Byzantine Agreement with Homonyms (Accord Byzantin avec des Homonymes)

Carole Delporte-Gallet, Hugues Fauconnier, Rachid Guerraoui, Anne-Marie Kermarrec, Eric Ruppert, Hung Tran-The

Byzantine Agreement with Homonyms
Accord Byzantin avec des Homonymes †

Carole Delporte-Gallet[1], Hugues Fauconnier[1], Rachid Guerraoui[2], Anne-Marie Kermarrec[3], Eric Ruppert[4] et Hung Tran-The[1]

[1]*LIAFA, Université Paris Diderot, France.* [2]*Ecole Polytechnique Fédérale de Lausanne Suisse.* [3]*INRIA Rennes-Bretagne Atlantique, France.* [4]*York University, Canada.*

Dans un système réparti, on suppose le plus souvent que chaque processus a une identité unique. Plus rarement certains travaux s'intéressent au cas où les processus sont anonymes. Ces deux cas sont les deux extrêmes d'un modèle général où n processus utilisent ℓ identités différentes. Dans ce modèle, combien faut-il d'identifiants pour permettre l'accord lorsque t processus peuvent être byzantins ?

On montre que $3t+1$ identifiants sont nécessaires et suffisants pour obtenir l'accord Byzantin dans un système synchrone. Mais de manière surprenante le nombre d'identifiants doit être supérieur à $\frac{n+3t}{2}$ dans un système partiellement synchrone. Ceci montre deux différences avec le modèle classique ($\ell = n$) : (1) dans le modèle classique, les conditions pour permettre l'accord byzantin dans un système synchrone sont les mêmes que dans un système partiellement synchrone. Ici il y a des valeurs de ℓ pour lesquelles ce n'est pas le cas, (2) dans un système partiellement synchrone augmenter le nombre de processus *corrects* peut rendre l'accord impossible.

Les preuves d'impossibilité utilisent le fait qu'un processus byzantin peut envoyer plusieurs messages à un même processus dans une ronde. On montre que si on enlève cette possibilité, l'accord byzantin devient plus facile à obtenir et que $t+1$ identifiants sont suffisants, même dans un système partiellement synchrone.

Keywords: Accord Byzantin, anonymat

1 Introduction

We consider a distributed system with Byzantine failures in which ℓ distinct identifiers are assigned to n processes, where $1 \leq \ell \leq n$. Several processes may be assigned the same identifier, in which case we call the processes *homonyms*. If a process p receives a message from a process q with identifier i, then p knows that the message sent by some process with identifier i, but p does not know whether the message was sent by q or another process q' having the same identifier i. This is true even if q is Byzantine : a Byzantine process cannot change its own identifier. A process cannot direct a message it sends to a particular process, but can direct the message to all processes that have a particular identifier.

This model generalizes the classical scheme where processes have distinct identifiers (i.e., $\ell = n$), and the less classical scheme where processes are anonymous (i.e., $\ell = 1$). Studying systems with homonyms provides a better understanding of the importance of identifiers in distributed computing, and there are two additional motivations for the new model. Firstly, assuming in systems such as Pastry or Chord [RD01, SMK+01] that all processes have unique (unforgeable) identifiers might be too strong an assumption in practice. We may wish to design protocols that still work if, by a rare coincidence, two processes are assigned the same identifier. This approach is also useful if security is breached and a malicious process can forge the identifier of a correct process, for example by obtaining the correct process's private key. Secondly, in many cases, users of a system may wish to preserve their privacy by remaining anonymous.

†This work has been accepted for publication at PODC 2011 [DGFG+11].This work is partially supported by the ERC Starting Grant project 204742 and the ANR VERSO SHAMAN.

Unfortunately, in a fully anonymous system where no identifiers are used, very few problems are solvable. In particular, Okun observed that Byzantine agreement is impossible in the fully anonymous model [Oku05], even with a single faulty process. With a limited number of identifiers, more problems become solvable, and some level of anonymity can be preserved by hiding, to some extent, the association between users and identifiers. For example, users of a distributed protocol might use only their domain names as identifiers. Others will see that some user within the domain is participating, but will not know exactly which one. If several users within the same domain participate in the protocol, they will behave as homonyms.

We ask in this paper how many distinct identifiers are needed to reach *agreement* in a system of n processes, up to t of which can be Byzantine. We need only to consider systems where $n > 3t$: this assumption is known to be a requirement for solving Byzantine agreement, even when $\ell = n$ [PSL80,LSP82], and it thus applies also for systems with homonyms. For the synchronous case, we prove using a scenario argument that $3t+1$ identifiers are necessary. The matching synchronous algorithm is obtained by a simulation that transforms any synchronous Byzantine agreement algorithm designed for a system with unique identifiers to one that works in a system with $\ell > 3t$ identifiers. For the partially synchronous case, we prove using a partitioning argument that the lower bound becomes $\ell > \frac{n+3t}{2}$. (The bound $\frac{n+3t}{2}$ is strictly greater than $3t$ because $n > 3t$.) We show that this bound is also tight by giving a new partially synchronous Byzantine agreement algorithm. This bound is somewhat surprising because the number of required identifiers ℓ depends on n as well as t. Counter-intuitively, increasing the number of correct processes can render agreement impossible. For example, if $t = 1$ and $\ell = 4$, agreement is solvable for 4 processes but not for 5. Another difference from the classical situation (where $\ell = n$) is that the condition that makes Byzantine agreement solvable is different for the synchronous and partially synchronous models.

