NF^t \Rightarrow$ *stable (t+1, b).*

Now we consider the case where a new fault occurs. If a processor, say *x*, which was non-faulty at time *t* becomes faulty at time *t+1*, the same commands as above will be executed. However, the system will not be in a stable configuration any more, because now the MSs of both non-faulty processors and *x* do not only contain non-faulty processors, but also the newly faulty processor *x*. Hence we introduce a new configuration that we call latent.

- *Definition 3:* Latent (t, x, z): bool=$x \in NF^{t-1} \wedge NF^{t-1} = NF^t/\{x\} \wedge$ recent (t, x, z) $\wedge \forall$p: ($p \in NF^t \vee p=x$) $\Rightarrow mem_p^t = NF^t \cup \{x\}$ $\wedge$ ($p = z \Leftrightarrow acc_p^t = rej_p^t+1$) $\wedge$ ($p \neq z \Rightarrow acc_p^t > rej_p^t+1$) $\wedge$ ($prev_p^t = T \Leftrightarrow p = z$) $\wedge$ $doubt_p^t = F \wedge integrat_p^t = F$
- *Lemma 3:* Let the system be in the stable configuration at time *t* with respect to *z* and let *b* denote the broadcaster at time *t*. If the processor *x* becomes faulty in the next step then the system will be in the configuration latent at time *t+1* with respect to *x* and *b*: *stable (t, z)* $\wedge NF^{t+1} = NF^t \setminus \{x\} \Rightarrow$ *latent (t+1, x, b).*

It is the duty of the GMP to assure that possibly all the processors, including *x*, know about the fault of *x* and eliminate it from their MSs and *x* will reintegrate in the group and so system returns in the stable configuration. The hypothesis of fault presumes that no new fault will occur during this time until system will become again stable. From the stable configuration to reintegration configuration, the processors detect the faulty processor *x* and eliminate it from their MSs. After that, faulty processor *x* will reintegrate in the group and construct its MS and so the system returns in the stable configuration. The GMP is said to be in the reintegration configuration at time t, for t>0, if the MS of all non faulty processors is equal to NF^t and the MS's faulty processor *x* is empty. The system transits into the reintegration configuration if *x* is the current broadcaster but fails to send a message command 2. Thus, the non-faulty processors remove *x* from their MSs by executing the command 19 or 9 in the case of *z*. Therefore, all non-faulty processors have the same MSs.

- *Definition 4:* reintegration (t, x, z): bool= $\exists$ i: $0 < i \leq n \Rightarrow$ sends(t-i)= x $\wedge$ $\forall$ p : (p $\in NF^t$ $\vee$ p= x) $\Rightarrow$ ($mem^t_p = \varnothing \Leftrightarrow$ p= x) $\wedge$ ($mem^t_p = NF^t \Leftrightarrow p \neq x$) $\wedge$ ($acc^t_p = rej^t_p +1 \Leftrightarrow$ (p= z $\vee$ p= x)) $\wedge$ ($integrat^t_p$= T $\Leftrightarrow$ p= x) $\wedge$ ($prev^t_p$= T $\Leftrightarrow$ p= z) $\wedge$ $doubt^t_p$= F

If a processor, say x, which was detected faulty and has its MS empty becomes integrator, the system will be into a new configuration, that we call reintegration-member.

- *Definition 5:* reintegration-member (t, x, z, R): bool= $\exists$ x, z: $x \notin NF^t \wedge z \in NF^t \wedge (\exists\, i : 0 < i \leq n \Rightarrow$ sends(t-i)= x) $\wedge$ ($\exists$ p : p$\in NF^t$ $\wedge$ before (t, p, z)) $\wedge$ $\forall$ p : (p $\in NF^t$ $\vee$ p= x) $\Rightarrow$ ($mem^t_p = R \Leftrightarrow$ p= x) $\wedge$ ($acc^t_p > rej^t_p +1 \Leftrightarrow$ p= x) $\wedge$ ($acc^t_p = rej^t_p +1 \Leftrightarrow$ p= z) $\wedge$ ($integrat^t_p$= T $\Leftrightarrow$ p)= x $\wedge$ ($prev^t_p$= T $\Leftrightarrow$ p= z) $\wedge$ $doubt^t_p$= F
- *Lemma 4:* Let the system be in the reintegration configuration at time t with respect to x and z, and let b denoted the broadcaster at time t. If MS's x is empty, and if no new fault occurs in the next step then the system will be in the reintegration-member configuration at time t+1 with respect to x, b, and the set {x, b}.

