Leveraging Altruism in Cooperative Services

Jean-Philippe Martin

June 20, 2007

Abstract

When there is no central administrator to control the actions of nodes in a distributed system, the users may deviate for personal gain. The BAR model describes the three types of nodes in these environments: Byzantine nodes deviate arbitrarily, Altruistic nodes follow the protocol, and Rational nodes deviate for gain.

Previous attempts at writing BAR-Tolerant protocols have relied on a form of policing that makes sure that all non-Byzantine nodes follow the same protocol. This method makes it impossible to take advantage of the Altruistic nodes.

In this paper we introduce a new equilibrium concept that does not rely on such strong policing. As a result, we can leverage Altruistic nodes when they are present. We give a condition that is sufficient to ensure that protocol guarantees hold despite Rational and Byzantine nodes, and we show how to use the new equilibrium concept and condition to build a BAR-Tolerant Terminating Reliable Broadcast protocol. Our protocol leverages Altruistic nodes to reduce the amount of extra work that Byzantine nodes can inflict on the system.

1 Introduction

Fault-tolerance approaches that were appropriate for clusters of computers are not appropriate for *cooperative services*,[1] peer-to-peer systems where there is no central administrator controlling all of the nodes in the system. Freed from the oversight of a central administrator, the humans (or *users*) can interfere with the configuration of their computers or even replace the software in order to maximize their benefit or minimize their costs. This cause of deviation is not a concern for clusters of computers because, there, the person controlling the computers wants them to run the software without deviation.

The behavior of maximizing one's benefit (potentially at the expense of others) that we find in cooperative services has been investigated in other contexts by economists for some time [18, 25, 34]; it also appears in game theory [16] where it is called *rational.* Rational behavior has been observed in distributed computer systems, in the context of network congestion [19] and "free riding" [2, 20] on file-sharing systems [17, 24]. For example, 66% of users on the Gnutella network share no file at all. Rational behavior can lead to the well-known *tragedy of the commons* [18, 25], in which the actions best for individuals are detrimental to the system as a whole.

Cooperative services should not be modeled using the Byzantine model [21] because many problems of interest cannot be solved when all nodes are Byzantine (such as consensus in eventual synchrony [12] or with a fair scheduler [7]), yet in cooperative services it is conceivable that *every* node will deviate from the protocol because of the actions of the rational users that are controlling them. Cooperative services should not be described only in terms of rational utility-maximizing nodes either [18, 25, 30, 34], because although this approach handles rational behavior it is brittle in the face of Byzantine failures. Since Byzantine deviations may go against a node's

[1]Other work also refers to them as MAD, for "multiple administrative domains".

best interest, they are not covered by this model. For example, the AS7007 incident [31]—where a misconfigured router announced that it was the best path to most of the Internet and disrupted global connectivity for over two hours—demonstrates the damage a single faulty node can cause in a system that is not Byzantine-tolerant.

The BAR model [5, 9, 29] is best adapted to cooperative services. Its name comes from the three types of nodes that it models: Byzantine, Altruistic, and Rational. Byzantine nodes may deviate arbitrarily from the protocol. Altruistic nodes follow the protocol, and Rational nodes only deviate from the protocol if that increases the utility they receive.

Rational nodes may not be willing to take part in computation steps that contribute to system performance if the utility that they get from the computation is less than the cost itself. So, optimization opportunities may be lost when nodes are Rational instead of Altruistic. This "price of anarchy" has been measured for the virus inoculation game [32]. Existing protocols for the BAR model (e.g. [22, 29]) can tolerate both Byzantine and Rational behavior, but they force Altruistic and Rational nodes to behave identically, thus paying the full price for anarchy even when Altruistic nodes are present.

In this paper we show how to design protocols that leverage altruistic behavior, if present. These protocols specify computation steps that Altruistic nodes will follow and Rational nodes may omit. As a result, protocol performance can be improved when nodes behave altruistically, even if the identity of the Altruistic nodes is not necessarily known.

The paper is organized along two main sections. After a quick reminder of the BAR model in Section 2, Section 3 focuses on the theoretical aspects and defines the BAR-Tolerance concept, describing the conditions under which Rational nodes may deviate without harming the system. Section 4 shows how to apply the concepts of the previous section to leverage altruism. We use terminating reliable broadcast to demonstrate our approach and present ATRB+, the first BAR-Tolerant protocol that leverages Altruistic nodes when they are present.

2 The BAR Model

The BAR model was developed by the authors of [5, 9, 29]. This section is a reminder of the model introduced in these earlier works.

Consider a set N of n nodes; each node knows N and can identify the nodes it communicates with (either through authenticated links or through shared secrets). Each node i is given a *suggested protocol* σ_i. Node i might follow a different protocol σ'_i for one of two reasons. First, since there is no central authority, the user controlling node i might replace σ_i with some other protocol σ'_i for its benefit. Second, some of the nodes may deviate from the protocol as a result of a hardware or software failure, or user mistake.

The Byzantine Altruistic Rational (BAR) model addresses these considerations by classifying nodes into the following three categories.

- *Byzantine* nodes may deviate arbitrarily from the suggested protocol for any reason. They may be misconfigured, compromised, malfunctioning, misprogrammed, or they may just be optimizing for an unknown utility function that differs from the utility function used by Rational nodes—for instance, by ascribing value to harm inflicted on the system or its users.

- *Altruistic* nodes follow their suggested protocol exactly. Intuitively, Altruistic nodes correspond to *correct nodes* in the fault-tolerance literature. Altruistic nodes may reflect the existence of Good Samaritans and "seed nodes" in real systems. However, in the BAR model, nodes that crash cannot be classified as Altruistic.[2]

[2]If needed, the BAR model could be expanded to allow for crashing Altruistic nodes.

- *Rational* nodes reflect self-interest and seek to maximize their benefit according to one of a known set U of utility functions. Rational nodes will deviate from the suggested protocol if and only if doing so increases their estimated utility from participating in the system. The utility function must account for a node's costs (e.g., computation cycles, storage, incoming and/or outgoing network bandwidth, power consumption, or threat of financial sanctions [23]) and benefits (e.g., access to remote storage [6, 10, 23, 28], network capacity [26], or computational cycles [36]) for participating in a system.

Nodes that deviate from the protocol (other than by crashing) have been considered Byzantine in the past. Tolerating Byzantine failures is costly and sometimes even infeasible, as we have seen. The BAR model allows us to model some of these deviating nodes using a stronger model in which it is possible to design protocols for situations where the Byzantine model would not apply (for example, when all nodes deviate because doing so is in their benefit). The BAR model is accurate [35] in the sense that rational behavior has been observed to take place, and we show that the BAR model is tractable in the sense that useful protocols can be designed for it.

Under BAR, the goal is to provide guarantees similar to those from Byzantine fault-tolerance to "all Rational and Altruistic nodes" as opposed to "all correct nodes." We distinguish two classes of protocols that meet this goal.

- *Incentive-Compatible Byzantine Fault-Tolerant* (IC-BFT) protocols: A protocol is IC-BFT if it tolerates Byzantine nodes and if it is in the best interest of all Rational nodes to follow the protocol exactly. An IC-BFT protocol therefore must define the optimal protocol for a Rational node.

