N-party BAR Transfer

Xavier Vilaça, João Leitão, Miguel Correia, and Luís Rodrigues

INESC-ID, Instituto Superior Técnico, Universidade Técnica de Lisboa

Abstract. We introduce the N-party BAR transfer problem that consists in reliably transferring arbitrarily large data from a set of N producers to a set of N consumers in the BAR model, i.e., in the presence of Byzantine, Altruistic, and Rational participants. The problem considers the existence of a trusted observer that gathers evidence to testify that the producers and consumers have participated in the transfer. We present an algorithm that solves the problem for $N \geq 2f + 1$, where f is the maximum number of Byzantine processes in each of the producer and consumer sets. We do not impose limits on the number of Rational participants, although they can deviate from the algorithm to improve their utility. We show that our algorithm provides a Nash equilibrium.

1 Introduction

Peer-to-peer systems may be used to provide temporary or long-term storage services. Such services are useful in a number of settings. For instance, peer-to-peer systems can be used to process large volumes of data using volunteer computation, as illustrated by projects such as SETI@home [4] and, more recently, by the Boinc infrastructure that supports several computationally intensive research projects [3]. If such computations are performed using MapReduce, information produced by mappers needs to be transferred to the reducers or to intermediate storage. Volunteer storage nodes may not be willing to store data indefinitely, so they have to transfer data to other nodes after serving the system for some time. In any case, volunteers expect to be recognized for their contribution, for instance by being awarded credits that make them appear in a chart with the top contributors of the project.

In scenarios such as the ones listed above, a reliable protocol to transfer data from a set of producers to a set of consumers is an important building block. Any realistic service for this environment has to consider the existence of both Byzantine and Rational nodes, i.e., of nodes that deviate from the protocol, respectively, in an arbitrary way (Byzantine) and with the purpose of gaining some measurable benefit like being listed as top contributors without really executing jobs (Rational). A system model that captures the existence of these different kinds of participants is the Byzantine-Altruistic-Rational (BAR) model [2].

This paper introduces the N-party BAR Transfer problem (BAR-Transfer). This problem can be informally defined as follows. There are N producers and N consumers, which we generically call processes. Up to f processes of each of these sets can be Byzantine; the remainder are either Altruistic or Rational. All non-Byzantine producers have the same piece of arbitrarily large data that they have to transfer to all non-Byzantine consumers. Altruistic processes follow the protocol, Byzantine processes deviate arbitrarily from the protocol (e.g., omitting or sending modified messages), and Rational processes deviate from the protocol following a strategy to increase their utility. There is an abstract trusted observer that is not involved in the transfer, but that collects evidence about it. BAR-Transfer is the problem of reliably transferring data from the producers to the consumers, while providing the trusted observer enough evidence to testify which processes participated in the transfer.

Systems not designed to cope with Rational behaviour may fall into the Tragedy of Commons [15]: the job is not done because all participants are Rational and aim for profit by not performing (part of) their role. To model Rational behaviour, we use an approach based on Game Theory [25]. The protocol executed by the processes is modelled as a game, in which each player (i.e., process) follows a strategy to increase its utility. To contradict this behaviour, an algorithm to solve BAR-Transfer should provide a Nash equilibrium, so that no Rational process has an incentive to deviate from the protocol. We model the BAR-Transfer problem as a strategic game, in which players choose a strategy simultaneously, once and for all [25], i.e., without knowledge of the others strategies and without the ability of changing it during the algorithm execution. This is not a restriction in the case of our algorithm as explained later. We do the usual assumption [9] that processes are risk-averse, i.e., that they do not follow a strategy that may put their profit at risk.

Besides introducing the BAR-Transfer problem, we present an algorithm that solves it in a synchronous message-passing distributed system. We prove its correctness and that it provides a Nash equilibrium which is a dominant strategy. Therefore, all Rational processes should follow our solution. The paper makes the following main contributions: i) defines the BAR-Transfer problem; ii) proposes an algorithm that solves BAR-Transfer; iii) proves the correctness of the algorithm and that it provides a Nash equilibrium.

The remaining of the paper is structured as follows. Section 2 compares this work with related work. Section 3 describes the system model and defines the BAR-Transfer problem. The algorithm to solve the problem is presented in Section 4. The correctness and cost of the algorithm are analysed in Section 5. Finally, Section 6 concludes the paper.

2 Related Work

The BAR-Transfer problem is related to classical distributed systems problems such as Byzantine Agreement (BA), Reliable Broadcast (RB), Terminating Reliable Broadcast (TRB), and Interactive Consistency (IC) [19, 11, 7]. A first and major difference is that these algorithms are executed among a single set of processes, while BAR-Transfer is about communication and agreement between two sets: producers and consumers. In that sense there is some resemblance with Paxos with its three process roles – proposers, acceptors, learners – but in BAR-Transfer all producers are proposers of the same value, a notion that does not exist in Paxos [18]. An algorithm for solving BAR-Transfer might be implemented by running N instances of algorithms that solved these problems, or even a single one in the case of IC. However, these solutions would be very inefficient in terms of message, time, and bit complexity because they would not exploit the fact that all (non-Byzantine) producers send the same data. Furthermore, these problems do not consider the BAR model. If there were Rational processes, algorithms that solved these problems would not satisfy their properties. The same discussion applies to All-to-All Reliable Broadcast (ATA-RB) algorithms [20, 14]. Although they reduce the number of messages sent when compared with parallel executions of BA, RB or TRB, to the best of our knowledge there is no work in ATA-RB algorithms in the BAR model. Besides, these algorithms only provide probabilistic guarantees.

Many Byzantine fault-tolerant algorithms have some relation to our work. Several papers presented implementations of registers based on Byzantine quorum systems [23, 24]. Others presented algorithms to implement state machine replication, a generic solution to implement fault-tolerant distributed services [8, 17]. In both cases the objective is to ensure that Byzantine nodes are unable to disrupt the consistency of the data stored in the servers or the service provided by the servers. In contrast, our work aims at ensuring the transference of a correct value from a set of nodes that produce the data independently, although following a deterministic function, to another set of nodes which have to determine which is the correct data. A third set of papers presented Byzantine fault-tolerant consensus algorithms for asynchronous systems, which might also be used as building blocks of less efficient solutions of BAR-Transfer [12, 6, 10]. Again, none of these works considers the BAR model.