reintegration(t,x,z) $\wedge$ *b* $\in NF^t \wedge NF^t = NF^{t+1} \Rightarrow$
reintegration-member(t+1,x,b,{x, b})

By systematically analyzing the possible cases for a given configuration one proceeds to develop the configuration diagram. Every transition either leads to a new configuration or to an already existing one. In some cases it may be necessary to generalize an existing configuration in order to establish the proof of a transition. The ultimate goal in this process is to end up with a configuration diagram which is closed. After analyzing of the behavior of the GMP and constructing the configuration diagram that describes all reachable states of all processors in the system, we explain how this diagram, as represented by the various lemmas described so far, can be used to accomplish the proofs of the three correctness properties: validity, agreement and liveness.

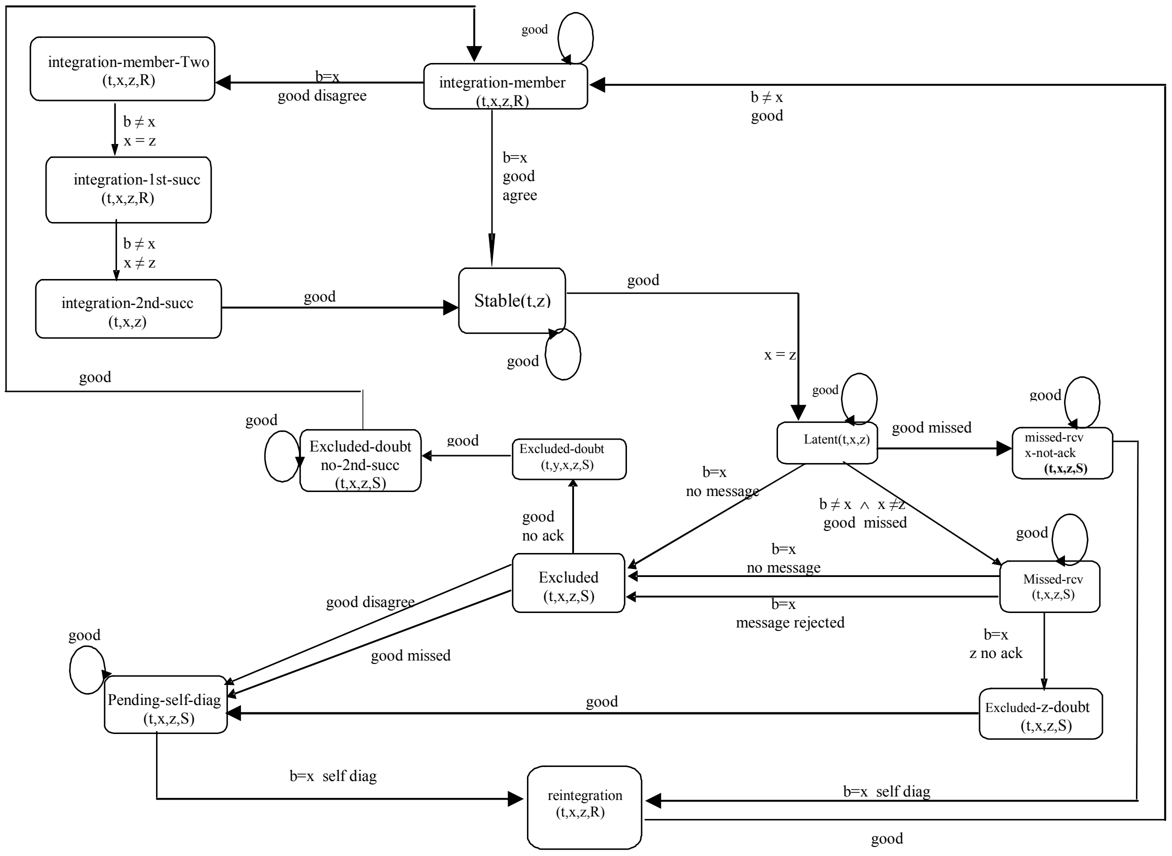

Figure 1. Configuration diagram for the global TTP membership algorithm.

6. Verification of the Protocol

6.1. Proving Safety Properties

The purpose to verifying the GMP is to prove that the safety properties (validity and agreement) hold in every step of the system. The configurations of the diagram are defined such that the safety properties hold for every configuration. For example, for the reintegration configuration, it is simple to prove the following lemmas and similar ones for every other configuration of the diagram.

- *Lemma 5:* If the GMP is in the configuration reintegration at time *t* with respect to processors *x, z* and a set *R* then the validity property holds at time *t.*
- *Proof:* From the definition 4, the MS of every non-faulty processor is equal to the set of all non-faulty

processors and hence the first constraint of the validity property is satisfied. The faulty processors x has emptied its MS and therefore satisfy the second validity constraint.

- *Lemma 6:* If the GMP is in the reintegration configuration at time t with respect to x and z then the agreement property holds at time t.
- *Proof:* From the definition 4, the MS of every non-faulty processor is equal to the set of all non-faulty processors. Therefore, all non-faulty processors have the same MSs and hence agreement holds.

