Communication-efficient and crash-quiescent Omega with unknown membership

Sergio Arévalo a, Ernesto Jiménez a,*, Mikel Larrea b, Luis Mengual c

a EUÍ, Universidad Politécnica de Madrid, 28031 Madrid, Spain
b Universidad del País Vasco, 20018 San Sebastián, Spain
c FI, Universidad Politécnica de Madrid, 28660 Boadilla del Monte, Spain

ARTICLE INFO

ABSTRACT

The failure detector class Omega (Ω) provides an eventual leader election functionality, i.e., eventually all correct processes permanently trust the same correct process. An algorithm is communication-efficient if the number of links that carry messages forever is bounded by n, being n the number of processes in the system. It has been defined that an algorithm is crash-quiescent if it eventually stops sending messages to crashed processes. In this regard, it has been recently shown the impossibility of implementing Q crash quiescently without a majority of correct processes. We say that the membership is unknown if each process p¡ only knows its own identity and the number of processes in the system (i.e., i and n), but p¡ does not know the identity of the rest of processes of the system. There is a type of link (denoted by ADD link) in which a bounded (but unknown) number of consecutive messages can be delayed or lost.

In this work we present the first implementation (to our knowledge) of Ω in partially synchronous systems with ADD links and with unknown membership. Furthermore, it is the first implementation of Ω that combines two very interesting properties: communication-efficiency and crash-quiescence when the majority of processes are correct.

Finally, we also obtain with the same algorithm a failure detector (OP) such that every correct process eventually and permanently outputs the set of all correct processes.

1. Introduction

Unreliable failure detectors were proposed by Chandra and Toueg [4] as an elegant way of circumventing the well known FLP impossibility result [6] on solving deterministically the consensus problem [7] in crash-prone asynchronous environments. Informally, an unreliable failure detector provides hints about which processes are correct and which ones are incorrect, i.e., have crashed so far. The failure detector can make mistakes, e.g., by erroneously suspecting a correct but slow process, or by not suspecting (yet) a crashed process. Nevertheless, to be useful it is required that failure detectors fulfill some completeness (i.e., suspect permanently crashed processes) and accuracy (i.e., stop suspecting correct processes) properties. For example, a failure detector such that every correct process eventually and permanently suspects all incorrect processes, and eventually does not suspect any correct process is named Eventually Perfect (♦P).

Another important failure detector is Omega (Ω) [3], defined in terms of a trusted process, such that eventually all correct processes permanently trust the same correct process. It is said that Ω provides an eventual leader election functionality. This type of failure detector is important
because it has been shown that $\Omega$ is the weakest failure detector for solving consensus [3].

We say that an algorithm implementing $\Omega$ is communication-efficient if the number of links that carry messages forever is bounded by $n$, being $n$ the number of processes in the system [2].

A new property called crash-quiescence has been recently proposed [9]. A distributed algorithm is said crash-quiescent if it eventually stops sending messages to crashed processes. Sastry et al. have shown in [9] the impossibility of implementing $\Omega$ (and thus $\Diamond P$) crash-quiescently without a majority of correct processes in a partially synchronous system with links where a bounded, but unknown, number of consecutive messages can be arbitrarily late or lost (called ADD links). They also propose the first implementation of a (non-communication-efficient) $\Diamond P$ algorithm that is crash-quiescent in any run with a majority of correct processes. ADD links [8] are interesting and realistic enough because they allow that an infinite number of messages could be lost or arbitrarily delayed, but guarantee that some subset of the messages sent on them will be received in a timely manner and such messages are not too sparsely distributed in time. In some sense, an ADD link successfully combines the losses property of a fair lossy link [1] (in which if an infinite number of messages are sent, then an infinite number of messages are received), and the timeliness property of an eventually timely link [1] (in which there is a time after which all the messages that are sent are received timely).

