Decentralized Polling With Respectable Participants

Rachid Guerraoui, Kévin Huguenin, Anne-Marie Kermarrec, Maxime Maxime Monod

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Decentralized Polling With Respectable Participants

Rachid Guerraoui — Kévin Huguenin — Anne-Marie Kermarrec — Maxime Monod

N° 6890

March 2009

Thème COM

ISSN 0249-6399 ISRN INRIA/RR--6890--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
RENNES - BRETAGNE ATLANTIQUE

Decentralized Polling With Respectable Participants

Rachid Guerraoui* , Kévin Huguenin , Anne-Marie Kermarrec , Maxime Monod*

Thème COM — Systèmes communicants
Équipe-Projet ASAP

Rapport de recherche n° 6890 — March 2009 — 16 pages

Abstract: We consider the polling problem in a social network where respectable participants do care about their reputation: they do not want their vote to be disclosed nor their misbehaving, if any, to be publicly exposed. Assuming this reputation concern, we show that a simple secret sharing scheme, combined with *a posteriori* verifications, can efficiently enable polling without the need for any central authority or heavyweight cryptography.

More specifically, we present a scalable distributed polling protocol where *(i)* misbehaving nodes are exposed with a non-null probability, *(ii)* the reputation of honest nodes is never tarnished, and *(iii)* the probability of malicious participants violating privacy is traded with their impact on the accuracy of the polling result. The trade-off is captured by a generic parameter of the protocol, an integer k we call *privacy parameter*, so that in a system of N nodes with $B < \sqrt{N}$ malicious participants, the probability of disclosing a participant's vote is bounded by $(B/N)^{k+1}$, whereas the impact on the polling result is bounded by $(6k+2)B$.

We report on the deployment of our protocol over 400 PlanetLab nodes with a privacy parameter $k = 2$. The polling result suffers a relative error of less than 10% in the face of message losses, crashes and asynchrony inherent in PlanetLab. In presence of malicious nodes, our experiments show that the impact on the polling result is $(4k+2)B$ on average, consistently lower that the theoretical bound of $(6k+2)B$.

Key-words: distributed polling, social networks, privacy, accountability

* Laboratoire de programmation distribuée (LPD), École Polytechnique Fédérale de Lausanne (EPFL)

Centre de recherche INRIA Rennes – Bretagne Atlantique
IRISA, Campus universitaire de Beaulieu, 35042 Rennes Cedex
Téléphone : +33 2 99 84 71 00 — Télécopie : +33 2 99 84 71 71

Sondage distribué pour pairs honorables

Résumé :

Mots-clés : sondage distribué, réseaux sociaux, vie privée, audit

1 Introduction

Social networks are growing exponentially, and one of the most celebrated examples, Facebook [7], currently gathers more than 150 millions active users. Many of these users regularly share images and videos as well as discuss various social and political matters both with close friends and people they hardly know. For there is usually a one to one mapping between social network identities and real ones (as opposed to virtual world platforms such as SecondLife [19]), *reputation* is a primary concern: information related to a user is intimately considered to reflect the associated real person.

In this paper, we look at *polling*, a privacy-sensitive task that might be appealing to trigger from within a social network, say a French political group on Facebook asking whether Obama would have been elected in France, or the organizers of a Saturday night party asking whether partners should be invited too. We consider *binary polling*: each participant starts with $+1$ or -1. The goal of the protocol is to privately and accurately compute the sum of the initial values despite malicious participants trying to discover votes or bias the computation. Basically, users would not want their vote to be revealed to other participants, nor do they want the polling to be actually conducted by a central server.[1]

The motivation of our work is to explore how companion decentralized (server-less) techniques can leverage reputation in social networks. We claim that privacy and accuracy can be obtained in a distributed system involving *respectable participants* by exploiting the very fact that they do care about their reputation within the social network. We propose an approach which, instead of masking (e.g., BFT [1]) or preventing (e.g., using heavyweight cryptography) malicious behaviors, dissuades such behaviors. We assume verification schemes, cross-checking information between participants, along the lines of *accountability* [21, 10], that may affect their reputation. This triggers malicious nodes to behave *rationally*, in the sense that they act as far as their misbehaving is not publicly exposed with probability 1. We prove that using appropriate verifications in accordance with the reputation scheme, a simple decentralized polling protocol can be made both accurate and private despite the presence of malicious nodes. The deliberate simplicity of our polling protocol ensures that subsequent verifications can be easily implemented.