To strengthen our results, we show that (a) both the synchronous and partially synchronous lower bounds hold even if correct processes are *numerate*, i.e., can count the number of processes that send identical messages in a round and (b) the matching algorithms are correct even if processes are *innumerate*. In systems with unique identifiers, senders can append their identifier to all messages, making it trivial for the receiver to count copies of messages. This is not possible in systems with homonyms, so the distinction between numerate and innumerate processes is important.

What has more impact, however, is the ability for a Byzantine process to send multiple messages to a single recipient in a round. In a classical system with unique identifiers, the Byzantine process has no advantage in doing this : algorithms could simply discard such messages. In systems with homonyms, there is a clear advantage. In fact, we prove that if each Byzantine process is restricted to sending a single message per round to each recipient (and processes are numerate), then $t+1$ identifiers are enough to reach agreement even in a partially synchronous model. We also show this bound is tight using a valency argument : $t+1$ identifiers are needed even in the synchronous case. The fact that $t+1$ identifiers are sufficient to reach agreement with restricted Byzantine processes has some practical relevance : In some settings, it is reasonable to assume that Byzantine processes are simply malfunctioning ordinary processes sending incorrect messages, and not malicious processes with the additional power to generate and send more messages than correct processes can.

Our results are summarized in Table 1. For lack of space, we refer the reader to the long version [DGFG+11] for formal definitions, theorems and proofs.

	Synchronous	Partially synchronous
Innumerate processes	$\ell > 3t$	$\ell > \frac{n+3t}{2}$
Numerate processes	$\ell > 3t$	$\ell > \frac{n+3t}{2}$
Numerate processes, restricted Byzantine processes	$\ell > t$	$\ell > t$

TABLE 1: Necessary and sufficient conditions for solving Byzantine agreement in a system of n processes using ℓ identifiers and tolerating t Byzantine failures. In all cases, n must be greater than $3t$.

2 Definitions

We consider a distributed message-passing system with $n \geq 2$ processes. Each process has an identifier from the set $\{1, ..., \ell\}$. We assume that $n \geq \ell$ and that each identifier is assigned to at least one process. In the

case where $n > \ell$, one or more identifiers will each be shared by several processes. We assume algorithms are deterministic. Processes with the same identifier execute the same code but processes with different identifiers may behave differently.

A *correct* process does not deviate from its algorithm specification. A process that is not correct is called *Byzantine*. The maximum possible number of Byzantine processes is denoted t, where $0 < t < n$. We need only to consider systems where $n > 3t$: this assumption is known to be a requirement for solving Byzantine agreement, even when $\ell = n$ [PSL80, LSP82], and it thus also applies to systems with homonyms. A Byzantine process may choose to send arbitrary messages (or no message) to each other process.

Without loss of generality, we assume a broadcast model : when a correct process sends a message, it can sends the same message to all processes. Byzantine processes are not bound by this assumption : they may send different messages to different processes, even if the recipients have the same identifier.

In the *synchronous model*, computation proceeds in rounds. In each round, each process can send messages to all other processes and then receive all messages that were sent to it during that round.

For the *partially synchronous model* we use the definition of Dwork, Lynch and Stockmeyer [DLS88] : computation proceeds in rounds, as in the synchronous model. However, in each execution, a finite number of messages might not be delivered to all of their intended recipients. As argued in [DLS88], this basic partially synchronous model is equivalent to other models with partially synchronous communication.

As mentioned in the introduction, we also consider variants of the models in which each Byzantine process is *restricted* to sending at most one message to each recipient in each round. In general, we consider unrestricted Byzantine processes unless the restriction is explicitly mentioned. We also distinguish the cases where processes are *innumerate* from the case where they are *numerate*. We say that a process is innumerate if the messages it receives in a round form a set of messages : the process cannot count the number of copies of identical messages it receives in the round. We say that a process is numerate if the messages it receives in a round form a multiset of messages : the process can count the number of copies of identical messages it receives in the round.

The goal of an agreement algorithm is for a set of processes proposing values to decide on exactly one of these values. We consider the classical *Byzantine agreement* problem [FLM86, PSL80], defined by the following three properties : *(Validity)* If all correct processes propose the same value v, then no value different from v can be decided by any correct process, *(Agreement)* No two correct processes decide different values, and *(Termination)* Eventually, each correct process decides some value.

