ABSTRACT
Fault-tolerant broadcast is a fundamental service in distributed systems, by which processes can communicate with each other consistently and reliably. It has two main forms: Reliable Broadcast (RB) and Uniform Reliable Broadcast (URB). This service has been extensively investigated in non-anonymous distributed systems where processes have unique identifiers, usually assume the communication channels are reliable, which is not always the case in real systems. In this paper, the fault-tolerant broadcast service is studied in an anonymous asynchronous message passing distributed system model with fair lossy communication channels. Firstly, two simple and non-quiescent algorithms implementing RB and URB are given. Secondly, two new classes of failure detectors \(A_\theta\) and \(AP^*\) are proposed. Finally, with the information provided by \(A_\theta\) and \(AP^*\), quiescent algorithms for both RB and URB are given.

Categories and Subject Descriptors
H.3.4 [System and Software]: Distributed Systems

General Terms
Algorithms, Design, Theory

Keywords
Anonymous, reliable broadcast, failure detector, quiescent.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author. Copyright is held by the owner/author(s).
ACM 978-1-4503-4676-7/15/07.
http://dx.doi.org/10.1145/2767386.2767443.
Processes communicate with each other by sending and receiving messages through fair lossy communication channels. Formally, a channel between two processes and is fair lossy if it satisfies the following properties [4]: If sends a message to an infinite number of times and is correct, then eventually receives from in (Fairness); If receives a message from , then previously sent to ; and if receives infinitely often from , then sends infinitely often to in (Uniform Integrity). Two basic primitives are used in the system to send and receive messages: broadcast() and receive(). We say that a process broadcasts a message when it invokes broadcast(). Similarly, a process receives a message when it invokes receive(). Note that in anonymous distributed systems, when a process receives a message it cannot determine who is the sender of.

The system model is denoted by either or . is an acronym for anonymous asynchronous message passing system with fair lossy communication channels; means that there is no additional assumption, while means that the system is enriched with a failure detector of class . The variable represents the total number of processes in the system, and represents the maximum number of processes that can fail. Reliable Broadcast (RB) is a broadcast form that satisfies the following three properties:

- **Validity**: If a correct process broadcasts a message , then it eventually delivers .
- **Agreement**: If a correct process delivers a message , then all correct processes eventually deliver .
- **Uniform Integrity**: For any message , every process delivers at most once, and only if was previously broadcast by a process.

Note that Validity and Agreement imply that all correct processes deliver all the messages broadcast by correct processes. Uniform Reliable Broadcast (URB) is a broadcast form stronger than RB, satisfying Validity and Uniform Integrity (as RB), and Uniform Agreement, defined as follows:

- **Uniform Agreement**: If some process (correct or not) delivers a message , then all correct processes eventually deliver .

### 3. IMPLEMENTATION OF FAULT TOLERANT BROADCAST IN \( AAS_F \)

A key point to implement the fault-tolerant broadcast service in is how to distinguish a process or a message from others. In this paper, we solve this by making each message to be unique. This idea is easily implemented in non-anonymous systems, usually by using the identifier of the sender process and a sequence number. However, in anonymous systems this method cannot be used due to the fact that processes do not have identifiers. Hence, we propose to use a random function, assigning a different random number to each message as its unique label or tag. We assume that each number generated randomly is unique in the system. Note that assigning a label to a message is better than to a process, because there is no way to deduce the messages sent from a given process. Indeed, all processes still maintain their anonymity in the system.

With the idea mentioned above and a simple flooding transmission technique, an implementation algorithm of RB in and another algorithm implementing URB in are proposed in our full paper [6]. Both algorithms are non-quiescent, i.e., delivered messages need to be re-broadcast periodically in order to cope with fair lossy communication. Additionally, we show that it is impossible to implement URB without a majority of correct processes. Hence, in the next section we enrich the anonymous system model with two failure detectors in order to implement RB and URB quiescently and with any number of correct processes.

### 4. TWO CLASSES OF ANONYMOUS FAILURE DETECTORS

The failure detector abstraction was proposed Chandra and Toueg [5], which defined it in terms of completeness and accuracy properties. A failure detector can be seen as a module that gives (possibly unreliable) failure information of processes. Usually, this failure information is composed of the identifiers of processes in non-anonymous distributed systems. However, in anonymous distributed systems processes have no identifier. So, the key point to define and implement a failure detector in an anonymous distributed system is how to identify a process without breaking the anonymity of the system. We were inspired by the definition of the failure detector class \( A_\Sigma \), which was introduced by Bonnet and Raynal [7], that assigns a random label to each process as a temporal identifier. This assignment does not break the anonymity of the system, because this label is given in the failure detector layer, and no process knows the mapping relationship between labels and processes.

#### 4.1 Failure Detector \( A\Theta \)

\( A\Theta \) provides each process with a read-only local variable \( a_{\text{theta}} \), that contains several pairs of (label, number), in which each label represents a temporary identifier of one process and number represents the number of correct processes who have known this label. For example, process ’s local variable at time is \( a_{\text{theta}} = \{(\text{label}_1, \text{number}_1), \ldots, (\text{label}_n, \text{number}_n)\} \) if has known the label of processes of the system. As we said previously, labels are assigned randomly to each process without breaking the anonymity of the system.