In a nutshell, the protocol works as follows. Participants, clustered in fully connected groups, make use of a simple secret sharing scheme to encode their vote. Then they send the shares of their vote (*ballots*) to proxies, belonging to another group (an office). Each office computes a partial tally that is further broadcasted to all other groups. Each participant eventually outputs the same tally. The polling protocol is fully decentralized and does not assign specific roles to any node.[2] This results in a simple, scalable, and easy to deploy protocol. The spatial complexity of the protocol is $\mathcal{O}(k\sqrt{N})$ and the number of messages sent is $\mathcal{O}(k\sqrt{N})$, k being the *privacy parameter* and N the number of participants.

We bound the impact of malicious participants and trade this with the level of privacy ensured. More specifically, in a system of N nodes with $B < \sqrt{N}$ malicious participants[3], we can choose any integer k such that the probability of revealing a participant's vote is bounded by $(B/N)^{k+1}$ and the impact on the result is bounded by $(6k+2)B$. This is due to the simple secret sharing scheme enabling to expose, with probability 1, malicious nodes affecting the outcome more than $6k+2$ with only public verifications i.e., without requiring to reveal the participants' vote. As we show in the paper, private verifications expose, with a non-null probability, malicious nodes further (i.e., even if their impact is less than $6k+2$), but require to inspect the content of a subset of ballots. In any case, the reputation of honest nodes is never tarnished.

To illustrate, consider a system of $10,000$ nodes with 99 malicious nodes ($\lceil\sqrt{N}\rceil - 1$) and assume a proportion α of nodes voting $+1$. Setting $k = 1$ ensures privacy with probability 99.99% and for any $\alpha > 0.54$, 100% of the participants compute the right binary decision (i.e., the sign of the outcome). While e-voting would require stronger guarantees, this amply fits polling applications requirements.

We report on the deployment of our protocol over 400 PlanetLab nodes with several values of k. The polling result suffers a relative error of less than 10% in the face of message losses, crashes and asynchrony inherent in PlanetLab. In presence of malicious nodes, our experiments show that the impact on the polling result is $(4k+2)B$ on average, consistently lower that the theoretical bound of $(6k+2)B$.

[1] For instance, every information entered by a Facebook user stays on a Facebook server and may be visible, even after the user deleted it from her profile and deactivated her account.

[2] Note that a central entity may be used for bootstrapping or specific verification procedures.

[3] This is reasonable in sybil-free systems: e.g., Yu *et al.* presented efficient techniques for preventing sybil attacks in social networks [23, 22].

The rest of the paper is organized as follows: Section 2 describes our system model. Section 3 gives a detailed description of our polling protocol together with an analysis of its characteristics. The impact of malicious nodes is presented in Section 4 assuming verifications. Experimental results from PlanetLab are reported in Section 5. Section 6 reviews related work and Section 7 concludes the paper and proposes perspectives on future work.

2 System Model

We consider a system of N uniquely identified nodes. Each node p votes for a binary value $v_p \in \{-1, +1\}$ and the expected output of the polling algorithm is $\sum_p v_p$. Each participant in the social network is assigned an integer reputation score, affected by the behavior in the polling protocol.

Nodes can either be honest or malicious. Honest nodes strictly follow the protocol and contribute to the verifications as long as their privacy is not compromised. More specificaly, honest nodes always collaborate with verification procedures that do not require them to reveal their vote (i.e., public verifications).[4] Malicious nodes may misbehave either to promote their opinion or reveal the opinion of honest nodes. Yet, they are rational in the sense that they never behave in such a way that their reputation is tarnished with probability 1. We consider colluding malicious nodes as a single coalition $\mathcal{B}$ ($|\mathcal{B}| = B$). When malicious nodes collaborate to bias the outcome of the poll, they are assumed to share the same opinion. Still, they act selfishly in the sense that they prefer protecting their own reputation rather than covering up their suspected accomplices. For the sake of readability, we consider that the coalition always promotes -1 in the rest of the paper. Note that a single coalition represents the worst case scenario.[5].

We assume a secure channel between every two nodes that maintains the secrecy of messages. More specifically, a malicious node cannot eavesdrop over the channel to gain information exchanged by others. Theoretical analysis (Section 3.3) assumes reliable channels and non-faulty nodes. We revisit these assumptions by measuring the impact of message losses and crashes in the implementation (Section 5) and by evaluating the impact of crashes in Appendix B.