Some works applied Game Theory to problems involving both Rational and Byzantine players. Eliaz introduced the notion of k-Fault Tolerant Nash equilibrium (k-FNTE), as an equilibrium in which no Rational participant has any incentive to unilaterally deviate from the expected behaviour, with up to k players whose strategy is arbitrary [13]. This concept was applied to auctions. Abraham et al. extended the work of [13] by introducing the notion of (k, t)-robustness, where k is the maximum number of colluding Rational participants and t is the upper limit to the number of Byzantine players [1]. The authors propose a solution for secret sharing that is (k, t)-robust. Contrary to our work, they assumed that the utility of each player depends only on the output of the algorithm, therefore ignoring communication costs. It has been proved that no non-trivial distributed protocol for which Rational nodes take into consideration communication costs can be (k, t)-robust [9].

The Byzantine-Altruistic-Rational (BAR) model was proposed as an abstraction for capturing these three distinct behaviours of processes [2]. The authors also proposed a general three-tied architecture for developing BAR-tolerant protocols, in cooperative distributed systems that span Multiple Administrative Domains (MAD). The first two levels of the proposed architecture implement a Replicated State Machine using a BAR-tolerant TRB protocol [9] and a mechanism than enforces periodic work and guarantees responses. Although this architecture might be used to solve the BAR-Transfer problem, the use of the TRB protocol for transferring arbitrarily large data is too costly and the guaranteed response mechanism requires the active participation of a witness, which must be either a centralized entity or implemented through message broadcast to all the remaining nodes. Furthermore, the proposed mechanisms are based on the long-term cooperation between participants modelled as a repeated game [25], which is not the case of the BAR-Transfer problem.

The same authors [9] have shown that the Dolev and Strong's TRB protocol [11] can be changed to provide a Nash equilibrium in the BAR model using ∞-tit-for-tat mechanisms [5]. The problem is modelled as a repeated game with an infinite number of rounds. Each round a different participant runs an instance of the protocol to broadcast its information to all the remaining non-Byzantine participants. They proved that Rational participants cannot expect any increase in their utility by omitting messages, even if a fraction of the participants is Byzantine. In this work we are interested in large peer-to-peer networks in which it is unlikely that the same participants interact more than once. For that reason we do not consider repeated executions, but model the algorithm instead in terms of a strategic game in which players interact only once. Therefore, in our case it is not possible to apply the incentive mechanisms of [2, 22, 21] based on tit-for-tat. Furthermore, none of these works addresses the problem of transferring an arbitrarily large value without using an active witness or direct reciprocity.

The BAR model has also been used with gossip data dissemination algorithms [22, 21]. These algorithms are not directly applicable to solve BAR-Transfer as they assume that the source of the information is trusted and provide no guarantee that the disseminated information reaches its destination. Furthermore, data transfer between each pair of nodes is performed using direct reciprocity in a fair exchange process. This requires that each Rational participant has incentives to transfer data if it expects to receive an equivalent contribution from its peer. In addition, the pestering mechanism of BAR Gossip [22] only provides a Nash equilibrium if a certain fraction of the participants are Altruistic [26]. In BAR-Transfer, consumers do not possess any data that may serve as currency to pay the producers for the transfer, and no assumption is made about the presence of Altruistic participants. Equicast [16] also implements a dissemination protocol in an environment with selfish participants, which is proven to provide a Nash equilibrium. However, it assumes that Rational processes only deviate from the protocol by adjusting a cooperation factor.

3 System Model and Problem Statement

3.1 System Model

The BAR-Transfer problem involves a set of *producers* $\mathcal{P}$ of cardinality $N_{\mathcal{P}}$ and a set of *consumers* $\mathcal{C}$ of cardinality $N_{\mathcal{C}}$. To simplify the description of our algorithm, in this paper, we consider that the cardinality of both sets is the same, i.e., $N_{\mathcal{P}} = N_{\mathcal{C}} = N$. We do not address the problem of forming these sets, in this paper. However, we assume that this mechanism ensures with high probability that the number of Byzantine processes is upper bounded and that processes cannot influence this mechanism. There is also a special process called *trusted observer* (TO). We use the words *processes* or *participants* to designate these entities. Sometimes we use the word *players* to designate producers and consumers, when we model their interaction as a game.

We assume that the system is synchronous (there are maximum communication and processing delays) and that all processes are fully connected by authenticated reliable channels. This is a reasonable assumption as we require that the transfer may terminate after a finite period of time such that Rational processes may have some guarantees that they will be eventually rewarded. However, it is not strictly necessary for the communication and processing delays to be upper bounded. Nevertheless, in order to simplify the description of our algorithm, we will make that assumption. We also assume that each process has a public-private key pair and that there is a public-key infrastructure in place, so every process has access to the public key of all others. Each process has access to a collision-resistant hash function (*hash*) and a signature function based on public-key cryptography (*sign*, *verifysig*).

Participants can be Byzantine, Altruistic, and Rational, in accordance with the BAR model. We assume that up to f elements of each of the $\mathcal{P}$ and $\mathcal{C}$ sets can be Byzantine. Any number of consumers and producers can be Altruistic or Rational. The trusted observer TO always follows its protocol.

An Altruistic process is one that follows the protocol. A Byzantine process can deviate arbitrarily from its behaviour, e.g., by sending or not sending certain messages, or by sending messages in a format or with content that is not according to the protocol. Byzantine processes however are not able to break the cryptographic mechanisms used in the algorithm (e.g., they are not able to generate signatures on behalf of Altruistic or Rational processes).

A Rational process is one that aims at maximizing a *utility function*, defined in terms of *benefits* and *costs*. A producer has a benefit by proving to the *TO* that it has contributed to the transfer; it incurs on the cost of sending the data. Consumers send to the *TO* acknowledgements of the reception of the data. A consumer benefits by obtaining the data and proving its reception to the *TO*; it incurs in the costs of receiving and processing messages and sending the acknowledgements to the *TO*. We assume that there is no collusion among Rational processes.

3.2 The BAR-Transfer Problem

The BAR-Transfer problem can be defined as follows. Each producer p has a value (or data) of arbitrary size v_p such that, for any two non-Byzantine producers p_i and p_j, $v_{p_i} = v_{p_j} = v$. Sometimes we refer to this value as the *correct value*, to denote that it is the value held by all non-Byzantine producers.