As the safety properties hold in every configuration, the overall proof can be accomplished if we can show that in every step the GMP is in one of the configurations of the diagram. In other words, we must prove that the diagram is complete in the sense that there are no other transitions than those proved for the diagram. To this end, we prove for every configuration a lemma which states all possible configurations successor the algorithm can move to in the next step. Consider, for example, the stable configuration, as long as no new fault occurs, the system will remain in this configuration. If, however, a new fault occurs, then the system will move to the latent configuration. The following lemma states that these are all possible transitions.

- *Lemma 7:* Let the system be in the stable configuration at time t with respect to some processor z. Then, in the next step at time $t + 1$, the algorithm will be either still in *stable* with respect to the broadcaster b at time t, or there exists a processor x such that the algorithm is in the *latent* configuration at time $t + 1$ with respect to x and b.

  $stable\ (t, z) \Rightarrow stable\ (t+1, b) \vee \exists x\text{: } latent(t+1, b)$
- *Proof:* Follows directly from the Lemmas 2 and 3. Similar lemmas can be proved for all other configuration as well, however, there is one difference to the proof of the lemma above. The preconditions of all transitions for configurations other than *stable* contain the conjunct $NF^{t+1} = NF^t$ and thus require that the set of non-faulty processors does not change, i.e., that no new fault occurs. The GMP is able to tolerate faults provided that they do not occur too frequently. More specifically, as long as a newly faulty processor x has not yet been detected faulty and reintegrated in the group, a new fault must not occur. We use a slightly different form of this assumption and assume that faults only arrive when the system is in the *stable* configuration.
- *Assumption 2:* If the GMP is in the configuration *stable* at time t with respect to some processor z, then the set of non-faulty processors is decreased by at most one processor x in the next step.

  $stable\ (t, z) \Rightarrow (NF^{t+1} = NF^t) \vee \exists x \in NF^t : NF^{t+1} = NF^t \setminus \{x\}$
- *Assumption 3:* If the GMP is *not* in the *stable* configuration at time t with respect to some processor z, then the set of non-faulty processors does not change in the next step.

  $(\forall z : \neg stable(t, z)) \Rightarrow NF^{t+1} = NF^t$

The proof for the lemmas about the successors of configurations, however, can still not be completed, since we must prove that all configurations (except stable) are in fact different from stable and hence new faults can be assumed not to occur in these configurations. For the configuration reintegration, for example, we must prove the following lemma.