- *Byzantine Altruistic Rational Tolerant* (BAR-Tolerant) protocols: A protocol is BAR-Tolerant if it tolerates Byzantine and Rational nodes, even if the Rational nodes deviate from the protocol. Note that all IC-BFT protocols are also BAR-Tolerant.

3 BAR-Tolerant Protocols

3.1 Overview

In our setup, each node i is given a protocol σ_i for consideration. We call σ_i node i's *suggested protocol*; Altruistic nodes will follow that protocol but we do not know a-priori what protocol the Rational and Byzantine nodes will decide to follow. Given a desired property P, our goal is to find a protocol σ that satisfies P when σ is given as the suggested protocol to each node in a cooperative service (or, more generally, to find a protocol profile $\vec{\sigma}$ with the same property; a protocol profile may assign a different protocol to each node).

The first step is to specify the beliefs of the Rational nodes. In this paper we assume that all Rational nodes believe that the other nodes may be Altruistic, Rational or Byzantine (but do not initially know which is which), and that there are at most f Byzantine nodes. We focus on synchronous protocols and consider Rational nodes that do not collude (colluding nodes are classified as Byzantine, so we can tolerate some collusion). We assume that Rational nodes only consider protocols that terminate[3] (we denote this set of protocols with Σ).

The second step is to specify what value the Rational nodes assign to events and possible future events. Events are valued using an utility function u that assigns a numerical score to an *outcome* of the protocol as seen by that particular node; the outcome is an execution trace, conceptually a history of the states that node was

[3]Recall that the protocol is repeated, so an infinite horizon game and terminating protocols are not mutually exclusive.

in. Outcomes with higher utility are preferred. Different nodes may have a different utility function, so we specify the set U of all utility functions that Rational nodes may have. For example some nodes may try to minimize their bandwidth costs while some other nodes may only care about their CPU costs. The value of possible future events is based on the utility function but it must also take into account the nodes' optimism or pessimism when comparing several possible future scenarios. Section 3.2 defines the *estimated utility* to quantify Rational nodes' outlook on the future.

These first two steps are common with previous methods for writing BAR protocols [5, 9, 29]. The next steps differ in that they allow us to leverage Altruistic nodes.

The third step is to pick a protocol profile $\vec{\sigma}$ and a rational envelope $\vec{\Upsilon}$. σ_i is the suggested protocol for node i, and Υ_i is a set of protocols ($\sigma_i \in \Upsilon_i$). The intention is that if i is Altruistic it follows σ_i and if it is Rational then it follows some protocol in Υ_i. A sufficient condition is that $\vec{\Upsilon}$ satisfies the BAR Equilibrium condition (Section 3.3).

The final step is to check whether $\vec{\sigma}$ and $\vec{\Upsilon}$ satisfy the BART property (Section 3.4) given the choice of U and estimated utility. We show in that section that if BART holds, then the goal is reached: property P holds when $\vec{\sigma}$ is the suggested protocol.

3.2 Estimating the Utility

As part of specifying the Rational nodes, one must codify how these nodes value possible future outcomes. This subsection is a reminder of the concept of estimated utility [5, 29, 9]. The estimated utility function $\hat{u}$ describes the utility that a given node estimates it will receive by continuing to follow a given protocol. The starting point for $\hat{u}$ is the utility function u: if a node i had sufficient information to determine the actions of every node—including the Byzantine nodes—then i could compute the future outcome and consequently directly compute its utility u_i for running the protocol e times (recall that u maps outcomes to real numbers). The outcome derives from the protocols that nodes follow and the utility derives from the outcome, so we can express the utility directly as a function of the protocols that nodes follow.

$$u_i(\vec{\sigma} \ominus \sigma'_i \ominus \phi_B \ominus \vec{v}_{N-A-B}, e) = u_i(\mathit{outcome}(\vec{\sigma} \ominus \sigma'_i \ominus \phi_B \ominus \vec{v}_{N-A-B}, e))$$

In the formula above, the protocol profile $\vec{\sigma}$ is a vector of length n that suggests a protocol to each node. The protocol profile ϕ_B is a vector of length n that assigns a protocol to each node of the set B; the remaining entries in the vector are $\perp$. B represents the set of Byzantine nodes. The operation $a \ominus b$ (read "a except b") returns a list of length n where every $\perp$ element of b is replaced by the corresponding element of a. Since A is the set of Altruistic nodes, the set $N-A-B$ contains all Rational nodes. Of course, a Rational node may not know ϕ_B or $\vec{v}_{N-A-B}$.

In our case, we consider *risk-averse* Rational nodes, meaning that the nodes aim to maximize the worst-case utility they may receive in response to their actions. The formula below formalizes our notion of worst-case for a node i based on the knowledge K_i that is available to i (this knowledge is expressed as a trace of the execution up to the current point, see [29] for more detail and an explanation of the need for K_i. See [9] for a generalization that can include non risk-averse Rational nodes).

$$\hat{u}_i(\vec{\sigma} \ominus \sigma'_i, K_i) = \underbrace{\min_{A \in \mathcal{A}(K_i)}}_{(1)} \underbrace{\min_{B \in \mathcal{B}(K_i)}}_{(2)} \underbrace{\min_{\vec{v} \in \vec{\Upsilon}}}_{(3)} \underbrace{\min_{\phi_B \in \Sigma(K_i)}}_{(4)} \underbrace{\lim_{e \to \infty} \frac{1}{e}}_{(5)} u_i(\vec{\sigma} \ominus \sigma'_i \ominus \phi_B \ominus \vec{v}_{N-A-B}, e)$$

The set $\mathcal{A}(K_i)$ is the set of possible assignments of altruistic nodes that is consistent with the observations in K_i (for example, if a node x

was observed to deviate from the suggested protocol then one can conclude that x is not Altruistic). Rational nodes are risk-averse with respect to which nodes are Altruistic (1), which nodes are Byzantine (2), the behavior of the Rational nodes within the rational envelope $\vec{\Upsilon}$ (3), and the behavior of the Byzantine nodes (4). Set $\Sigma(K_i)$ is the set of protocols that terminate and that are consistent with the observations in K_i. Term (5) allows us to compute the average utility over an infinite game. This compounding is necessary because the utility may be different for certain instances. For example, a node may only receive a benefit on some instances: this is the case in the Terminating Reliable Broadcast example that we discuss in Section 4, where nodes only receive a benefit when they have the role of sender.

The equation above shows how to compute the estimated utility function $\hat{u}$ from the utility function u. The set $\hat{U}$ of all the estimated utility functions that nodes can have is computed by applying this transformation to each member of U.

3.3 BAR Equilibrium

To leverage altruism, the suggested protocol must specify tasks that Rational nodes would not normally perform. An Altruistic node a follows its suggested protocol σ_a, but a Rational node r may instead choose to follow any of the protocols in its rational envelope Υ_r.