An algorithm solves Byzantine agreement in a system of n processes with ℓ identifiers tolerating t failures if these three properties are satisfied in every execution in which at most t processes fail, regardless of the way the n processes are assigned the ℓ identifiers.

3 Synchronous Model

We prove the condition $\ell > 3t$ is necessary even if processes are numerate using a scenario argument, in the style of Fischer, Lynch and Merritt [FLM86].

Our agreement algorithm that solves Byzantine agreement assuming $\ell > 3t$ is generic : given any synchronous Byzantine agreement algorithm for ℓ processes with unique identifiers (such algorithms exist when $\ell = n > 3t$, e.g., [LSP82]), we transform it into an algorithm for n processes and ℓ identifiers, where $n \geq \ell$. In our transformation, we divide processes into groups according to their identifiers. Each group simulates a single process. If all processes within a group are correct, then they can reach agreement and cooperatively simulate a single process. If any process in the group is Byzantine, we allow the simulated process of that group to behave in a Byzantine manner. The correctness of our simulation relies on the fact that more than two-thirds of the simulated processes will be correct (since $\ell > 3t$), which is enough to achieve agreement. Our algorithm solves synchronous Byzantine agreement even with innumerate processes if $\ell > 3t$.

4 Partially Synchronous Model

We prove that having $\ell > \frac{3t+n}{2}$ is necessary and sufficient for solving Byzantine agreement in a partially synchronous system, regardless of whether the processes are numerate or innumerate. Intuitively, this condition means that at least $3t+1$ of the identifiers must each be assigned to a single process (since $2\ell - n > 3t$).

We prove the necessity of the condition $\ell > \frac{n+3t}{2}$ using a partitioning argument. We show that if there are too few identifiers, and messages between two groups of correct processes are not delivered for sufficiently long, then the Byzantine processes can force processes in the two groups to decide different values.

Our algorithm is based on the algorithm given by Dwork, Lynch and Stockmeyer [DLS88] for the classical case where $n = \ell$, with several novel features. Generalizing the algorithm is not straightforward. Some of the difficulty stems from the following scenario. Suppose two correct processes share an identifier and follow the traditional algorithm of [DLS88]. They could send very different messages (for example, if they have different input values), but recipients of those messages would have no way of telling apart the messages of the two correct senders, so it could appear to the recipients as if a single Byzantine process was sending out contradictory information. Thus, the algorithm has to guard against inconsistent information coming from correct homonym processes as well as malicious messages sent by the Byzantine processes.

5 Restricted Byzantine Processes

We now consider the effect of restricting the Byzantine processes so that each Byzantine process can send at most one message to each recipient in each round. This restriction reduces the number of identifiers needed to reach agreement if processes are numerate but does not help if processes are innumerate. With numerate processes, having $\ell > t$ is necessary and suf ?cient for solving Byzantine agreement. With innumerate processes, the previous results remain unchanged.

With numerate processes, the algorithm that solves partially synchronous Byzantine agreement is similar to the one presented in Section 4. Like this algorithm, it uses an authenticated broadcast primitive [ST87], that ensures the agreement property, but here accept actions have an extra parameter indicating the multiplicity of the accepted message. This multiplicity is greater than the number of correct processes that sent the message and does not exceed the number of correct processes by more than the actual number of Byzantine processes in the execution. Eventually, all correct processes agree on the multiplicity of each message. As $\ell > t$, at least one identifier is assigned only to correct processes ensuring the termination property.

Références

[DGFG+11] Carole Delporte-Gallet, Hugues Fauconnier, Rachid Guerraoui, Anne-Marie Kermarrec, Eric Ruppert, and Hung Tran-The. Byzantine agreement with homonyms. In *Proc. 30th ACM Symposium on Principles of Distributed Computing*, pages 21–30, 2011.

[DLS88] Cynthia Dwork, Nancy A. Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the ACM*, 35(2) :288–323, April 1988.

[FLM86] Michael J. Fischer, Nancy A. Lynch, and Michael Merritt. Easy impossibility proofs for distributed consensus problems. *Distributed Computing*, 1(1) :26–39, January 1986.

[LSP82] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3) :382–401, July 1982.

[Oku05] Michael Okun. Agreement among unacquainted Byzantine generals. In *Proc. 19th International Conference on Distributed Computing*, volume 3724 of *LNCS*, pages 499–500, 2005.

[PSL80] Marshall Pease, Robert Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2) :228–234, April 1980.

[RD01] Antony I. T. Rowstron and Peter Druschel. Pastry : Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware*, volume 2218 of *LNCS*, pages 329–350, 2001.

[SMK+01] Ion Stoica, Robert Morris, David R. Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord : A scalable peer-to-peer lookup service for internet applications. In *ACM SIGCOMM*, pages 149–160, 2001.

[ST87] T. K. Srikanth and Sam Toueg. Optimal clock synchronization. *Journal of the ACM*, 34(3) :626–645, July 1987.