The definition of \( A\Theta \) is given as follows:

- **Completeness**: There is a time after which variables \( a_{\text{theta}} \) contain pairs of (label, number) associated to correct processes.
- **Accuracy**: If there is a correct process, then at every time, all pairs of (label, number) output by failure detector \( A\Theta \) hold that every subset \( T \) of size number of processes that know label contains at least one correct process (i.e., for each label, there always exists one correct process in any set of processes that know this label).

Let us define \( A\Theta \) more formally:

- \( S(\text{label}) = \{i \mid \exists \tau \in \mathbb{N}, (\text{label}, -) \in a_{\text{theta}}^\tau \} \). \( S(\text{label}) \) is the set of all processes that know label at time .
- \( a_{\text{theta}} \) contains: \( \exists \tau \in \mathbb{N}, \forall i \in \text{Correct}, \forall \tau' \geq \tau, \forall (\text{label, number}) \in a_{\text{theta}}^\tau, |S(\text{label}) \cap \text{Correct}| = \text{number} \).
• $A\Theta$-accuracy: $\text{Correct} \neq \emptyset \implies \forall \tau \in \mathbb{N}, \forall i \in \mathbb{N}, \forall (\text{label}, \text{number}) \in a_{\theta i}\; S(\text{label}) \subseteq \text{Correct} \neq \emptyset$.

4.2 Failure Detector $AP^*$

The anonymous perfect failure detector $AP^*$ provides each process $p_i$ with a read-only local variable $a_{p_i}$ that contains several pairs of $(\text{label}, \text{number})$, similar to failure detector $A\Theta$. The definition of $AP^*$ is as follows:

• $AP^*$-completeness: There is a time after which variables $a_{p_i}$ only contain pairs of $(\text{label}, \text{number})$ associated to correct processes.

• $AP^*$-accuracy: If a process crashes, the label of this process and the corresponding number to the label will be permanently removed from variables $a_{p_i}$.

It is assumed that the number of each label is monotonically non-increasing. Eventually the number of pairs of $(\text{label}, \text{number})$ is equal to the number of correct processes.

Let us define $AP^*$ more formally:

$S(\text{label}) = \{ i \mid \exists \tau \in \mathbb{N}, (\text{label}, \_ \_ ) \in a_{\tau i} \}$. $S(\text{label})$ is the set of all processes that know label at time $\tau$.

• $AP^*$-completeness: $\exists \tau \in \mathbb{N}, \forall i \in \text{Correct}, \forall \tau' \geq \tau, \exists (\text{label}, \text{number}) \in a_{\text{p}_i}\; |S(\text{label}) \cap \text{Correct}| = \text{number}$.

• $AP^*$-accuracy: $\forall i, j \in \mathbb{N}, i \in \text{Correct}, j \in \text{Faulty}, \exists \tau, \forall \tau' \geq \tau: (\text{label}_i, \text{number}_j) \notin a_{\text{p}_i}\; ^{\tau'}$.

5. QUIESCENT FAULT TOLERANT BROADCAST IN $AAS_F[A\Theta, AP^*]$

Before solving the quiescence problem, the impossibility result of implementing URB without a majority of correct processes is circumvented firstly with $A\Theta$. To do so, in our algorithm an adequate use of $A\Theta$ ensures that the acknowledge of a message $m$ has been received by at least one correct process before delivering $m$.

The reason why both the RB and URB algorithms of Section 3 are non-quiescent is that each correct process has to broadcast periodically the messages it has delivered in order to overcome message losses caused by the fair lossy communication channels. Therefore, the intuitive idea to make them to be quiescent is to stop the periodical broadcast when a message has been delivered by all correct processes. The following question arises: how can we confirm that a message has been delivered by all correct processes in an anonymous system? In order to answer this question, we propose to combine two mechanisms: (1) each process broadcasts an acknowledgment message after it has delivered a message $m$, and waits for the acknowledgment messages of $m$ from all processes (including itself); (2) the failure detector $AP^*$ is used to provide the information of all correct processes. With these two mechanisms, a process $p_i$ can stop the periodical broadcast of message $m$ when it has received the acknowledgment of $m$ from all correct processes. Note that these two mechanisms do not break the anonymity of the system.

Following this approach, a quiescent RB algorithm implemented in $AAS_F[AP^*]$ and a quiescent URB algorithm in $AAS_F[A\Theta, AP^*]$ are given in our full paper [6].

6. CONCLUSION

In this paper, the fault-tolerant broadcast service has been studied in anonymous asynchronous message passing distributed systems with fair lossy communication channels. Initially, two non-quiescent algorithms have been proposed: the non-quiescent reliable broadcast (RB) algorithm can be implemented under the assumption of any number of correct processes, while the non-quiescent uniform reliable broadcast (URB) algorithm requires either a majority of correct processes or a failure detector of class $A\Theta$. In order to implement the fault-tolerant broadcast service quiescently, a new anonymous perfect failure detector $AP^*$ is proposed. Finally, two quiescent algorithms using $AP^*$ and implementing RB and URB respectively are proposed.

Acknowledgment

We would like to thank the anonymous reviewers. This research is partially supported by the Community of Madrid under grant S2013/ICE-2894, the Spanish Research Council under grants TIN2013-41123-P and TIN2013-46883-P, the Basque Government under grant IT309-10, the University of the Basque Country (UPV/EHU) under grant UF11/45, and the scholarship of Chinese Scholarship Council.

7. REFERENCES