The goal of the BAR Equilibrium is to describe the conditions under which a Rational node r is guaranteed not to stray from its assigned Υ_r. The utility that r gets is influenced by the choice of protocol that other nodes follow. Since Rational nodes only deviate to increase their estimated utility, a sufficient condition for stability is that for each choice of protocol from the other Rational nodes, the protocol that maximizes r's utility lies in Υ_r. The definition below formalizes this notion.

Definition 1. *The pair $(\vec{\sigma}, \vec{\Upsilon})$ of the suggested protocol profile $\vec{\sigma}$ and the rational envelope $\vec{\Upsilon}$ is a* BAR Equilibrium *if and only if:*

$$\forall j, \forall \hat{u}_j \in \hat{U}, \forall K_j \in \mathcal{K}, \forall \sigma'_j \in \Sigma, \exists \sigma''_j \in \Upsilon_j : \\ \hat{u}_j(\vec{\sigma} \ominus \sigma'_j, K_j) \leq \hat{u}_j(\vec{\sigma} \ominus \sigma''_j, K_j)$$

If $(\vec{\sigma}, \vec{\Upsilon})$ is a BAR Equilibrium, then no Rational node has an incentive to unilaterally deviate from its rational envelope.

In some cases, a punishment mechanism can be used to make sure that the rational envelope is a BAR Equilibrium: punish nodes that are observed to deviate. Punishment would reduce the utility of the deviating node by, for example, assigning it more work or excluding it from the system. This punishment must be credible, in the sense that it must not reduce the utility of the punisher (otherwise the node observing the deviation has an incentive to deviate and omit the punishment). If the rational envelope has the property that all deviations from it are detectable (in the sense that nodes communicating with the deviating node can establish that it left the rational envelope), then punishment can be effective. Otherwise, punishment must be combined with careful protocol design to make sure that the deviations that cannot be observed do not increase the deviating node's estimated utility. It is best to choose a $\vec{\Upsilon}$ that reduces as much as possible the number of deviations that are not detectable, since each such deviation must then be shown not to increase the estimated utility.

3.4 BAR-Tolerance

We now have all the necessary concepts to define BAR-Tolerance. Informally, a protocol profile $\vec{\sigma}$ that provides some property P is BAR-Tolerant if P is maintained despite Byzantine and Rational deviation from the nodes. By Rational deviation of node i we mean a deviation from the suggested protocol σ_i that increases $\hat{u}_i$.

Definition 2. *Given a threshold f on the maximum number of Byzantine nodes and a threshold a on the minimum number of Altruistic nodes* *(the remainder nodes being Rational), and given the set $\hat{U}$ of estimated utility functions that Rational nodes use, we say that a protocol profile $\vec{\sigma}$ that provides some property P is* BAR-Tolerant *if there exists a rational envelope $\vec{\Upsilon}$ such that both of the following two conditions hold.*

1. $\forall B(|B| \leq f), \forall R(|R| \leq n - f - a), \forall \vec{v}_R \in \vec{\Upsilon}_R$*: P holds despite nodes in B being Byzantine, nodes in R following $\vec{v}_R$, and the remainder nodes following $\vec{\sigma}$.*

2. $(\vec{\sigma}, \vec{\Upsilon})$ *is a BAR Equilibrium.*

We say that the protocol σ is BAR-Tolerant if $\vec{\sigma} = (\sigma, \ldots, \sigma)$ is BAR-Tolerant. Note that if $\vec{\sigma}$ is IC-BFT [29] then $\vec{\sigma}$ is also BAR-Tolerant, for the rational envelope $\vec{x}$ that only contains $\vec{\sigma}$ (formally: $x_i = \{\sigma_i\}$). BAR-Tolerance is strictly stronger than IC-BFT.

Instead of using thresholds f and a to describe the number of Byzantine and Altruistic nodes we can adapt the more general notion of fail-prone systems [27] and define a *BAR-prone system* $\mathcal{R}$ as a set of triplets (B, A, R) such that B,A, and R partition the set N of nodes. For every run r of the system, there must exist an element $r \in \mathcal{R} = (B_r, A_r, R_r)$ such that all Byzantine nodes are in B_r, all Altruistic nodes are in A_r and all Rational nodes are in R_r. The (f, a)-threshold is a special case of the BAR-Prone system, where $\mathcal{R} = \{(B, A, R) : |B| \leq f \wedge |A| \geq a \wedge B \cup A \cup R = N \wedge |B| + |A| + |R| = |N|\}$. This generalization can describe more complex situations, for example when some nodes are more likely to fail than others.

4 Example: ATRB+

4.1 Model and Problem Statement

Having introduced a framework for reasoning about BAR-Tolerant protocols, we now demonstrate its use with ATRB+, a BAR-Tolerant protocol for Terminating Reliable Broadcast (TRB) that leverages Altruistic nodes to shift work away from busy nodes. The model is synchronous, so the execution is split into rounds where nodes first receive, and then send messages. All messages sent in a round are received on the next round. Messages can be unforgeably signed and unsigned messages can be sent on authenticated links. The protocol can tolerate f Byzantine nodes and the remainder nodes being Rational or Altruistic, as long as there are at least $f + 3$ nodes in total.

In TRB, a distinguished node—the *sender*—initiates a broadcast. We call the broadcasted value the *proposal.* TRB guarantees four properties:

- *Validity*: if the sender is correct and broadcasts a message m, then every correct node eventually delivers m;

- *Agreement*: all correct nodes deliver the same message;

- *Integrity*: all correct nodes deliver only one message; and

- *Termination*: every correct node eventually delivers some message.

Our goal for ATRB+ is to guarantee the same properties when "correct" is changed to "Altruistic or Rational". Further, these guarantees should hold even if no Altruistic node is present. In ATRB+, nodes can indicate whether they are busy or idle, and participating in the protocol is less costly when nodes are idle. The goal of ATRB+ is to leverage altruism to improve the *social welfare*, which for this problem we define to be the sum of every non-Byzantine node's utility. To reach this goal, ATRB+ shifts work from the busy nodes to the idle nodes. Of course, it is not possible to check whether nodes report their busy/idle status truthfully and Rational nodes may pretend to always be busy: writing an IC-BFT protocol would be challenging since IC-BFT requires the suggested protocol to be optimal for all nodes. ATRB+ guarantees that idle Altruistic nodes, if present, will take on work from busy nodes. When Altruistic nodes are busy, work is allocated fairly between Rational and Altruistic nodes. Thus, social welfare is optimized without taking undue advantage of the Altruistic nodes.

The ATRB+ protocol is based on TRB+ [29], an IC-BFT TRB protocol developed by the authors of [9] that, in turn, is based on Lamport's TRB protocol [21].

4.2 Utility to Rational nodes

Following the framework established in Section 3.1, the first step of deriving ATRB+ is to establish a utility function for the Rational nodes. Each node can be *busy* or *idle*, indicating whether the machine is currently being used for other tasks or not. When a machine is busy, the cost to a rational node is proportional to the number of bytes it sends and the number of digital signatures it generates (these approximate the bandwidth and CPU costs). When a machine is idle, the cost is a tenth of the busy cost (BAR-Tolerance is maintained for any fraction between zero and one, choosing 1/10th is arbitrary). Rational nodes only get a benefit on instances when they have the role of sender and if they believe that Validity, Agreement, Integrity and Termination hold despite arbitrary deviation from the Byzantine nodes and deviation within $\vec{\Upsilon}$ for the Rational nodes. We use α, β, and γ to denote the constants for the cost of sending and signing and the benefit of broadcasting, respectively. Constant γ must be such that there is an overall benefit for participating in the protocol.