The algorithm terminates successfully when every non-Byzantine consumer consumes v. A consumer c is said to *consume* value v_c when the primitive *consume*(c, v_c) is called. All non-Byzantine producers start the algorithm by producing value v. A producer p is said to *produce* value v_p by calling the primitive *produce*(p, v_p). The *TO* is said to *produce evidence* about the transfer by calling primitive *certify*(*TO*, *evidence*). There are also two predicates *hasProduced(evidence,* p_i*)* and *hasAcknowledged(evidence,* c_j*)* that take as input the evidence produced by the *TO* to indicate, respectively, if producer p_i participated in the BAR-Transfer and if consumer c_j notified the reception of the correct value. The problem consists informally in i) transferring the value from the producers to the consumers; and ii) providing evidence about the transfer. More formally the problem is defined in terms of the following properties:

– **BAR-Transfer 1** *(Validity):* If a non-Byzantine consumer consumes v, then v was produced by some non-Byzantine producer.
– **BAR-Transfer 2** *(Integrity):* No non-Byzantine consumer consumes more than once.
– **BAR-Transfer 3** *(Agreement):* No two non-Byzantine consumers consume different values.
– **BAR-Transfer 4** *(Termination):* Every non-Byzantine consumer consumes a value.
– **BAR-Transfer 5** *(Evidence):* The trusted observer produces evidence about the transfer.
– **BAR-Transfer 6** *(Producer Certification):* if producer p is non-Byzantine, then *hasProduced(evidence, p)* is *true.*
– **BAR-Transfer 7** *(Consumer Certification):* if consumer c is non-Byzantine, then *hasAcknowledged(evidence, c)* is *true.*

With these definitions in mind, we can provide a more precise characterization of the *benefits* that Rational nodes aim to obtain. The benefit of a producer p is to have *hasProduced(evidence, p) true.* The benefit of a consumer c is twofold: i) to obtain the correct value and ii) to have *hasAcknowledged(evidence, c) true.*

4 BAR-Transfer Algorithm

We now present an algorithm that solves the BAR-Transfer problem (Alg. 1). The algorithm requires $N \geq 2f+1$ producers and consumers. The algorithm aims at ensuring that each consumer receives the value and can decide which is the correct value, in case it receives several different values (e.g., due to Byzantine producers). To satisfy this goal, each producer is not required to send a copy of the (possibly large) value to every consumer. In fact, it is enough that it sends the value to $f+1$ consumers and a signed hash of the value to the remaining $N-f-1$ consumers.

We define a deterministic function that returns the set of consumers that receive a copy of the value from producer p_i, denoted $consumerset_i$, as: $consumerset_i = \{c_j | j \in [i...(i+f)\ mod\ N]\}$. The intuition behind this function is that the consumers are seen as a circular space where each producer is responsible for sending the value it has computed to a set of consecutive consumers of cardinality $f+1$, which are shifted from one another by one position.

We model the operation of the algorithm in *rounds.* The round of a process is increased as result of a *nextRound* event. The system is synchronous, so non-Byzantine processes have their clocks synchronized and the *nextRound* event occurs simultaneously in all of them. The synchrony of the system and reliability of the channels ensure that if in response to event $nextRound(n)$ a non-Byzantine process sends a message to another non-Byzantine process, that message is delivered to the destination before $nextRound(n+1)$ is triggered. This implies that *nextround* events are triggered periodically with a period greater than the worst case latency of communication channels. The algorithm executes in three rounds. In round 0, the producers send values or hashes to consumers. In round 1, consumers send certificates of reception to the trusted observer. In round 2, the trusted observer produces the evidence.

In round 0, a producer computes the hash of the value and signs it (lines 106-108). When the first round starts, it sends the value, its hash, and signature to the consumers in $consumerset_i$ (lines 111-113), but only the hash and signature to the remaining consumers (lines 114-116).

A consumer starts by waiting for signed values and hashes from producers in round 1 (lines 209 and 215). Each value, hash, and signature received is stored in an array named *values* (lines 214 and 219). If a node does not send the message it was supposed to in this round, or if the hash or signature are not valid, the entry in the values set for that producer remains with the special value $\perp$, which will serve to build a proof of misbehaviour for the TO (if $f+1$ consumers provide similar certificates).

When round 1 ends, the consumer picks the value v such that $hash(v)$ appears in more than f positions of the array (lines 222-223). There are at most f faulty producers in the system, thus there is at most one value that matches this condition. Then, the consumer prepares the *confirm* array to serve as a *certificate* that vouches for the correct or incorrect behaviour of all producers, and that simultaneously proves that it has received and picked the correct value as described below (lines 224-225). For each producer p_i, the consumer either stores in *confirm*: i) the received hash and corresponding signature (extracted from the values set) or ii) the special value $\perp$ when no data, or incorrect data, was received from that producer. The consumer then signs this data structure with its private key and sends it as a proof of reception to the trusted observer (lines 226-228). The consumer terminates by outputting the value (line 229).

The trusted observer waits for a certificate from each consumer in round 2 (line 306). The certificates are collected in an array called *evidence* (line 309). In the end, the trusted observer produces the array as evidence (line 311).

Considering the data structure that is created by the trusted observer as evidence, we can now define with more detail the predicates *hasProduced* and *hasAcknowledged*. Let $h(v)$ denote the hash of the value v and let $s_{p_k}(h(v))$ denote the hash of v signed by the producer p_k:

- *hasProduced(evidence, p_i)* is true if the following condition holds: there are at least $N-f$ consumers $c_k \in \mathcal{C}$: $evidence[c_k][p_i] = \langle h(v), s_{p_i}(h(v))\rangle$. It is false otherwise.
- *hasAcknowledged(evidence, c_j)* is true if exists a set of producers, named $correctset_j$, such that $|correctset_j| \geq N-f$ and for $\forall p_k \in correctset_j$ *hasProduced*$(evidence, p_k)$ is true and $evidence[c_j][p_k] = \langle h(v), s_{p_k}(h(v))\rangle$. It is false otherwise.

The algorithm does not require the observer to actively participate in the execution of the algorithm. Furthermore, the verification process performed by the trusted observer is independent for each transfer. Therefore many instances of BAR-Transfer can be executed in parallel under the jurisdiction of one or