A system with unknown membership allows each process to execute the algorithm of $\Omega$ without having to know initially the identifiers of all processes (only knowing its own identifier and the total number of processes in the system is enough). Hence, you can execute the same code of $\Omega$ in runs with different processes without having to statically change the set of addresses of processes that participate in each run. This is possible because processes dynamically learn about the existence of other processes from the reception of messages. Note that this reduction of initial knowledge also supposes that if some process crashes before sending a message, the rest of processes will not be able to find out its existence as member of the system. For example, let us suppose you have a p2p environment in Internet where each server is identified by an IP-address. We know that the number of servers will be, for example, 10. In systems where the membership is known, each process has to know each of the 10 server's IP-addresses in advance to correctly execute the algorithm of $\Omega$ to elect some of them as leader. In systems with unknown membership, the leader can be chosen without this knowledge by learning IP-addresses from the messages received from other processes.

Jiménez et al. show in [5] that it is necessary for each process to know the identity of the rest of processes of the system (that is, the membership) to implement a failure detector of anyone of the original eight classes proposed by Chandra and Toueg in [4] (and, hence, $\Diamond P$). Interestingly, they also show in the same paper that $\Omega$ can be implemented without this knowledge. What it is possible to implement is the complement of $\Diamond P$, that is, a failure detector such that every correct process eventually and permanently only trusts (instead of suspects) all correct (instead of incorrect) processes. We denote this failure detector as $\Diamond \bar{P}$. Note that when the membership is known, $\Diamond \bar{P}$ can easily be transformed into $\Diamond P$ (we can obtain the set of suspected processes simply by removing from the set of all processes the set of trusted processes output by $\Diamond \bar{P}$).

1.1. Our results

In this work we present the first implementation (to our knowledge) of $\Omega$ in partially synchronous systems with ADD links and where the membership is unknown. Furthermore, it is the first implementation that combines two very interesting properties: communication-efficiency and crash-quiescence when the majority of processes are correct. Finally, we also want to emphasize that the implementation presented in this paper also satisfies the properties of $\Diamond \bar{P}$.

2. The model

We consider a system $S$ composed of a finite set $\Pi$ of $n$ processes. The process identifiers are totally ordered, but do not need to be consecutive. We denote them by $p_1, p_2, p_3, \ldots \ldots$. We assume that processes only know about $\Pi$ their own identifier and what is the number $n$ of processes in $S$ (actually, they only need to know the majority, that is, $n/2$). For example, process $p_i$ initially only knows $i$ and $n$.

Processes communicate only by sending and receiving messages. We assume that processes have a broadcasting primitive to send messages (called broadcast($m$)). This primitive allows a process to send a message $m$ to the rest of processes in the system (that is, the same mechanism used in Ethernet or in IP-multicast). Furthermore, processes also have the possibility to use the primitive send ($m$) to $q$ to send the message $m$ to process $q$.

A process can fail by crashing permanently. We say that a process is correct if it does not fail, and we say a process is crashed (or it is a faulty process) if it fails. We use $C$ to denote the set of correct processes, and $F$ to denote the set of faulty processes. We consider that in a system $S$ there may be any number of crashed processes, and that this number is not known a priori. We assume that processes are synchronous in the sense that there are lower and upper bounds on the number of processing steps they execute per unit of time. These bounds are not known by processes. For simplicity, the local processing time is negligible with respect to message communication delays. Processes have timers that accurately measure intervals of time.

We assume that every pair of processes is connected by a pair of directed links. We consider that all links belong to type ADD (Average Delayed/Dropped) [8]. An ADD link allows an infinite number of messages to be lost or arbitrarily delayed, but guarantees that some subset of the messages sent on it will be received in a timely manner and such messages are not too sparsely distributed in time. More precisely, there are two unknown constants $A$ and $B$
The implementation transformed into OP (changing the output of each process with known membership this failure detector can be easily noted that in a system for all process correct, process e correct, always has correct, p e C, after which each process always has correct, p e C, trusts all correct processes. More formally, there is a time after which every correct process permanently only suspects all faulty processes. In [5] it is proven that processes are in the system, process p_i has a set membership_i (initially only contains itself). The set correct_i maintains the processes it believes are alive (always containing at least itself). If a majority of processes are alive, then eventually |correct_i| > n/2 and process p_i will send heartbeats periodically to successor_i, instead of broadcast them (line 08). This variable successor_i contains the process returned by function next_to_i_in_correct_i (line 07). This function obtains the identifier of the process closest to the identifier of p_i in the sequence formed by all elements of correct_i, in increasing order and cyclic (that is, like operation mod but with the elements of correct_i instead of a subset of natural numbers). For example, if process p_3 has correct_3 = {1, 3, 5, 8}, then next_to_i_in_correct_3() will return 5. In another example, if process p_3 has correct_3 = {1, 2, 3}, then next_to_i_in_correct_3() will return 1. As we will show, if |correct_i| > n/2, each correct process p_i will eventually send messages only to one correct process, establishing a cycle (we will define it below as ring) formed by all correct processes. The variable leader_i has the identifier of the process that p_i considers its leader. Its value is the smallest value in correct_i.