The second step is to define the estimated utility. We use definition from Section 3.2, i.e. we assume Rational nodes are risk-averse.

4.3 Establishing a BAR Equilibrium

Again following the framework (Section 3.1), the next step is to choose the suggested protocol $\vec{\sigma}$ and the rational envelope $\vec{\Upsilon}$. The starting point is clear: $\vec{\sigma}$ indicates that nodes should report they busy/idle status truthfully, and $\vec{\Upsilon}$ allows for lies.

The next step is to come up with a protocol that implements TRB but leverages the busy/idle indicator to optimize social welfare. We start from the IC-BFT TRB+ protocol and apply two observations.

First, as Dolev and Strong observed [11], it is possible to reduce the number of messages in TRB by designating $f + 1$ nodes as *relay* and allowing other nodes to send messages only to the relay nodes and not to each other directly. We deviate slightly from how they applied that observation by using $f+2$ relay nodes and a few more messages to tolerate Rational nodes and to ensure that the protocol still completes in $f + 1$ rounds rather than $f + 2$ in their case. We also modify the credible punishment part of TRB+ so that non-relay nodes do not have to participate.

Second, if the relay nodes are chosen preferentially among the idle nodes (only resorting to busy nodes if too few nodes are idle) then social welfare increases. When there is a choice between nodes, we prefer nodes whose identity number are closest to that of the sender. Since the role of sender is rotated round-robin among the nodes, this ensures that (for long periods where no node changes its idle/busy status) the role of relay is spread evenly among the idle nodes.

Applying these observations yields the ATRB+ protocol (Figures 1 and 2). Altruistic nodes truthfully set their busy/idle status and Rational nodes pretend to always be busy in order to reduce their work. If all the nodes are Rational, then work is spread equally among the nodes. If some nodes are Altruistic, then some work is shifted toward them when they are idle, increasing social welfare. As more nodes behave altruistically, more work is correctly shifted to idle nodes and the social welfare is further increased. The pseudocode of Figures 1 and 2 use *m:x* to denote message m signed by node x, and $S - x$ to denote $S \setminus \{x\}$. Variables that contain sets start with an uppercase. We use G_p to

```
**Round 1** for process $p$ on instance $k$ of the protocol :
10.  $Relay := \textbf{getRelay}(k, n, busy[\,])$
11.  if $f > 0$ then : $Dest := Relay$
12.  else : $Dest := \{0 \ldots (n-1)\} - sender$
13.  if $p == sender$ then :
14.      $msg := (proposal, \textbf{isBusy}())$
15.      send $((k, j){:}p,\ (k{,}msg){:}p)$ to every process $j$ in $G_p \cap Dest$
16.      $busy[p] := msg[1]$ ; deliver $msg[0]$; end
17.  else if $p \in Dest$ :
18.      if received $(k{,}p){:}sender$ then :
19.          $ticket := (k, p){:}sender$
20.      else :
21.          $ticket := \textbf{penance}(p)$
22.          $G[p, sender] := false$
23.          $G_p := G_p - sender$
24.      if received $(k{,}msg){:}sender \wedge msg == (proposal, newbusy) \wedge sender \in G_p$ then :
25.          $Extracted := \{msg\}$
26.          $Fwd := \{\ (k{,}msg){:}sender\ \}$
27.      else :
28.          $Extracted := Fwd := \emptyset$
29.          $G_p := G_p - sender$
30.  else : // $p \notin$ Dest
31.      $Extracted := Fwd := \emptyset$

**Round $i$** $(2 \le i \le f+1)$ for process $p \neq sender$ :
40.  if $(p \in Relay) \vee (i == f+1)$ then :
41.      $Dest := G_p - sender$
42.  else :
43.      $Dest := (G_p - sender) \cap Relay$
44.  // 1. Send penance to relay nodes when necessary
45.  if $i == 2 \wedge p \in Relay$ then : send $ticket$ to all procs. in $(G_p - sender) \cap Relay$
46.  // 2. Send two messages to all participants who have not deviated
47.  for each $s \in Fwd$ :
48.      send $s{:}p$ to all processes in $Dest$
49.  if $|Fwd| < 1$ then : send $\textbf{filler}(1, i, p)$ to all procs. in $Dest$
50.  if $|Fwd| < 2$ then : send $\textbf{filler}(2, i, p)$ to all procs. in $Dest$
51.  $Fwd := \emptyset$
52.  // 3. Receive penances when necessary
53.  if $i == 2 \wedge p \in Relay$ then :
54.      for each process $j \in (G_p - sender) \cap Relay$ :
55           $G[j, sender] := G[j, sender] \wedge$ (received $(k{,}j){:}sender$ from $j$)
56.          if received neither $(k{,}j){:}sender$ nor $\textbf{penance}(j)$ from $j$ then :
57.              $G_p := G_p - j$ // process $j$ deviated from the protocol
58.  // 4. Receive messages, check for deviation
59.  for each process $j \in Dest$ : // set of nodes $p$ sends to == set of nodes $p$ expects to receive from
60.      if also received from $j$ $s_1$ and $s_2$ s.t. $\textbf{valid}(s_1, i, j) \wedge \textbf{valid}(s_2, i, j) \wedge \textbf{head}(s_1) \neq \textbf{head}(s_2)$,
         and nothing else then :
61.          for each integer $l$ s.t. $1 \le l \le 2$ :
62.              if $\textbf{interesting}(s_l, i, j)$ then :
63.                  $Extracted := Extracted \cup \{\textbf{head}(s_l)\}$
64.                  if $|Fwd| < 2$ then : $Fwd := Fwd \cup \{s_l\}$
65.      else : $G_p := G_p - j$ // process $j$ deviated from the protocol

**Postprocessing** : // At the end of the last round, decide
80.  if $\exists msg$ s.t. $Extracted == \{msg\}$ then :
81.      $busy[sender] := msg[1]$
82.      deliver $msg[0]$
83.  else : // sender faulty
84.      $G_p := G_p - sender$
85.      deliver **SF**
```