Algorithm 1: BAR-Transfer Algorithm

```
producer $p_i$:
101   upon init do
102     myvalue := $\perp$;
103     myhash :=$\perp$;
104     myhashsig := $\perp$;
105     round := 0;
106   upon produce($p_i$,myvalue) $\wedge$ round = 0 do
107     myhash := hash(myvalue);
108     myhashsig := sign ($p_i$, myhash);
109   upon nextRound $\wedge$ round = 0 do // start of round 1
110     round := 1;
111     msgsig := sign ($p_i$, VALUE || myvalue || myhash || myhashsig);
112     forall $c_j \in consumerset_i$ do
113       send ($p_i$, $c_j$, [VALUE, myvalue, myhash, myhashsig, msgsig])
114     msgsig := sign ($p_i$, SUMMARY || myhash || myhashsig);
115     forall $c_j \in \mathcal{C} \setminus consumerset_i$ do
116       send ($p_i$, $c_j$, [SUMMARY, myhash,myhashsig, msgsig])

consumer $c_j$:
201   upon init do
202     myvalue :=$\perp$;
203     myhash:=$\perp$;
204     confirm := $[\perp]^P$;
205     values := $[\perp]^P$;
206     round := 0;
207   upon nextRound $\wedge$ round = 0 do // start of round 1
208     round := 1;
209   upon deliver ($p_i$, $c_j$, [VALUE, pvalue, phash, phashsig, msgsig]) $\wedge$ round = 1 do
210     if ($c_j \in consumerset_i$)then
211       if verifysig($p_i$, VALUE || pvalue || phash || phashsig, msgsig)then
212         if verifysig($p_i$,phash, phashsig) then
213           if verifyhash(pvalue, phash) then
214             values[$p_i$] := $\langle$pvalue, phash, phashsig$\rangle$;
215   upon deliver ($p_i$, $c_j$, [SUMMARY, phash, phashsig, msgsig]) $\wedge$ round = 1 do
216     if ($c_j \notin consumerset_i$)then
217       if verifysig($p_i$, SUMMARY ||phash || phashsig, msgsig) then
218         if verifysig($p_i$, phash, phashsig) then
219           values[$p_i$] := $\langle \perp$, phash, phashsig$\rangle$;
220   upon nextRound $\wedge$ round = 1 do // start of round 2
221     round := 2;
222     myhash := $h : \#(\{p | value[p] = \langle *, h, * \rangle\}) > f$.
223     myvalue := $v : \{p | value[p] = \langle v, myhash, * \rangle\}$.
224     forall $p_i$: values[$p_i$] = $\langle$*, myhash, *$\rangle$ do
225       confirm[$p_i$] := $\langle$values[$p_i$].hash, values[$p_i$].signature$\rangle$;
226     confsig := sign ($c_j$, confirm);
227     msgsig := sign ($c_j$, CERTIFICATE||confirm||confsig);
228     send ($c_j$, TO, [CERTIFICATE, confirm, confsig, msgsig])
229     consume ($c_j$, myvalue);

trusted observer $TO$:
301   upon init do
302     evidence:= $[\perp]^C$;
303     round := 0;
304   upon nextRound $\wedge$ round $<$ 2 do
305     round := round+1;
306   upon deliver ($c_j$, TO, [CERTIFICATE, confirm, confsig, msgsig]) $\wedge$ round = 2 do
307     if verifysig ($c_j$, CERTIFICATE||confirm||confsig, msgsig) then
308       if verifysig ($c_j$, confirm, confsig) then
309         evidence[$c_j$] := $\langle$confirm, confsig$\rangle$;
310   upon nextRound $\wedge$ round = 2 do // start of round 3
311     certify (TO, evidence);
```

more trusted observers, without the trusted entity being a single point of failure or a bottleneck.

5 Analysis

The analysis of the algorithm has three parts. First, we prove its correctness. Then, we demonstrate that it is a Nash equilibrium. Finally, we perform a complexity analysis in terms of communication costs.

5.1 Correctness

This section provides a proof of the correctness of the algorithm, i.e., that it satisfies the properties BAR-Transfer 1-7. The proof assumes that at most f producers and f consumers are Byzantine and that the rest of the processes follow the algorithm, i.e., are Altruistic. The case of Rational processes is left for Section 5.2, in which we show that Rational processes also follow the algorithm.

We now show with the following Lemmas that the algorithm presented in Section 4, satisfies each of the BAR-Transfer properties.

Lemma 1. (Validity) *If a non-Byzantine consumer consumes v, then v was produced by some non-Byzantine producer.*

Proof. A non-Byzantine consumer c consumes v only if it receives a $hash(v)$ from at least $f + 1$ producers and v from at least one producer. There are at most f Byzantine producers, which implies that c receives $hash(v)$ from at least a non-Byzantine producer p_i. Thus, c consumes v only if v was input by p_i.

Lemma 2. (Integrity) *No non-Byzantine consumer consumes more than once.*

Proof. A consumer consumes a value when the *consume* primitive is called. A trivial inspection of the algorithm shows that this primitive can be called only once in a non-Byzantine consumer, thus it consumes the value no more than once.

Lemma 3. (Agreement) *No two non-Byzantine consumers consume different values.*

Proof. By Lemma 1, if a non-Byzantine consumer consumes v, then v was produced by some non-Byzantine producer. By assumption, every non-Byzantine producer produces the same value. Therefore, non-Byzantine consumers never deliver a value different from v.

Lemma 4. (Termination) *Every non-Byzantine consumer consumes a value.*

Proof. All the non-Byzantine producers produce and send v or its hash to all the consumers in the beginning of round 1. Given that channels are reliable and synchronous, all non-Byzantine consumers receive these values in that round. Therefore, when round 2 begins, every non-Byzantine consumer must possess both the correct value and $f + 1$ or more hashes of that value, so it executes the *consume* primitive which means consuming v.

These four properties ensure the reliable transfer of the correct value in the presence of Byzantine participants. In the following Lemmas, we prove that the properties BAR-Transfer 5-7 related to Rational behaviour are fulfilled, therefore ensuring that each node that obeys the protocol is rewarded after the completion of the transfer.

Lemma 5. (Evidence) *The trusted observer produces evidence about the transfer.*

Proof. A trivial inspection of the algorithm shows that *certify*(TO, *evidence*) is executed at the end of round 2, which is the same as saying the trusted observer produces evidence.

Lemma 6. (Producer Certification) *If producer p is non-Byzantine, then hasProduced(evidence, p) is true.*

Proof. By Lemmas 3 and 4, every non-Byzantine consumer delivers the same value v. Before delivering these consumers send their *confirm* vectors to the trusted observer. Therefore, there are at least $N - f$ non-Byzantine consumers $c_k \in \{c_1 \dots c_{N-f}\}$ that send confirm vectors to the trusted observer at the start of round 2. If producer p_i followed the algorithm, each of these consumers c_k has received *hash*(v) from p_i, and included $\langle h(v), s_{p_i}(h(v))\rangle$ in the message sent to the trusted observer. Since all those messages are included in the evidence generated by the trusted observer, *hasProduced(evidence,* p_i*)* is true.