- *Lemma 8:* If the GMP is in the reintegration configuration at time t with respect to processors x and z and the set R, then the algorithm is not in the stable configuration at time t for any processor p.

  $reintegration\ (t, x, z, R) \Rightarrow \forall p : \neg stable(t, p)$
- *Proof*: By the definition 4, we have one faulty processor x which emptied its MS whereas stable configuration denotes that all processors are non-faulty and their MSs are equal to NF^t; so this violates the definition 2. Similar lemmas are proved in the same way for all other configurations. This allows stating and proving a following lemma about the successor configurations of reintegration.
- *Lemma 9:* Suppose that the GMP is in the reintegration configuration at time t with respect to some processors x and z then, in the next step at time t+1, the algorithm will be in the reintegration-member configuration with respect to x, z, the broadcaster b and the set $R= \{x, b\}$.

  $Reintegration\ (t, x, z) \Rightarrow Reintegration\text{-}member\ (t+1, x, b, \{x, b\})$
- *Proof:* Follows directly from the lemma 4. Now all prerequisites are fulfilled to prove that the configuration diagram completely describes the behavior of the GMP under the assumed fault hypotheses. The following definition formally describes this property and expresses that at all times the system is in one of the stated configurations.
- *Definition 6:* We write total (t) if the GMP is in one of the following configurations:
  stable, latent, missed-rcv-x-not-ack, missed-rcv, excluded, excluded-z-doubt, excluded-doubt, excluded-doubt-no-2nd-succ, pending-self-diag, reintegration, reintegration-member, reintegration-member-two, reintegration-1st-succ, reintegration-2nd-succ with respect to appropriate values of processors and sets.
- *Lemma 10:* At all times t, the GMP is in one of the stated configurations: $\forall$t: total (t).
- *Proof:* By induction on t, for $t = 0$ from Lemma 1 the GMP is in the stable configuration with respect to the processor labeled n - 1. For $t \rightarrow t + 1$, suppose total (t) holds; we can split on the actual configuration the GMP is in. Suppose the algorithm is in the stable configuration for some processor z, then by Lemma 7, the algorithm will be in stable or

latent at time t +1 and hence total (t+1) holds. The same argument holds for reintegration by Lemma 9, and analogously for all other configurations. Now the safety properties of validity and agreement are easily proved using the totality proposition above.

- *Theorem 1:* At all times, the GMP preserves a valid view on the membership status of the system.
- *Proof:* By Lemma 10, GMP is in one of the configurations at all times t. Suppose the algorithm is in reintegration configuration, validity follows from Lemma 5. Hence, for all other configurations similar arguments hold.
- *Theorem 2:* At all times, the GMP establishes agreement among the non-faulty processors.
- *Proof:* Analogously to the proof of the previous theorem, using lemmas similar to Lemma 6.

6.2. Proving Liveness Properties

The third correctness property that the GMP must meet is Self diagnosis-reintegration. The maximum number of slots executed by the algorithm before a faulty processor diagnoses its fault and reintegrates into the group is bounded by $3n-1$. All configurations other than stable and reintegration allow for a faulty processor to be contained in its own MS. We therefore prove the stronger liveness property which states: The system remains outside the stable configuration for at most $3n-1$ slots. Once the system leaves the *stable* configuration, it takes at most $2n-1$ slots for coming to reintegration configuration detection fault phase and at most n slots to return to stable configuration reintegration phase.