Figure 1: ATRB+, a BAR-Tolerant protocol for Terminating Reliable Broadcast. The protocol is run continuously, nodes take turn being sender for an instance. The next figure shows initialization.

```
Initialization for process $p$ :
1.   $G_p := \{0 \ldots (n-1)\} \setminus \{p\}$
2.   $G[x,y] := true$ for each $(x,y) \in \{0 \ldots n-1\}^2$
3.   $busy[x] := true$ for each $x \in \{0 \ldots (n-1)\}$

valid$(m,i,j)$ :
100. if head$(m)[0] \neq k$ : return $false$
101. if   head$(m)[1] \neq \bot_1 \wedge$ head$(m)[1] \neq \bot_2$
102.      $\wedge$ first signature on $m$ is from $sender$
103.      $\wedge$ last signature on $m$ is from $j$
104.      $\wedge$ tail$(m)$ is a chain of $i$ signatures :
105.      return $true$
106. if $m ==$ filler$(1,i,j) \vee m ==$ filler$(2,i,j)$ : return $true$
107. return $false$

interesting$(m,i,j)$ :
110. return (
111.      head$(m)[0]{==}k$
112.      $\wedge$ head$(m)[1] \neq \bot_1 \wedge$ head$(m)[1] \neq \bot_2$
113.      $\wedge$ first signature on $m$ is from $sender$
114.      $\wedge$ last signature on $m$ is from $j$
115.      $\wedge$ tail$(m)$ is a chain of $i$ distinct signatures
116.      $\wedge$ $m$ does not contain our own signature
117.      $\wedge \forall x \in Fwd$ : head$(x) \neq$ head$(m)$  )

padding$(l)$ :
120. return a sequence of zeroes of length $l$.

filler$(pos,i,p)$ :
130. return $((k, \bot_{pos}),$ padding$(\eta_s(i-1))){:}p$ // $\eta_s$ is the size of a signature

penance$(j)$ :
140. return a message starting with $k$ and of at least the same cost
     (number of bytes, number of signatures) as faithfully following the protocol.

getRelay$(k, n, busy[\,])$
150. $sender = k \bmod n$
151. let $score(x) = ((n + x - sender) \bmod n) + (n$ if $busy[x])$
152. let $score(X) = \sum_{x \in X} score(x)$
153. pick $Relay \subseteq \{0 \ldots (n-1)\} \setminus \{sender\}$ that minimizes $score(Relay)$ and s.t. $|Relay| = f + 2$.
154. return $Relay$
```

Figure 2: Helper functions for ATRB+

denote the value of variable G on node p.

To make sure that the rational envelope is a BAR Equilibrium, we follow the advice from Section 3.3 and use a credible punishment mechanism to deter nodes from deviating from the rational envelopes. In order to further social welfare, we design the punishment mechanism so that only the relay nodes are involved, leaving the other (potentially busy) nodes undisturbed.

The main difference between ATRB+ and the TRB+ protocol it derives from is the mechanism that focuses work on the relay nodes. The lines in Figure 1 that are related to this mechanism are numbered in bold.

4.4 ATRB+ Correctness

To prove that ATRB+ is correct, we first define the ATRB+ protocol and the rational envelope $\vec{\Upsilon}$, and then we show that (ATRB+,$\vec{\Upsilon}$) is BAR-Tolerant by clearing the two requirements of Definition 2.

1. ATRB+ is Byzantine fault-tolerant for all protocols in $\vec{\Upsilon}$, and
2. (ATRB+,$\vec{\Upsilon}$) is a BAR Equilibrium

Recall that we chose, for $\vec{\Upsilon}$, to allow Rational nodes to lie about their idle/busy status (so their

getBusy() (Figure 1, line 14) may return *true* even if the node is not actually busy).

4.4.1 ATRB+ is Byzantine Fault-Tolerant

The next few lemmas allow us to establish that ATRB+ is Byzantine fault-tolerant regardless of which protocol in $\vec{\Upsilon}$ the Rational nodes follow. Lemmas 1–8 assume that there are at most f Byzantine nodes and that all non-Byzantine nodes stay within the rational envelope $\vec{\Upsilon}$. Unless otherwise indicated, line numbers 10–90 refer to Figure 1 and other line numbers refer to Figure 2.

Lemma 1. *For the first instance of the protocol, $Relay_p$ is identical across all nodes p that follow $\vec{\Upsilon}$.*

Proof. *Relay* is built deterministically on the basis of *busy*[] (Figure 2, lines 150–154). This amounts to showing that there is a unique minimal *score* on line 153. By construction: for all $x \neq y \in \{0 \ldots (n-1)\}$, $score(x) \neq score(y)$ and *Relay* can be built by adding the lowest-scoring node, $f+2$ times. All nodes initially agree on the value of *busy*[] (Figure 2, line 3) and so all non-Byzantine nodes start out with the same *Relay*. □

Lemma 2. *If ATRB+ implements TRB and if nodes r and p stay within the rational envelope $\vec{\Upsilon}$, then $r \in G_p$.*

Proof. There are three places where r could be removed from G_p on the first instance of the protocol: lines 23, 29, 57, 65, and 84. We show that in each case, being in Υ_r guarantees that r is not removed from G_p.

Initially, G_p contains all nodes other than p (Figure 2, line 1). Node p executes line 23 if p is in *Relay* and the sender did not sent a ticket and line 29 is executed if p is in *Relay* and the sender did not send a proposal. If r is the sender then it will send both a ticket and a proposal to all the nodes in $G_r \cap Relay$ on line 15.

Line 57 removes a node from G_p if it should have sent a penance but did not. Since $Relay_p = Relay_r$, node r sends the expected penance on line 45 (the two conditions are identical).

Line 65 checks that r sent two distinct values that satisfy the **valid**(...) predicate. These values are sent in lines 47–50. No more than two values are sent because nothing is added to *Fwd* if it already has size two (line 64). The messages sent pass the **valid**(...) test because they are constructed either from **filler**(...) or from a messages that passes the **interesting**(...) test (Figure 2, lines 110–117) and the required signature is added (lines 15, 48).

Finally, line 84 only removes r if **SF**is delivered, which never happens if ATRB+ implements TRB and r and p follow the rational envelope. □

To prove that ATRB+ implements TRB, we start with a simpler version, ATRB+', where we removed the condition on line 64, so that any number of values can extracted (not just the first two). Putting the test back causes some messages to be omitted, but it does not cause any node to deliver a different value, so safety is maintained.

Lemma 3. *In ATRB+', if non-Byzantine nodes have the same value for* Relay*, then if non-Byzantine r extracts v, then all non-Byzantine nodes extract v as well.*

Proof. The value v can either have been extracted on a round $i < f$, $i = f$, or on round $f+1$. There are no other rounds. If $i < f$, then the extracted value was put in the *Fwd* set, signed, and forwarded to all the other non-Byzantine, non-sender nodes (*Extracted* is assigned on lines 25, 28, 31, or 63, and in each case the same value is added to *Fwd* since $|Fwd| < 2$ on line 64. Since $i < f+1$, there are at least two more rounds. On the next round *Fwd* are signed and forwarded to all processes in $(Relay \cap G_r) - sender$ (lines 40–43 and 48) because *Dest* always includes nodes that are in *Relay* and G_r. Since *Relay* has size $f+2$ and non-Byzantine nodes are not removed from G_r (Lemma 2), some correct relay node c receives value v. In ATRB+' all interesting values are extracted so c will, like r, add v to its *Fwd* set and forward it with the required additional signature. This time, v is sent to all nodes in $G_c - \mathit{sender}$ (line 41, $c \in \mathit{Relay}$) and so all non-Byzantine nodes other than the sender extract v as well. The sender already effectively extracted v (line 16) since the forwarded value must have been originally sent by the sender (a message without a valid signature would not pass **valid**$(\ldots)$, see Figure 2, line 102). The conclusion thus holds for the case $i < f$.