Lemma 7. (Consumer Certification) *If consumer c is non-Byzantine, then hasAcknowledged(evidence, c) is true.*

Proof. A non-Byzantine consumer sends its *confirm* vector to the trusted observer. Also, since there are at least $N - f$ non-Byzantine producers, consumer c_j includes $\langle h(v), s_{p_i}(h(v))\rangle$ for each of these non-Byzantine producers p_i in the *confirm* vector sent to the trusted observer. According to Lemma 6, there exists a set *correctset* of at least $N - f$ producers $p_k \in \{p_1 \dots p_{N-f}\}$ for which *hasProduced(evidence,*p_k*)* is true (the set of $N - f$ non-Byzantine producers). Therefore, *hasAcknowledged(evidence,* c_j*)* becomes true for any non-Byzantine consumer c_j.

Theorem 1. (Correctness) *If all non-Byzantine participants follow the protocol, then the provided algorithm solves the BAR-Transfer problem defined in terms of properties BAR-Transfer 1-7.*

Proof. The proof follows directly from Lemmas 1, 2, 3, 4, 5, 6, and 7.

5.2 Game Theoretic Analysis

To prove that the protocol provides a Nash equilibrium, we model the BAR-Transfer problem as a strategic game $\Gamma = (M, S_M, \boldsymbol{u})$, where $M = \mathcal{P} \bigcup \mathcal{C}$ is the set of players, S_M the set of all possible strategies, and $\boldsymbol{u}$ is a vector with the utility functions of all players.

Each player decides its *strategy* (or plan of action) once and it remains valid for all its actions during the execution of BAR-Transfer. These decisions about the strategy are made simultaneously and, as Rational players do not collude among themselves, without knowledge of the strategies selected by other players. The set of possible strategies for player i is denoted S_i. S_M consists on the set of all possible strategies, i.e., of S_i for all $i \in M$. The set of all possible strategies of *producers* is S_P. Altruistic producers send $hash(v)$ to all consumers and the value v to the consumers of $consumerset_i$. Rational producers send $hash(v)$ to any subset of $\mathcal{C}$ and the value to any subset $\mathcal{C}' \subseteq \mathcal{C}$. Similarly, $S_{\mathcal{C}}$ denotes the set of all possible strategies that can be followed by *consumers*. Altruistic consumers process all the information received from producers, send it to the TO, and consume one value. Rational consumers may or may not: consume a value, process all the values or hashes received from producers, and send the received information to the TO. Byzantine players follow an arbitrary strategy from $S_{\mathcal{F}}$, where $\mathcal{F} = \mathcal{F}_{\mathcal{P}} \cup \mathcal{F}_{\mathcal{C}}$ is the set of all Byzantine producers ($\mathcal{F}_{\mathcal{P}}$) and Byzantine consumers ($\mathcal{F}_{\mathcal{C}}$), such that $|\mathcal{F}_{\mathcal{P}}| \leq f$ and $|\mathcal{F}_{\mathcal{C}}| \leq f$. Notice that these are pure strategies, that is, the decisions about which strategy to follow is deterministic.

We now identify the reasons why modelling the BAR-Transfer problem as a strategic game is not a limitation of our analysis. Strategic games are appropriate for interactions between players where a player cannot form his expectation from the behaviour of the other players on the basis of information about the way that the game was played in the past. The information gathered by each process regarding the past behaviour of other processes is determined by the number of instances of BAR-Transfer in which those processes interacted, which depends on the mechanism used to form the sets of processes in each instance. This mechanism must ensure that with high probability the number of Byzantine processes of each set is upper bounded by f. Furthermore, in a large peer-to-peer network, it is true that N is much smaller than the total number of processes in the system, and the processes connected to the network during the periods when that mechanism is applied vary from instance to instance. Thus, it is reasonable to assume that processes interact with a very small frequency, which implies that the information of each process regarding the nature of other processes is limited and never certain. Since processes do not incur in risks, they cannot form their expectation from the behaviour of the other players on the basis of information about the way that the game was played in the past. Therefore, it is reasonable to model our solution as a strategic game.

In addition, it is only adequate to model a protocol as a strategic game if players do not change their strategy during the execution of the game, which is true in our protocol. Producers cannot increase their knowledge of the strategies of other players during the execution of the algorithm, as they do not obtain any information from any other participant. Thus, the initial chosen strategy remains adequate for the three rounds of the protocol. On the other hand, each consumer c_j learns about the behaviour of producers in round 1, so it can determine which producers adopted the strategy of sending it the value or its hash. However, according to the definition of *hasAcknowledged*, the TO rewards c_j based not only on the information included in the certificate sent by the consumer, but also based on the information the producers (certified by c_j) sent to the remaining consumers. Therefore, the information gathered by c_j is insufficient for the consumer to determine if an alternative strategy provides greater expected profits. For these reasons, it is reasonable to assume that Rational participants do not change their strategy during the execution of the protocol.

We define a *profile of strategies* as the correspondence between players and their respective strategy: $\boldsymbol{\sigma}_M : M \mapsto S_M$. By definition, σ_i denotes the strategy followed by player $i \in M$. We define $\boldsymbol{\sigma}_M$ as the composition of different profile strategies for disjoint subsets of players $M_1, M_2, ..., M_N$: $\boldsymbol{\sigma}_M = (\boldsymbol{\sigma}^1_{M_1}, \boldsymbol{\sigma}^2_{M_2}, ..., \boldsymbol{\sigma}^N_{M_N})$, where $\boldsymbol{\sigma}^i_{M_i}$ is the strategy followed by all players of M_i and $M = M_1 \cup M_2 \cup \ldots \cup M_N$.

We also define an *utility function* $u_i(\boldsymbol{\sigma}_M) = \beta_i(\boldsymbol{\sigma}_M) - \nu_i(\boldsymbol{\sigma}_M)$ as the profit that player i obtains when all the players follow the strategy specified by $\boldsymbol{\sigma}_M$. β_i denotes the benefit obtained by player i. It is assumed that a producer p gets a benefit of ϕ_P only if *hasProduced(evidence, p)* holds true. Otherwise, the benefit is 0. A consumer c only gets a benefit ϕ_C if it consumes the correct value v (therefore, all non-Byzantine consumers consume the correct value) and if *hasAcknowledged(evidence, c)* holds true. The function $\nu_i(\boldsymbol{\sigma}_M)$ maps the costs incurred by player i when every player follows the strategies specified by $\boldsymbol{\sigma}_M$. We assume that $\phi_P > \nu_p(\boldsymbol{\sigma}_M)$ and $\phi_C > \nu_c(\boldsymbol{\sigma}_M)$ for any non-Byzantine producer p and consumer c, respectively. To distinguish the arbitrary behaviour of Byzantine players from the strategies of Altruistic and Rational players, we denote by $\boldsymbol{\pi}_{\mathcal{P}} \in \mathit{\Pi}_{\mathcal{P}}$ the profile of strategies of Byzantine producers and by $\boldsymbol{\pi}_{\mathcal{C}} \in \mathit{\Pi}_{\mathcal{C}}$ the profile of strategies of Byzantine consumers.