- *Definition 7:* The GMP is said to satisfy the liveness property at time t, if the algorithm is not in the stable configuration for at most $3n-1$ slots:
  $\forall z : stable(t, z) \Rightarrow \exists\ z , s : 0 < s \ \wedge\ s \leq 3n\text{-}1 \wedge$
  $stable(t+s, z)$

For this property, we must show that once the system leaves the stable configuration it will not remain in other configuration for more than 3n-1 slots. The proof will be split into two parts: first, we demonstrate that every configuration will be eventually left, that is, the system does not loop forever on one of the configurations, and second, bound the number of steps it takes for the system to return to stable from any given configuration. The proof of the first requirement that there are no infinite loops in any configuration other than stable is similar for every configuration and will be accomplished in two steps. To illustrate these we consider, for example, the reintegration-member configuration. From the configuration diagram shown in Figure 1 we know that in reintegration-member the system either leaves this configuration in the next step to reintegration-member-two or stable, or remains in reintegration-member. We realize that the system can only loop on reintegration-member if the current broadcast is not the faulty processor x. Hence we can deduce that either the system leaves the reintegration-member configuration after some s steps, or for all further slots the respective broadcasters are not x.

- *Lemma 11:* Suppose the GMP is in the configuration reintegration-member at time t with respect to some processors x and z and a set R, then there either exists a number of slots s such that the algorithm makes a transition to reintegration-member-two or stable after s slots, or the current broadcaster (i.e., b= sender (t)) is not x for all next slots:
  $reintegration\text{-}member^{t}(z,x,R) \Rightarrow \exists s, z, b, R:$
  $reintegration\text{-}member\text{-}two^{t+s}(z,x,b,R) \vee \exists s, z, b:$
  $stable^{t+s}(b) \vee \forall s: sender(t+s) \neq x.$

The above lemma can be reached as the number of non-faulty processors is n-1. This implies that the integrator processor x will eventually become the next broadcaster. Thus, the system will leave the reintegration-member configuration and we can prove the liveness property.

- *Lemma 12:* Suppose the GMP is in the configuration reintegration-member at time t with respect to some processors x and z and a set R, then there exists either a number of slots s such that the algorithm makes a transition to reintegration-member-two or stable after s slots.
  $reintegration\text{-}member^{t}(z,x,R) \Rightarrow \exists s, z, b, R:$
  $reintegration\text{-}member\text{-}two^{t+s}(z,x,b,R) \vee \exists s, z, b:$
  $stable^{t+s}(b)$

For all configurations, corresponding lemmas can be proved in the same way that the system leaves on the broadcast of x. The second part of the proof of the liveness property is concerned with establishing the bound on the number of steps for the system to return to the stable configuration. This is accomplished by analyzing the length of all possible paths through the configuration diagram. We proceed backwards and first consider all configurations from which the system can only make a transition to stable, such as reintegration-member and reintegration-2nd-succ. For this, we have defined on every configuration a parameter i that either counts the number of slots since the last broadcasting of the faulty processor x for excluded configuration and all reachable from there, or is a lower bound of the number of slots since the last broadcast of x that can be at most n for other configurations. For the latter type of configurations, such as reintegration-member, if the algorithm is in one of these configurations with a counter value of i then after at most n-i steps a transition will be made to exit the configuration.

- *Lemma 13:* let us suppose the algorithm is in the reintegration-member configuration at time t with respect to x, z and R with counter value of i according to the definition of the configuration; then after at most n-i slots the algorithm returns to the stable configuration:

reintegration-member $(t, z, x, R) \Rightarrow \exists s, z: s \leq n - i \wedge$ *stable* $(t+s, b)$

The value of *i* is increased by one on every transition.

- *Lemma 14:* Suppose the algorithm is in the reintegration configuration at time *t* with respect to *x*, *z*, a set *R* and counter value of *i* according to the definition of the configuration; then after at most *n-i* slots the algorithm returns to the stable configuration:

  reintegration $(t, z, x, R) \Rightarrow \exists s, z: s \leq n - i \wedge$ *stable* $(t+s, b)$
- *Lemma 15:* Suppose the algorithm is the in missed-rcv-x-not-ack configuration at time *t* with respect to *x*, *z*, *S* and *i* denotes the counter value according to the definition of the configuration; then after at most *n* - *i* slots the algorithm makes a transition to the stable configuration:

  missed-rcv-x-not-ack $(t, x, z, S) \Rightarrow \exists s; z: s \leq n - i \wedge$ *stable* $(t+s, z)$

The final lemma that has to be proved is the one for the latent configuration. If the system leaves this configuration to missed-rcv-x-not-ack then it will return to the stable configuration within at most *n* steps. On the path via missed-rcv it will take at most 3*n* -*2* steps, as the counter *i* of missed_rcv set to 1 initially.