The case $i = f$ is similar to $i < f$, except that then in round $f + 1$, node r will forward v to all nodes in $G_r - \mathit{sender}$, not just those in *Relay* (see line 40). At non-Byzantine nodes that have not yet extracted v these messages will pass **interesting**$(\ldots)$, and thus all non-Byzantine nodes will extract V.

If $i = f + 1$, then the received value was forwarded $f + 1$ times and so it was forwarded by at least one non-Byzantine node c. Node c received the value in an earlier round, we can use the previous two cases to conclude that every non-Byzantine node extracts v as well. □

Lemma 4. *In ATRB+', if non-Byzantine nodes have the same value for* Relay*, then all non-Byzantine nodes deliver the same message.*

Proof. By Lemma 3, all non-Byzantine have the same *Extracted* set at the end of round $f + 1$. Since the delivery value is determined deterministically from *Extracted*, all non-Byzantine nodes deliver the same message. □

Lemma 5. *In ATRB+, if non-Byzantine nodes have the same value for* Relay*, then all non-Byzantine nodes deliver the same message (Agreement).*

Proof. The difference between ATRB+' and ATRB+ is that in the latter, Rational nodes do not forward more than two values. However, once a non-Byzantine node has extracted two distinct values, extracting more will not change its decision.

Consider non-Byzantine node r. It can extract either zero, one, or two values. It cannot extract more, by design (line 64). If it extracts zero values then no other non-Byzantine node extracted a value, and so all non-Byzantine nodes deliver **SF**and Agreement holds. If it extracts a single value, then all other non-Byzantine nodes also extract that value and, further, no other non-Byzantine node extracted a different value. Therefore, the conclusion holds again. Finally, if r extracts two values then there is a non-Byzantine node that received both of these values. However, it may not have forwarded them if it extracted a different set of two values. Still, it follows that all non-Byzantine nodes extract two values and so all decide **SF**. □

Lemma 6. *If non-Byzantine nodes have the same value for* Relay*, then if the sender is non-Byzantine and broadcasts a message m, then every non-Byzantine node eventually delivers m (Validity).*

Proof. The non-Byzantine sender, after broadcasting m, delivers it (line 16). The conclusion follows from Lemma 5. □

Lemma 7. *If non-Byzantine nodes have the same value for* Relay*, then the ATRB+ protocol guarantees Validity, Agreement, Integrity, and Termination in the presence of up to f Byzantine nodes, as long as all non-Byzantine nodes stay within the rational envelope.*

Proof. Lemmas 6 and 5 prove validity and agreement, respectively. Integrity holds because the code only calls deliver once (line 16 for the sender, 82 or 85 otherwise). Termination holds because there are no loops and none of the calls are blocking. The protocol completes at round 1 for the sender and round $f + 1$ for the other nodes. □

Lemma 8. *If $\mathit{Relay}_a = \mathit{Relay}_b$ for all non-Byzantine nodes a and b at instance k of the protocol, then $\mathit{Relay}_a = \mathit{Relay}_b$ at instance $k+1$.*

Proof. All nodes start with the same *Relay* (Lemma 1). *Relay* is computed on the basis of *busy*[ ] and *busy*[ ] is only updated on lines 16 and 81, just before nodes deliver a value. If the sender is non-Byzantine or if *Extracted* has size one, all non-Byzantine nodes have the same element in *Extracted* (Lemma atrb+agreement). Therefore, all non-Byzantine nodes agree on the update to *busy*[*sender*] and thus both *busy*[ ] and *Relay* remain consistent across non-Byzantine nodes. □

Theorem 1. *The ATRB+ protocol guarantees Validity, Agreement, Integrity, and Termination in the presence of up to f Byzantine nodes, as long as all non-Byzantine nodes stay within the rational envelope.*

Proof. Initially, all non-Byzantine nodes agree on *Relay* (Lemma 1). The first instance of ATRB+ therefore guarantees Validity, Agreement, Integrity and Termination (lemma 7). This, in turn, ensures that all non-Byzantine nodes agree on *Relay* for the next instance as well (Lemma 8) and so all instances of ATRB+ satisfy Validity, Agreement, Integrity and Termination. □

This theorem concludes our proof that ATRB+ is Byzantine fault-tolerant.

4.4.2 (ATRB+,$\vec{\Upsilon}$) is a BAR Equilibrium

Having shown that ATRB+ is Byzantine fault-tolerant, the next step is to show that (ATRB+,$\vec{\Upsilon}$) is a BAR Equilibrium, as per Definition 1. We must show that deviating outside of the rational envelope is never better than staying inside. We chose the rational envelope in such a way that leaving it causes a node to be shunned, so we show that it is never worthwhile to leave the rational envelope. To this effect, we choose the **penance**(j) method to generate a message for node j that has the following property: sending that message to all the nodes that j does not currently shun is more costly that sending the messages required by the protocol to all the nodes that j currently shuns. Since nodes' busy or idle status influences which nodes may participate in *Relay*, and since the status is only updated when a node successfully broadcasts a value, it could potentially be in a node's best interest to prevent another node from going from idle to busy. We show that a node cannot, by itself, prevent another from broadcasting a value.

Lemma 9. *A Rational node r cannot prevent a non-Byzantine sender s from broadcasting a value.*

Proof. In its first step, node b sends its value to all nodes in *Relay* that it does not shun (lines 10, 15). *Relay* has size $f+2$ (line 153) and nodes that follow the protocol are not shunned (Lemma 2), so at least two non-Byzantine nodes receive s's value. So even if r is in *Relay* and tries to suppress messages, the other non-Byzantine node in *Relay* will correctly forward messages, so every non-Byzantine node will receive the sender's value. The only other way to prevent s's value from being successfully broadcast would be for r to cause the other nodes to extract a second value. However, only values that bear the sender's signature can be extracted (lines 62 and 113) and a non-Byzantine sender only signs a single value. □

Lemma 10. *It is never in the best interest of a Rational node r to be shunned by another Rational node s.*

Proof. Once r is shunned by s, there is no longer any incentive for r to do anything that s expects, so a benefit to r is that it can omit sending messages to s and can omit computations (if there were any) that are only checked by s. The cost of being shunned by s is that r no longer receives a ticket from s, so it must send the expensive **penance**(r) message. By construction, **penance**(r) returns a message that is strictly costlier to generate and send than following the protocol with respect to s would be. It is therefore in r's best interest not to be shunned by other rational nodes. □

Having shown that shunning is a sufficient deterrent, we now show that every unilateral deviation from ATRB+ either directly reduces the estimated utility or results in being shunned. To increase their estimated utility nodes must reduce their cost, so we look at every cost in turn and consider the set of deviations that could prevent the cost. We show that they all result in shunning.