The remaining of this section provides a proof that the protocol provides a Nash equilibrium. In the BAR model, Rational players also take into consideration Altruistic and Byzantine behaviour [2]. A utility function for Rational player i that considers Byzantine, Altruistic and Rational behaviour, denoted by $\bar{u}_i$, is the expected utility for i if it obeys a given Rational strategy σ_i when all the remaining participants either obey a non-Byzantine strategy specified by the profile $\boldsymbol{\sigma}_M$ (that includes the Altruistic strategy of following the protocol) or follow a Byzantine strategy specified by the profile $\boldsymbol{\pi}_{\mathcal{F}}$. Given that Byzantine participants may behave arbitrarily, in the definition of the expected utility function it is necessary to consider not only the expected number of Byzantine players but also the probability of each Byzantine player following each of the possible Byzantine strategies. In this work, we assume that players are risk-averse, therefore the expected utility considers the worst possible scenario of Rational and Byzantine behaviour, i.e., it assumes that all non-Byzantine players are Rational and all Byzantine players adopt a strategy that minimizes the utility of non-Byzantine players.

Hereupon, we provide a definition for the *expected utility* $\bar{u}_i(\boldsymbol{\sigma}_M)$ of player $i \in M$ when all Rational players follow the strategy specified by $\boldsymbol{\sigma}_M$. Let $\boldsymbol{\sigma}'_{M\setminus\mathcal{F},\boldsymbol{\pi}_{\mathcal{P}},\boldsymbol{\pi}_{\mathcal{C}}} = (\boldsymbol{\sigma}_{M\setminus\mathcal{F}}, \boldsymbol{\pi}_{\mathcal{P}}, \boldsymbol{\pi}_{\mathcal{C}})$ be a profile of strategies where no player in M is Altruistic, all Rational players follow the strategy specified by $\boldsymbol{\sigma}_{M\setminus\mathcal{F}}$, Byzantine producers follow the strategy specified by $\boldsymbol{\pi}_{\mathcal{P}}$, and Byzantine consumers follow the strategy specified by $\boldsymbol{\pi}_{\mathcal{C}}$. The expected utility of player i is given by the following equation:

$$\bar{u}_i(\boldsymbol{\sigma}_M) = \min_{\mathcal{F}_{\mathcal{P}}:|\mathcal{F}_{\mathcal{P}}|\leq f, \mathcal{F}_{\mathcal{C}}:|\mathcal{F}_{\mathcal{C}}|\leq f} \circ \min_{\boldsymbol{\pi}_{\mathcal{P}}\in \Pi_{\mathcal{P}}, \boldsymbol{\pi}_{\mathcal{C}}\in \Pi_{\mathcal{C}}} u_i(\boldsymbol{\sigma'}_{M\setminus\mathcal{F}, \boldsymbol{\pi}_{\mathcal{P}}, \boldsymbol{\pi}_{\mathcal{C}}}) \tag{1}$$

Notice the distinction between the expected utility $\bar{u}_i(\boldsymbol{\sigma}_M)$, which denotes the minimum utility Rational player i expects to obtain when all Rational participants follow the strategy specified by $\boldsymbol{\sigma}_M$, and the effective utility $u_i(\boldsymbol{\sigma'}_{M\setminus\mathcal{F}, \boldsymbol{\pi}_{\mathcal{P}}, \boldsymbol{\pi}_{\mathcal{C}}})$, which is the difference between the benefits obtained and the costs incurred by i when Byzantine players follow the specific strategies specified by $\boldsymbol{\pi}_{\mathcal{P}}$ and $\boldsymbol{\pi}_{\mathcal{C}}$.

We can now define the functions $\bar{\beta}_i(\boldsymbol{\sigma}_M)$ and $\bar{\nu}_i(\boldsymbol{\sigma}_M)$ as the expected benefits and costs for the worst possible scenario of Rational and Byzantine behaviour. Thus, the expected utility of player $i \in M$ can also be defined as $\bar{u}_i(\boldsymbol{\sigma}_M) = \bar{\beta}_i(\boldsymbol{\sigma}_M) - \bar{\nu}_i(\boldsymbol{\sigma}_M)$.

We now introduce the notion of *Nash equilibrium*. Let $\boldsymbol{\sigma}^*_{M\setminus\{i\},\sigma^*_i} = (\boldsymbol{\sigma}_{M\setminus\{i\}}, \sigma^*_i)$ denote the profile of strategies where all Rational players follow the strategy specified by $\boldsymbol{\sigma}_{M\setminus\{i\}}$ and player i follows a given strategy σ^*_i. A Nash equilibrium is a profile of strategies for which no player benefits from deviating from its strategy, which can be stated as follows:

Definition 1. *$\boldsymbol{\sigma}_M$ is a Nash equilibrium if* $\forall_{i\in M}\forall_{\sigma^*_i\in S_i}\bar{u}_i(\boldsymbol{\sigma}_M) \geq \bar{u}_i(\boldsymbol{\sigma}^*_{M\setminus\{i\},\sigma^*_i})$.

The following Lemmas provide the complete proof that neither the producers nor the consumers benefit from deviating from the protocol. We use $\boldsymbol{\sigma}_{\mathcal{P}}$ and $\boldsymbol{\sigma}_{\mathcal{C}}$ to denote the profile of strategies of, respectively, producers and consumers that comply with the protocol. $\boldsymbol{\sigma}_M$ denotes the composition of the profiles of strategies $\boldsymbol{\sigma}_{\mathcal{P}}$ and $\boldsymbol{\sigma}_{\mathcal{C}}$, and $\boldsymbol{\sigma}^*_M$ denotes an alternative profile of strategies.

In the next Lemma and Corollary, we show that a producer does not benefit from sending the expected information to less than N consumers and from not sending the value to all consumers of *consumerset*. Then, in Theorem 2, we show that no producer can increase its utility by deviating from the protocol, when all consumers follow the expected strategy.