5. Correctness proof

Let D be a subset of correct processes (D ⊆ C). We say that there is a relation p_i → p_j, p_i, p_j ∈ D, if there is a time after which process p_i is permanently sending heartbeats to p_j. We say that p_i is the predecessor of p_j, and p_j is the successor of p_i in D. We say there is a ring among all processes in D, denoted by ring(D), if each process p_i ∈ D has a unique predecessor and a unique successor with respect to all processes in D. For example, if D contains the subset of correct processes (p_i, p_k, p_r, p_s), and p_i → p_k → p_i → p_s → p_r, then we say that there is a ring(D).

The following lemma states that eventually crashed processes cannot be in the set correct_i of any correct process p_i.

Lemma 1. For every process p_i ∈ C, there is a time after which if j ∈ correct_i, then p_j ∈ C.
Proof. Let us consider a process $p_j \in F$. We prove that eventually and permanently $j \notin \text{correct}_i$, $\forall p_i \in C$. Let $\text{seq}_j$ be a permutation of the processes in $C$, where the first element of the sequence $\text{seq}_i$ is the process whose identifier $\text{next_to}_i\text{in}_i\text{correct}_i$ would return if $\text{correct}_j = C$. Let $\text{seq}_j(x)$ be, for $x \geq 1$, the $x$th element of the sequence $\text{seq}_j$. The element $\text{seq}_j(x)$, $x \geq 2$, is the process whose identifier $\text{next_to}_i\text{in}_i\text{correct}_i(\text{seq}_j(x-1))$ would return if $\text{correct}_i(\text{seq}_j(x-1)) = C$.

We are going to prove that, for all $x \geq 1$, eventually and permanently $j \notin \text{correct}_i(\text{seq}_j(x))$, by induction on $x$. Base case is $x = 1$. Let us consider a time $\tau$ at which all faulty processes are crashed, and all messages sent by these processes have disappeared from the system (i.e., they have been delivered or lost). After $\tau$, process $\text{seq}_j(1)$ will never receive any heartbeat with a set containing $j$. This is so, because a process sending a heartbeat by execution of line 10, it only includes its own identifier (which cannot be $j$ since process $p_j$ is already crashed). On the other hand, if the heartbeat is sent by execution of line 08, the values in this set, $\text{seq}_j(0)$, sent to process $\text{seq}_j(1)$ do not include $j$ (otherwise, the heartbeat would have been sent to $p_j$ instead). For example, let us consider that $J = 2$, $\Pi = \{p_0, p_1, p_2, p_3\}$, $p_j \in F$ and $t > \tau$. We know that $\text{seq}_j = 3 \mapsto 0 \mapsto 1 \mapsto 2$ (where $a \mapsto b$ denotes that $a$ precedes to $b$ in the sequence), and, hence, $\text{seq}_j(1) = 3$. Let us also consider that at time $\tau > \tau$ we have $\text{correct}_0 = \{2, 0\}$ and $\text{correct}_1 = \{3, 0, 1\}$. Then, process $p_0$ sends its heartbeats to process $p_2$ (which is crashed), and process $p_1$ sends its heartbeats to process $p_3$ (and as base case states, $\text{correct}_1$ does not contain 2). If $j \notin \text{correct}_i(\text{seq}_j(1))$ at time $\tau$, eventually the timer $\text{timer}_i(\text{seq}_j(1))$ will expire and $j$ will be permanently removed from $\text{correct}_i(\text{seq}_j(1))$.