- *Lemma 16:* Suppose the algorithm is the in latent configuration at time *t* with respect to *x*; then after at most 3*n* -2 slots the algorithm returns to the *stable* configuration:

  latent $(t, x) \Rightarrow \exists s; z: s \leq 3n - 2 \wedge$ *stable*$(t+s, z)$
- *Theorem 3:* At all times *t*, the liveness property holds for the GMP.
- *Proof:* Follows from lemma 16. If a new fault occurs then the system will make a transition to latent, which adds 1 to the bound established for this latter configuration.

The original goal was to establish the self-diagnosis-reintegration property for the GMP which follows directly from the theorem 3: after a processor *x* becomes faulty, the system will return to the stable configuration, in which *x* has been detected faulty and has reintegrated into the group, after at most 3*n-1* slots.

7. Verification Results

The obtained results are relative to the verification of self-diagnosis-reintegration property for a model of seven, eight, nine and ten nodes. We have stated that the self-diagnosis-reintegration phase is split into two parts: detection phase and reintegration phase. Thus, the Table 1 shows the number of slots taken in each phase. According to the rank (in the ring) of the faulty processor, the minimum (min) and the maximum (max) number of slots taken in the detection phase are:

- In case of sending fault: If a faulty processor is the first one in the ring, the duration of the detection phase ddp is minimal. Contrary if it is the last one, the ddp is maximal;
- In case of receiving fault: The ddp is minimal when the faulty processor is the last one and is maximal if the faulty processor is the first one.

Beyond these extreme cases, the ddp is as follows:

$$4 \lneq ddp \lneq 2n$$

The duration of the reintegration phase drp is equal to the number of slots in the round (number of processors). These results are summarized in the following Table:

Table1. Detection bound and reintegration duration.

No. Nodes	ddp		drp
	Min	Max	
7	4	13	7
8	4	15	8
9	4	17	9
10	4	19	10

All the transitions in the diagram shown in Figure 1 have been proved with the assistance of the PVS theorem prover.

8. Conclusions

In this paper, we have addressed the threat dependability of embedded systems caused by ambient cosmic radiation. This phenomenon is the most important cause of transient failures in future advanced embedded systems. Thus, the transient fault rate is predicted to increase dramatically in the future. Hence, the transient fault recovery must be handled considerably. In the previous GMP of TTP/C, any detected faulty node, is immediately excluded from the group. If the node reintegration is not implemented, this continuous exclusion process risks invalidating the protocol after n-3 successive failures. Our contribution in this paper is to solve this serious problem. Therefore, we have proposed a formal framework to model the GMP with node reintegration. This framework allows GMP to get more availability in the context of critical embedded applications. The proofs of the main correctness properties of the algorithm have been constructed and then checked with the PVS.

In the future we intend to generalize the fault model of the protocol. Thus, we will consider that at most a given number k of transient faults may occur during one round.

References

[1] Aliouat Z., "Formal Analysis of Fault-Tolerant Algorithm in the Time-Triggered Architecture," *Journal of Computer Science*, vol. 3, no. 1, pp. 28-34, 2007.

[2] Anceaume E., Charron-Bost B., Minet P., and Toueg S., "On the Formal Specification of Group

Membership Services," *Technical Report*, Cornell University, Unité de Recherche INRIA Rocquencourt, 1995.

[3] Barbosa R. and Johan Karlsson J., "Formal Specification and Verification of A Protocol for Consistent Diagnosis in Real-Time Embedded Systems," *in Proceedings of the 3rd IEEE International Symposium on Industrial Embedded Systems,* France, pp. 216-223, 2008.