Lemma 8. *For each producer $p \in \mathcal{P}$, for each k such that $0 \leq k < N$, let $\boldsymbol{\sigma}^*_M = (\boldsymbol{\sigma}_{\mathcal{P}\setminus\{p\}}, \boldsymbol{\sigma}_{\mathcal{C}}, \sigma^*_p)$ be a deviating profile of strategies, where σ^*_p is the strategy of sending a value or its signature to k consumers. Then, $\bar{\beta}_p(\boldsymbol{\sigma}^*) = 0$.*

Proof. According to the Equation 1, Rational players determine their utility considering the worst case scenario of Byzantine and Rational behaviour. Suppose the set of k consumers to which p sends the information includes all the Byzantine players. According to the protocol, the trusted observer only receives vectors containing signed hashes in the second round. Hence, if p only sends the signature of the value or its hash to $k < N$ consumers at the beginning of the first round and if no Byzantine consumer sends their vectors to the trusted observer, the trusted observer only receives $max(k - f, 0) < N - f$ vectors that contain $\langle h(v), s_p(h(v))\rangle$ in the second round. Therefore, *evidence* will not contain $N - f$ entries with $\langle h(v), s_p(h(v))\rangle$, *hasProduced(evidence, p)* will hold false, and $\bar{\beta}_p(\boldsymbol{\sigma}^*) = 0$.

Corollary 1. *For each producer $p_i \in \mathcal{P}$, for each k such that $0 \leq k < f+1$, let $\boldsymbol{\sigma}^*_M = (\boldsymbol{\sigma}_{\mathcal{P}\setminus\{p_i\}}, \boldsymbol{\sigma}_{\mathcal{C}}, \sigma^*_{p_i})$ be a deviating profile of strategies, where $\sigma^*_{p_i}$ is the strategy of sending the value to k consumers. Then, $\bar{\beta}_{p_i}(\boldsymbol{\sigma}^*) = 0$.*

Proof. The proof comes trivially from the previous lemma.

Theorem 2. *No producer has any incentives to deviate from the protocol.*

Proof. It follows from Lemma 8 and Corollary 1 that if the producer p follows an alternative strategy specified by $\boldsymbol{\sigma}^*_M$, then $\bar{\beta}_p(\boldsymbol{\sigma}^*_M) = 0$, $\bar{u}_p(\boldsymbol{\sigma}^*_M) = -\bar{\nu}_p(\boldsymbol{\sigma}^*_M)$, and $\bar{u}_p(\boldsymbol{\sigma}^*_M) < 0$, for the worst possible scenario. According to the Theorem 1, $\bar{\beta}_p(\boldsymbol{\sigma}_M) = \phi_P$, $u_p(\boldsymbol{\sigma}_M) = \phi_P - \bar{\nu}_p(\boldsymbol{\sigma}_M)$, and $\bar{u}_p(\boldsymbol{\sigma}_M) > 0$, since $\phi_P > \bar{\nu}_p(\boldsymbol{\sigma}_M)$. Therefore, $\bar{u}_p(\boldsymbol{\sigma}_M) > \bar{u}_p(\boldsymbol{\sigma}^*_M)$. Since it is assumed that Rational participants are risk-averse, producers do not have incentives to deviate from the protocol.

We now show that no consumer benefits either by not sending the *confirm* vector to the *TO* or by not processing all the information it receives from the producers. Then, in Theorem 4, we prove that no consumer can increase its utility by deviating from the protocol, given that producers follow the expected behaviour.

Lemma 9. *For any consumer $c \in \mathcal{C}$, let $\boldsymbol{\sigma}^*_M = (\boldsymbol{\sigma}_{\mathcal{P}}, \boldsymbol{\sigma}_{\mathcal{C}\setminus\{c\}}, \sigma^*_c)$ be a deviating profile of strategies, where σ^*_c is the strategy of not sending its vector containing hashes sent by producers to the trusted observer in round 2. Then, $\bar{\beta}(\boldsymbol{\sigma}^*_M) = 0$.*

Proof. The proof derives directly from that fact that, if a consumer c does not send its vector, this information is not included in the evidence and *hasAcknowledged(evidence, c)* hods false. Hence, $\bar{\beta}_c(\boldsymbol{\sigma}^*_M) = 0$.

Lemma 10. *For any consumer $c \in \mathcal{C}$, let P_c be the set of producers that sent the correct value or hash to the consumer c, and let $\boldsymbol{\sigma}^*_M = (\boldsymbol{\sigma}_{\mathcal{P}}, \boldsymbol{\sigma}_{C\setminus\{c\}}, \sigma^*_c)$ be a deviating profile of strategies, where σ^*_c is the strategy of sending an incomplete vector of hashes to the trusted observer with only $f+1 \leq k < |P_c|$ entries different from the $\perp$ value. Then, $\bar{\beta}_c(\boldsymbol{\sigma}^*_M) = 0$.*

Proof. The worst possible scenario for a non-Byzantine consumer c_j occurs when $|F_{\mathcal{P}}| = f$ and for all these Byzantine producers *hasProduced* is *false*, while they still send valid information to c_j. In this case, there is only one set *correctset*$_j$, where, for all $p \in$ *correctset*$_j$, *hasProduced(evidence,p)* is true: the set of non-Byzantine producers. If c_j does not set $hashes[p_i] = \langle hash(v), s_{p_i}(hash(v))$ and p_i is non-Byzantine, then, at the trusted observer, $evidence[c_j]$ will not contain the information of at least $N - f$ producers from *correctset*$_j$. Therefore, *hasAcknowledged(evidence,c)* holds *false*, and $\bar{\beta}_c(\boldsymbol{\sigma}^*_M) = 0$.

Theorem 3. *No consumer has any incentives to deviate from the protocol.*

Proof. It follows from Lemmas 9 and 10 that if the consumer c follows an alternative strategy specified by $\boldsymbol{\sigma}^*_M$, then $\bar{\beta}_c(\boldsymbol{\sigma}^*_M) = 0$, $\bar{u}_c(\boldsymbol{\sigma}^*_M) = -\bar{\nu}_c(\boldsymbol{\sigma}^*_M)$, and $\bar{u}_c(\boldsymbol{\sigma}^*_M) < 0$, for the worst possible scenario. According to the Theorem 1, $\bar{\beta}_c(\boldsymbol{\sigma}_M) = \phi_C$, $\bar{u}_c(\boldsymbol{\sigma}_M) = \phi_C - \bar{\nu}_c(\boldsymbol{\sigma}_M)$, and $\bar{u}_c(\boldsymbol{\sigma}_M) > 0$, since $\phi_C > \bar{\nu}_c(\boldsymbol{\sigma}_M)$. Therefore, $\bar{u}_c(\boldsymbol{\sigma}_M) > \bar{u}_c(\boldsymbol{\sigma}^*_M)$. Since it is assumed that Rational participants are risk-averse, consumers do not have any incentive to deviate from the protocol.

The following Theorem concludes that the protocol provides a Nash equilibrium.