Let us now consider $x \geq 2$. By induction hypothesis, after $\tau_{x-1}$, we have that $j \notin \text{correct}_i$, for all $y \in \{1, \ldots, x-1\}$. Note that now process $\text{seq}_j(x)$ will receive heartbeats from $\text{seq}_j(x-1)$ with a set $\text{correct}_i(\text{seq}_j(\text{seq}_j(x-1)))$ not containing $j$. Then, following the same reasoning of the base case, process $\text{seq}_j(x)$ will never receive any heartbeat with a set containing $j$. Hence, there is a time $\tau_x > \tau_{x-1}$ after which $j \notin \text{correct}_i(\text{seq}_j(x))$ permanently.

Therefore, for all $x \geq 1$, eventually and permanently $j \notin \text{correct}_i(\text{seq}_j(x))$. As the sequence $\text{seq}_i$ includes all correct processes, and the proof holds for all $p_j \in F$, the lemma follows. □

The following lemma assures that if eventually a correct process $p_i$ believes that a majority of processes are alive, then eventually all correct processes in the system will believe that $p_i$ is alive.

Lemma 2. If there is a process $p_i \in C$ such that eventually $|\text{correct}_i| \leq n/2$, then there is a time after which $i \in \text{correct}_j$, for every $p_j \in C$.

Proof. If there is a time after which a process $p_j \in C$ always has $|\text{correct}_j| \leq n/2$, then it will be permanently broadcasting heartbeats with $i$ each $\eta$ time (line 10). As all links are ADD, then at least one from each $B$ messages is receive in each process $p_j$ in at most $\eta B + \Lambda$ time. Hence, eventually $\text{timer}_i(i)$ will never expire, and process $p_j$ will always have $i \in \text{correct}_j$. □

The following lemma shows that if a majority of processes are crashed, eventually all correct processes have $|\text{correct}| \leq n/2$.

Lemma 3. If $|C| \leq n/2$, then there is a time after which $|\text{correct}| \leq n/2$, for every process $p_i \in C$.

Proof. From Lemma 1, there is a time after which every correct process $p_i \in C$ will have $|\text{correct}_i| \leq |C|$. Hence, if $|C| \leq n/2$, then $|\text{correct}| \leq n/2$. □

In the following lemma we show that if there is a minority of correct processes, eventually all correct processes have the same set correct.

Lemma 4. If $|C| \leq n/2$, there is a time after which correct = correct, for all $p_i, p_j \in C$.

Proof. From Lemma 3, if $|C| \leq n/2$, there is a time after which every correct process $p_i$ will broadcast heartbeats permanently with its identifier $i$ each $\eta$ time (line 10). Then, eventually no heartbeats with $j$, where $p_j \in F$, will be received by correct processes. Then, $\text{timer}_j(i)$ will expire (if it was previously set by $p_j$) and $j$ will not be in correct, anymore. Hence, there is a time after which correct does not contain identifiers of crashed processes. From Lemma 2 and Lemma 3, if $|C| \leq n/2$, there is a time after which every $p_i \in C$ will have $j \in \text{correct}_i$, for every process $p_j \in C$. Therefore, if $|C| \leq n/2$, there is a time after which correct = correct, for all $p_i, p_j \in C$. □

The following lemma shows that if a minority of processes are crashed, eventually all correct processes have $|\text{correct}| > n/2$.

Lemma 5. If $|C| > n/2$, then there is a time after which $|\text{correct}| > n/2$, for every process $p_i \in C$.

Proof. By contradiction, let us suppose that $|C| > n/2$, and there is a subset $E \neq \emptyset$ such that eventually each process $p_k \in E$ has $|\text{correct}_k| \leq n/2$. Then, there is also a complementary subset $G$ such that eventually each process $p_k \in G$ has $|\text{correct}_k| > n/2$ (being $E \cap G = \emptyset$, and $E \cup G = C$). We have two cases to study:

Case 1. $G = \emptyset$. That is, every correct process $p_k \in E$. From Lemma 2, there is a time after which $k \in \text{correct}_j$, for every $p_j \in C$. So, as by hypothesis of contradiction $|C| > n/2$, we eventually have $|\text{correct}_j| > n/2$, for every $p_j \in C$, and hence, we reach a contradiction.