[4] Bauman R., "Soft Errors in Advanced Computer Systems," *IEEE Design and Test of Computers*, vol. 22, no. 3, pp. 258-266, 2005.

[5] Bouajjani A. and Merceron A., "Parametric Verification of a Group Membership Algorithm," *Journal of Theory and Practice of Logic Programming Cambridge University*, vol. 6, no. 3, pp. 321-353, 2006.

[6] Constantinescu C., "Impact of Deep Submicron Technology on Dependability of VLSI Circuits," *in Proceedings of The International Conference on Dependable Systems and Networks*, pp. 205-209, 2002.

[7] Cristian F., "Reaching Agreement on Processor-Group Membership in Synchronous Distributed Systems," *Distributed Computing*, vol. 4, no. 4, pp. 175-187, 1991.

[8] Cristian F. and Schmuck F., "Agreeing on Processor Group Membership in Timed Asynchronous Distributed Systems," *Technical Report*, University of California, 1995.

[9] Heiner G. and Thurner T., "Time-Triggered Architecture for Safety-Related Distributed Real-Time Systems in Transportation Systems," *in Proceedings of 28th Annual International Symposium on Fault-Tolerant Computing*, Germany, pp. 402-407, 1998.

[10] Kopetz H. and Bauer G., "The Time-Triggered Architecture," *in Proceedings of the IEEE Special Issue on Modeling and Design of Embedded Software*, pp. 1-14, 2002.

[11] Owre S., Rushby J., Shankar N., and Henke F., "Formal Verification for Fault-Tolerant Architectures: Prolegomena to the Design of PVS," *IEEE Transactions on Software Engineering*, vol. 21, no. 2, pp. 107-125, 1995.

[12] Pascoe J., Loader R., and Sunderam V., *Working Towards the Agreement Problem Protocol Verification Environment*, in Alan C., Majid M., and Henk M., (Eds.), Communication Process Architectures, IOS Press, UK, 2001.

[13] Pfeifer H., "Formal Verification of the TTP Group Membership Algorithm," *in Proceedings of Formal Methods for Distributed System Development*, Tommaso B., and Diego L., (Eds.), Italy, pp. 3-18, 2000.

[14] Ramasamy H., Cukier M., and Sanders W., "Formal Specification and Verification of A GMP for an Intrusion-Tolerant Group Communication System," *in Proceedings of the Pacific Rim International Symposium on Dependable Computing*, USA, pp. 9-18, 2002.

[15] Ricciardi A. and Birman K., "Using Process Groups to Implement Failure Detection in Asynchronous Environments," *in Proceedings of the 10th ACM Symposium on Principles of Distributed Computing*, NY, pp. 341-352, 1991.

[16] Schiper A., "Dynamic Group Communication," *Distributed Computing*, vol. 18, no. 5, pp. 359-374, 2006.

[17] Ziade H., Ayoubi R., and Velazco R., "A Survey on Fault Injection Techniques," *The International Arab Journal for Information Technology*, vol. 1, no. 2, pp. 171-186, 2004.

Zibouda Aliouat obtained her engineer diploma in 1984 and MSc in 1993 from Constantine University. She received her PhD from Setif University of Algeria. She was an assistant at Constantine University from 1985 to 1994. She is currently an assistant professor in Computer Engineering Department at Setif University of Algeria. Her research interests are in the areas of wireless mobile networks modelling and simulation, wireless sensor networks and fault tolerance of embedded systems.

Makhlouf Aliouat received the engineer diploma degree from Computer Science Department of Constantine University, Algeria in 1978, and the MS and PhD degrees from polytechnic national institute of Grenoble, France in 1983 and 1986 respectively. He also received the "enabling to conduct research diploma" from Constantine University in 2008. He was assistant professor from 1986 to 1990 in Computer Science Department of Constantine University. Since 1995, he is associate professor in Computer Science Department of Ferhat Abbès University of Sétif, Algeria. His areas of research are operating systems, distributed systems, wireless mobile computing and wireless network security.