Lemma 11. *Not sending the ticket from line 15 results in shunning.*

Proof. The first opportunity for Rational node p to cut cost is to avoid sending the ticket *(k,j):p*, sending a ticket that is shorter or is missing the signature, or resending an old ticket so as not to have to compute the signature again. There is no other cost-cutting opportunity from deviating at that line of the protocol. All these changes result in the destination node shunning p (line 23) because the circumstances under which the message should be sent are the same as the circumstances under which the message is expected (recipient is not shunned and $f = 0$ or the recipient is in *Relay*). All tickets have the same size (and the recipient expects the ticket to be of the right size), so a malformed ticket will be detected. The instance number is part of the ticket, so it is impossible to reuse a ticket from a previous instance (only one ticket is issued per instance). There is no risk if the destination is known to be Byzantine, but the ATRB+ protocol already compels p to shun all nodes that it believes are Byzantine, since any deviating node is immediately removed from G_p and thus shunned.

Short of shunning nodes (which we already know is detrimental), there is nothing that p can do to modify the set of nodes to which it is expected to send the ticket: it is determined from f and *Relay*, both of which are outside of p's control (Lemma 9). □

Lemma 12. *Not sending the proposal from line 15 results in shunning.*

Proof. Proposals are constant-size, so a node cannot cut costs by sending a shorter proposal. Not sending the proposal, even to one node that is supposed to receive it, would result in the sender being shunned for the same reason as not sending the ticket: the set of nodes that the sender must send a proposal to (computed in lines 10–12) is the same set of nodes that expect a proposal from the sender and will shun it otherwise (lines 27–29), with the exception of the nodes that already shun the sender (those ignore the sender's proposal (line 24), but will accept the *ticket* if they receive it (line 18)). □

Lemma 13. *Not sending a ticket or penance at line 45 results in shunning.*

Proof. Line 45 requires that, in round 2, all *Relay* nodes forward a valid ticket or penance to all other *Relay* nodes (that they don't already shun). At lines 54–57, nodes in *Relay* check that they receive a ticket or penance from all other *Relay* nodes, or they start shunning them. So not sending the message or sending an invalid message results in shunning.

It is not possible, either, to send a cheaper message than required (meaning shorter or with fewer signatures) as recipients check that the message has the right form (line 56). □

Lemma 14. *There is no benefit in sending a* **filler**(...) *message rather than a Fwd message at lines 47–50.*

Proof. These two messages have exactly the same cost: they contain the instance number, *msg* (a proposal and the busy flag), i signatures, and the last signature must be new. The **filler**(...) replaces the contents with filler of the same length, so there is no incentive to deviate. Since the message is specific to the instance and grows with every round, it is not possible to reuse a previous signature. □

Lemma 15. *Not sending a Fwd or* **filler**(...) *message at lines 47–50 results in shunning.*

Proof. Lines 47–50 call for exactly two messages to be sent to all nodes in *Dest*. The pattern is that (except for shunned nodes), *Relay* nodes send to all but the sender, and non-*Relay* nodes send only to *Relay* nodes. So *Relay* nodes expect to receive messages from all but the sender, and non-*Relay* nodes expect to receive messages from *Relay* nodes. Line 59 selects each node that must forward, and line 60 checks that it sent two distinct valid messages. Nodes that fail that test are shunned. So, all recipients of the messages sent in lines 47–50 check that they received the message. The **valid**(...) function only accepts messages of the same cost as what is required to send, so all deviations from lines 47–50 that otherwise reduce costs result in shunning. □

Since there are no other places where nodes incur costs, the lemmas above show that no unilateral deviation from the rational envelope can increase a node's utility.

5 Related Work

The BAR model [5, 9, 29] bridges the world of game theory and the world of fault-tolerance. The former is mostly concerned with the behavior of self-interested or Rational nodes, and the latter is mostly concerned with the behavior of nodes that fail either by crashing or by arbitrarily deviating from the protocol (Byzantine nodes). Two previous protocols implement TRB in the BAR model (TRB+ [29] and BaN TRB [9]), but they cannot leverage Altruistic nodes. We list below some other works that have explored the interface between game theory and fault-tolerance before us.

The field of *mechanism design* [30] studies how to maximize the *social welfare*, a function of the outcome that indicates how beneficial the outcome is to society as a whole (instead of the individual players). In mechanism design, the designer modifies the *outcome* function so that, when the Rational nodes act to maximize their individual utility, the resulting social welfare is also maximized. This approach is useful in contexts where the protocol designer has control over what should happen in response to the nodes' actions, for example in auctions. There are similarities between our approach to building BAR-Tolerant protocols and mechanism design, but we differ on a fundamental point: in our approach to designing BAR-Tolerant protocols, we cannot change the *outcome* function: the outcome observed by the nodes is a direct result of their interactions, which in turn depend only on each node's strategy. Since we cannot manipulate *outcome*, instead we manipulate the initial protocol σ that we propose to the nodes, with the goal of influencing their final strategy choice.

Several researchers have begun to examine constructing incentive compatible systems [3, 33, 37, 38]. There has been some success in applying mechanism design to distributed systems in routing [4, 14, 15] and caching [8]. Unfortunately, these efforts generally ignore Byzantine behavior and adopt a model where every node is rational.

Eliaz [13] suggests the notion of k-fault tolerant Nash equilibrium (k-FTNE) in the context of mechanism design. Their parameter k plays the same role as our f, the bound on the number of Byzantine nodes and so their equilibrium concept bridges game theory and fault-tolerance. Abraham et al. [1] added the notion of t colluding nodes, creating the notion of (k, t)-robustness. As we have shown [9], both notions are useful but they are not applicable when sending messages can be considered a cost, because then their definitions require that there exist no deviation that would, *ex-post*, increase the utility. However, if some node crashed, then it could be beneficial to deviate by omitting messages to that node (even before it crashes) and so no protocol is k-FTNE or (k, t)-robust. The notion of BAR-Tolerance (as well as IC-BFT) escapes this difficulty because it considers *ex-ante* utility, meaning the utility that nodes estimate they may gain, before the protocol is run to completion.

The work of Abraham et al. [1] also denotes some nodes as "altruistic", but their meaning differs from ours. They use this name for nodes that will deviate from the protocol in ways that these nodes believe are beneficial to the system as a whole. Our model, instead, separates the notion of Altruistic nodes (in our case these nodes follow a specially tailored protocol that is beneficial to the system) from non-rational nodes (the Byzantine nodes, whose utility function is unknown and whose deviation may aim to help or harm the system).

Moscibroda et al. [32] analyze the interaction of Rational and Byzantine nodes in the context of a specific one-shot game, the virus inoculation game. They measure the "price of anarchy" and the "price of malice", defined respectively as the impact of having Rational nodes (instead of all Altruistic) and having Byzantine and Rational nodes (instead of all Rational). Like us, they find that the presence of Byzantine nodes can simplify the design of BAR-Tolerant protocol when the nodes are risk-averse. What remains to be seen is to what extent the presence of Altruistic nodes can reduce the price of anarchy.

5.1 Acknowledgments

The idea of the BAR Equilibrium was already present in earlier work [5] and we recognized its potential for leveraging Altruistic nodes. The contribution of this paper is to formalize the concept of BAR Equilibrium and show the first BAR-Tolerant protocol that leverages Altruistic nodes. We would like to acknowledge the authors of earlier BAR work [5, 9, 22] for their influence in the results presented here.