Case 2. $G \neq \emptyset$. By hypothesis of contradiction we also have that $E \neq \emptyset$. Note that eventually each process $p_k \in G$, as it has $|\text{correct}_k| > n/2$, sends heartbeats permanently with
In the following lemma we show that if there is a majority of correct processes, eventually all correct processes have the same set correct.

**Lemma 7.** If $|C| > n/2$, there is a time after which correct$_i =$ correct$_j$, for all $p_i, p_j \in C$.

**Proof.** From Lemma 6, if $|C| > n/2$, there is ring$(C)$. From Lemma 1, the set correct$_i$ of each process $p_i \in C$ eventually can only contain correct processes. Then, each process $p_i \in C$ sends correct$_i$ to its successor (line 08). This successor of $p_i$ (for example $p_k$) includes the processes sent by process $p_i$ in its set correct$_k$ (line 20). Hence, as there is a ring, eventually correct$_i =$ correct$_j$, for all $p_i, p_j \in C$. Therefore, if $|C| > n/2$, there is a time after which correct$_i =$ correct$_j$, for all $p_i, p_j \in C$. □

**Theorem 1.** Let process $p_i \in C$. There is a time after which every process $p_i \in C$ permanently has leader$_i = l$.

**Proof.** From Lemma 1, there is a time after which all processes in correct$_i$ have to be correct, for every $p_i \in C$. From Lemmas 4 and 7, there is a time after which correct$_i =$ correct$_j$, for every $p_i, p_j \in C$. Note that correct$_i$ is never empty, for every process $p_i$. This is so because initially process $p_i$ includes itself in this set (line 01), and this value $i$ is never removed from correct$_i$ (because timer$_i(i)$ is never started, lines 14 and 21). Hence, there is a time after which process $p_i$ permanently has leader$_i = l$, being $l = \min(\text{correct}_i)$ (lines 23 and 27). □

**Theorem 2.** If $|C| > n/2$, the algorithm of Fig. 1 is communication-efficient and crash-quiescent.

**Proof.** From Lemmas 5 and 6, there is a ring$(C)$ if $|C| > n/2$. Then, this ring will be formed by sending heartbeats permanently among all correct processes in a cyclic way, and thus, the number of links that eventually carry messages forever is $|C|$. Hence, the algorithm of Fig. 1 is communication-efficient and crash-quiescent. □

In the following theorem we show that the algorithm of Fig. 1 also implements $\bar{P}$.

**Theorem 3.** There is a time after which each process $p_i \in C$ always has $|\text{correct}_i| = |C|$, and $p_j \in \text{correct}_i$, for all $p_j \in C$.

**Proof.** From Lemma 1, eventually crashed processes cannot be in the set correct$_i$, $\forall p_i \in C$.

If $|C| \leq n/2$, from Lemmas 2 and 3, eventually each process $p_i \in C$ has $j \in \text{correct}_i$, $\forall p_j \in C$. Hence, if $|C| \leq n/2$, there is a time after which each process $p_i \in C$ always has $|\text{correct}_i| = |C|$, and $p_j \in \text{correct}_i$, for all $p_j \in C$.

If $|C| > n/2$, from Lemma 7 we have that eventually correct$_i =$ correct$_j$, for all $p_i, p_j \in C$, and from Lemma 6 there is a unique ring formed by all correct processes. Looking at the algorithm of Fig. 1, each process $p_i$ of this unique ring sends heartbeats to its successor (i.e., process $p_j$ with correct$_j$ (line 08), and this successor $p_j$ includes the values of correct$_j$ in its correct$_j$ (lines 14 and 20). We
know that $\text{correct}_i$ always contains at least $i$, for all process $p_i$. This is so because initially process $p_i$ includes itself in this set (line 01), and this value $i$ is never removed from $\text{correct}_i$ (because $\text{timer}_i(i)$ is never started, lines 14 and 21). Hence, if $|C| > n/2$, there is a time after which each process $p_i \in C$ always has $|\text{correct}_i| = |C|$, and $p_j \in \text{correct}_i$, for all $p_j \in C$.

Therefore, there is a time after which each process $p_i \in C$ always has $|\text{correct}_i| = |C|$, and $p_j \in \text{correct}_i$, for all $p_j \in C$. □

References