Theorem 4. ***(Nash equilibrium)*** *The profile of strategies $\boldsymbol{\sigma}_M$ where every player follows the protocol is a Nash equilibrium.*

Proof. It follows from Theorems 2 and 4 that for every player $i \in M$ and, for all alternative profiles of strategies $\boldsymbol{\sigma}^*_M$ where i deviates from the protocol, $\bar{u}_i(\boldsymbol{\sigma}^*_M) < \bar{u}_i(\boldsymbol{\sigma}_M)$. Hence, $\boldsymbol{\sigma}_M$ is a Nash equilibrium.

From the previous proofs, it is possible to observe that our solution is a dominant strategy, that is, any other Nash equilibrium has a utility lower than the utility that each Rational process expects to obtain when following our solution. Hence, Rational processes should obey our algorithm.

5.3 Complexity Analysis

This section briefly evaluates the algorithm in terms of time, message, and bit complexity. The time complexity is the number of rounds for termination and in this case is constant: 3 rounds. For the other two we consider the case in which all processes follow the protocol. The message complexity, i.e., the number of messages sent by the algorithm, is $N^2 + N$, or $O(N^2)$. The bit complexity, i.e., the number of bits sent, is $O(Nfl_v + N^2 l_s)$, where l_v is the bit length of the value and l_s the bit length of a signature, assuming that $3l_s \gg 2l_h$, where l_h is the bit length of an hash.

6 Conclusions

In this paper we have introduced the BAR-Transfer problem that abstracts the problem of transferring data from a set of producers to a set of consumers under the BAR system model. We have presented an algorithm that solves the BAR-Transfer problem for $N \geq 2f + 1$, where N is the number of producers and consumers. We have shown that our algorithm is a Nash equilibrium, so Rational participants are unable to extract any benefit from deviating from the algorithm. BAR-Transfer is a powerful construct to build peer-to-peer systems that support distributed storage and parallel processing based on volunteer nodes. We are building such a system, based on a P2P architecture, which aims at supporting distributed computations using the MapReduce model.

Acknowledgment: This work was partially supported by the FCT (INESC-ID multi annual funding through the PIDDAC Program fund grant and by the project PTDC/EIA-EIA/102212/2008).

References

1. Abraham, I., Dolev, D., Gonen, R., Halpern, J.: Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation. In: PODC'06. pp. 53–62. Denver, USA (Jul 2006)

2. Aiyer, S., Alvisi, L., Clement, A., Dahlin, M., Martin, J.P., Porth, C.: BAR fault tolerance for cooperative services. In: SOSP'05. pp. 45–58. Brighton, United Kingdom (Oct 2005)
3. Anderson, D.: Boinc: A system for public-resource computing and storage. In: GRID'04. pp. 4–10. Pittsburgh, USA (Nov 2004)
4. Anderson, D., Cobb, J., Korpela, E., Lebofsky, M., Werthimer, D.: SETI@home: an experiment in public-resource computing. Communications of the ACM 45(11), 56–61 (Nov 2002)
5. Axelrod, R.: The Evolution of Cooperation. Basic Books, New York (1984)
6. Baldoni, R., Helary, J.M., Raynal, M., Tanguy, L.: Consensus in Byzantine asynchronous systems. J. Discrete Algorithms 1(2), 185–210 (2003)
7. Canetti, R., Rabin, T.: Fast asynchronous Byzantine agreement with optimal resilience. In: STOC'93. pp. 42–51. New York, USA (1993)
8. Castro, M., Liskov, B.: Practical Byzantine fault tolerance and proactive recovery. ACM Transactions on Computer Systems 20(4), 398–461 (2002)
9. Clement, A., Napper, J., Li, H., Martin, J.P., Alvisi, L., Dahlin, M.: Theory of bar games. In: PODC'07. pp. 358–359. Portland, USA (Aug 2007)
10. Correia, M., Neves, N.F., Lung, L.C., Verissimo, P.: Low complexity Byzantine-resilient consensus. Distributed Computing 17(3), 237–249 (2005)
11. Dolev, D., Strong, H.: Authenticated algorithms for Byzantine agreement. SIAM J. Comput. 12(4), 656–666 (1983)
12. Dwork, C., Lynch, N., Stockmeyer, L.: Consensus in the presence of partial synchrony. J. of ACM 35, 288–323 (April 1988)
13. Eliaz, K.: Fault tolerant implementation. Review of Economic Studies 69(3), 589–610 (2002)
14. Fraigniaud, P.: Asymptotically optimal broadcasting and gossiping in faulty hypercube multicomputers. Computers, IEEE Transactions on 41(11), 1410–1419 (Nov 1992)
15. Hardin, G.: The tragedy of the commons. Science 162(3859), 1243–47 (1968)
16. Keidar, I., Melamed, R., Orda, A.: Equicast: Scalable multicast with selfish users. In: PODC06. pp. 63–71 (Jul 2006)
17. Kotla, R., Alvisi, L., Dahlin, M., Clement, A., Wong, E.: Zyzzyva: speculative Byzantine fault tolerance. In: SOSP'07. pp. 45–58. Stevenson, USA (Oct 2007)
18. Lamport, L.: The part-time parliament. ACM Trans. on Computer Systems 16(2), 133–169 (May 1998)
19. Lamport, L., Shostak, R., Pease, M.: The Byzantine generals problem. ACM Trans. Program. Lang. Syst. 4, 382–401 (Jul 1982)
20. Lee, S., Shin, K.: Interleaved all-to-all reliable broadcast on meshes and hypercubes. Parallel and Distributed Systems, IEEE Transactions on 5(5), 449–458 (May 1994)
21. Li, H., Clement, A., Marchetti, M., Kapritsos, M., Robison, L., Alvisi, L., Dahlin, M.: Flightpath: Obedience vs choice in cooperative services. In: OSDI'08. San Diego, USA (Dec 2008)
22. Li, H., Clement, A., Wong, E., Napper, J., Roy, I., Alvisi, L., Dahlin, M.: BAR gossip. In: OSDI'06. pp. 191–204. Seattle, USA (Nov 2006)
23. Malkhi, D., Reiter, M.: Byzantine quorum systems. In: STOC'97. pp. 569–578. El Paso, USA (1997)
24. Martin, J.P., Alvisi, L., Dahlin, M.: Minimal Byzantine storage. In: DISC'02. pp. 311–325. Toulouse, France (Oct 2002)
25. Martin, O., Ariel, R.: A Course in Game Theory. MIT Press (1994)
26. Wong, E.L., Leners, J.B., Alvisi, L.: It's on me! the benefit of altruism in BAR environment. In: DISC'10. pp. 406–420. Cambridge, USA (Sep 2010)