6 Conclusion

This paper presents two results: the first is an extension of the BAR model [5, 9, 29] that gives a new equilibrium concept (BAR Equilibrium) that makes it possible to leverage Altruistic nodes, unlike the previous concept of Incentive-Compatible Byzantine Fault-Tolerance. The key contribution of BAR Equilibrium is that it tolerates some degree of deviation from the Rational nodes.

The second result is the ATRB+ protocol. This protocol implements Terminating Reliable Broadcast despite Byzantine and Rational behavior and Altruistic nodes, if present, are used to shift work from busy to idle nodes. This shift improves the common good (or social welfare, computed by summing the utility function of every non-Byzantine node). This sort of optimization was not possible without the BAR Equilibrium concept as there is no way to prevent Rational nodes from lying about whether they are busy or idle. If all nodes lie then work is distributed evenly and, as Altruistic nodes are added, work is shifted to them when they are idle. The protocol, while interesting in its own right, also serves the purpose of illustrating how one can design and verify a BAR-Tolerant protocol.

References

[1] I. Abraham, D. Dolev, R. Gonen, and J. Halpern. Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation. In *Proc. 25th PODC*, pages 53–62, July 2006.

[2] E. Adar and B. A. Huberman. Free riding on Gnutella. *First Monday*, 5(10):2–13, Oct. 2000.

[3] M. Afergan. *Applying the Repeated Game Framework to Multiparty Networked Applications*. PhD thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, Aug. 2005.

[4] M. Afergan and R. Sami. Repeated-game modeling of multicast overlays. In *IEEE INFOCOM 2006*, Apr. 2006.

[5] A. S. Aiyer, L. Alvisi, A. Clement, M. Dahlin, J.-P. Martin, and C. Porth. BAR fault tolerance for cooperative services. In *Proc. 20th SOSP*, pages 45–58, Oct. 2005.

[6] C. Batten, K. Barr, A. Saraf, and S. Trepetin. pStore: A secure peer-to-peer backup system. Technical Memo MIT-LCS-TM-632, Massachusetts Institute of Technology Laboratory for Computer Science, October 2002.

[7] G. Bracha and S. Toueg. Asynchronous consensus and broadcast protocols. *J. ACM*, 32(4):824–840, 1985.

[8] B.-G. Chun, K. Chaudhuri, H. Wee, M. Barreno, C. H. Papadimitriou, and J. Kubiatowicz. Selfish caching in distributed systems: a game-theoretic analysis. In *Proc. 23rd PODC*, pages 21–30. ACM Press, 2004.

[9] A. Clement, J. Napper, H. Li, J.-P. Martin, L. Alvisi, and M. Dahlin. A theory of BAR games. Technical Report 06-63, University of Texas at Austin Computer Sciences, Nov. 2006.

[10] L. P. Cox and B. D. Noble. Samsara: honor among thieves in peer-to-peer storage. In *Proc. 19th SOSP*, pages 120–132, 2003.

[11] D. Dolev and H. R. Strong. Authenticated algorithms for Byzantine agreement. *Siam Journal Computing*, 12(4):656–666, Nov. 1983.

[12] C. Dwork, N. Lynch, and L. Stockmeyer. Consensus in the presence of partial synchrony. *J. ACM*, 35(2):288–323, 1988.

[13] K. Eliaz. Fault tolerant implementation. *Review of Economic Studies*, 69:589–610, Aug 2002.

[14] J. Feigenbaum, C. Papadimitriou, R. Sami, and S. Shenker. A BGP-based mechanism for lowest-cost routing. In *Proceedings of the twenty-first annual symposium on Principles of distributed computing*, pages 173–182. ACM Press, 2002.

[15] J. Feigenbaum, R. Sami, and S. Shenker. Mechanism design for policy routing. In *Proc. 23rd PODC*, pages 11–20. ACM Press, 2004.

[16] D. Fudenberg and J. Tirole. *Game theory.* MIT Press, Aug. 1991.

[17] Gnutella. http://www.gnutella.com/.

[18] G. Hardin. The tragedy of the commons. *Science*, 162:1243–1248, 1968.

[19] B. Huberman and R. Lukose. Social dilemmas and internet congestion. *Science*, 277:535–537, July 1997.

[20] D. Hughes, G. Coulson, and J. Walkerdine. Free riding on Gnutella revisited: the bell tolls? *IEEE Distributed Systems Online*, 6(6), June 2005.

[21] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[22] H. C. Li, A. Clement, E. Wong, J. Napper, I. Roy, L. Alvisi, and M. Dahlin. Bar Gossip. In *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI '06)*, pages 191–204, Nov. 2006.

[23] M. Lillibridge, S. Elnikety, A. Birrell, M. Burrows, and M. Isard. A cooperative internet backup scheme. In *USENIX ATC*, pages 29–41, june 2003.

[24] LimeWire. http://www.limewire.com/.

[25] W. Lloyd. *Two Lectures on the Checks to Population.* Oxford University Press, 1833.

[26] R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan. Sustaining cooperation in multi-hop wireless networks. In *NSDI*, May 2005.

[27] D. Malkhi and M. Reiter. Byzantine quorum systems. *Distributed Computing*, 11(4):203–213, 1998.

[28] P. Maniatis, D. S. H. Rosenthal, M. Roussopoulos, M. Baker, T. Giuli, and Y. Muliadi. Preserving peer replicas by rate-limited sampled voting. In *Proc. 19th SOSP*, pages 44–59. ACM Press, 2003.

[29] J.-P. Martin. *Byzantine Fault-Tolerance and Beyond.* PhD thesis, The University of Texas at Austin, Dec. 2006.

[30] A. Mas-Colell, M. D. Whinston, and J. R. Green. *Microeconomic Theory.* Oxford University Press, 1995.

[31] S. Misel. Wow, AS7007! NANOG mail archives http://www.merit.edu/mail.archives/nanog/1997-04/msg00340.html.

[32] T. Moscibroda, S. Schmid, and R. Wattenhofer. When selfish meets evil: Byzantine players in a virus inoculation game. In *Proc. 25th PODC*, pages 35–44, July 2006.

[33] C. Papadimitriou. Algorithms, games, and the internet. In *Proc. 33rd STOC*, pages 749–753. ACM Press, 2001.

[34] D. C. Parkes. *Iterative Combinatorial Auctions: Achieving Economic and Computational Efficiency.* PhD thesis, Department of Computer and Information Science, University of Pennsylvania, May 2001.

[35] F. B. Schneider. What good are models and what models are good? In *Distributed Systems, 2nd ed.*, chapter 2, pages 17–25. Addison Wesley, July 1993.

[36] Seti@home. http://setiathome.ssl.berkeley.edu/.

[37] J. Shneidman and D. C. Parkes. Specification faithfulness in networks with rational nodes. In *Proc. 23rd PODC*, pages 88–97. ACM Press, 2004.

[38] E. Tardos. Network games. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 341–342. ACM Press, 2004.