We assume a cluster-ring-based structure, which could be provided by the social network infrastructure. The N nodes are clustered into r ordered groups, from g_1 to g_r. A node p in group g_i maintains two sets of nodes: a set $\mathcal{P}_o$ of *officemates* containing all nodes belonging to the same group ($\mathcal{P}_o = g_i \backslash \{p\}$) and a fixed-size set $\mathcal{P}_p$ of *proxies*, containing nodes in the next group ($\mathcal{P}_p \subseteq g_{i+1}$). Therefore, all groups virtually form a ring, g_1 being the successor of g_r. Each group g_i is a clique. We define a *client* of a node p, a node for which p acts as a proxy. Every node maintains a list of its clients in the previous group ($\mathcal{P}_c \subseteq g_{i-1}$). A node discards every message originating from a node that is not in $\mathcal{P}_c \cup \mathcal{P}_o$ and can decide when it has received all expected messages. We assume a uniform distribution of nodes across the r groups: nodes in the successor groups are distributed uniformly as proxies in the predecessor group. Therefore, the groups are of similar sizes. In addition, the number of clients $|\mathcal{P}_c|$ of all proxies is approximatively the same, since the proxy views are uniformly distributed and their size $|\mathcal{P}_p|$ is fixed.

3 The Polling Protocol

We first give an overview of the polling protocol. Then we prove the correctness of the protocol and provide a theoretical analysis of its complexity, considering no malicious nodes. Finally, we analyze how the protocol resists to a privacy attack. Attacks on the outcome of the poll are discussed in Section 4.

3.1 Polling in a nutshell

The polling protocol is composed of three phases: *(i)* voting, *(ii)* counting and *(iii)* broadcasting. We overview these phases in the following. During the voting phase, a node generates a set of ballots reflecting its opinion and sends each ballot to one of its proxies. In the counting phase, each node in a group computes the sum of the votes of the nodes in the previous group (local tally). This is achieved by having each proxy summing up the ballots it received and broadcasting the result to its officemates.

[4] However, they may refuse to reveal their vote for a verification procedure (i.e., private verification). This is further discussed in Section 4.2.

[5] Two or more independent coalitions promoting the same opinion have a greater impact when merged. Several coalitions promoting opposite opinions have compensating effects.

Finally, the local tallies are forwarded along the ring so that all nodes eventually compute the final outcome.

Algorithm 1 Polling algorithm at node p in group g_i, $i \in \{1, \ldots, r\}$

Input: a vote $v \in$ {-1, +1}
Variables: an individual tally $t'' = 0$
a local tally $t' = 0$
an array of local tally sets $\mathcal{S}[\{1, \ldots, r\} \to \varnothing]$
a local tally array $\mathcal{T}[\{1, \ldots, r\} \to \perp]$
Output: the global tally $\hat{t}$

Polling Algorithm

1: vote(v, $\mathcal{P}_p$)
2: local_count(t'', $\mathcal{P}_p$)
3: $t' = t' + t''$
4: local_tally_broadcast(i, t'', $\mathcal{P}_p$)
5: $\hat{t} = \sum_i \mathcal{T}[i]$

Voting phase

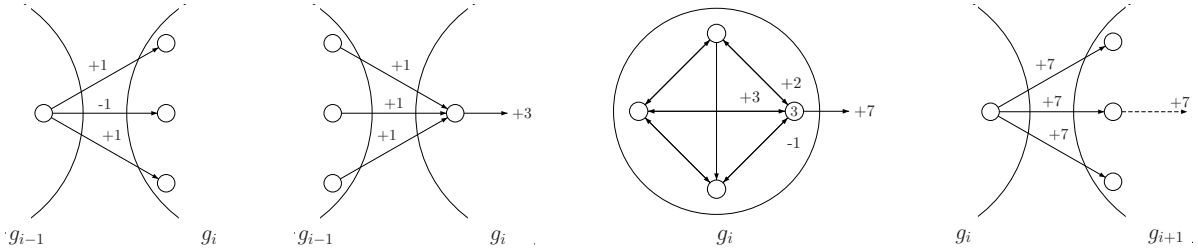
procedure vote(v,$\mathcal{P}_p$) **is**
6: $b = v$
7: **for each** $proxy \in \mathcal{P}_p$ **do**
8: send [BALLOT, b] ($proxy$)
9: $b = -b$
10: **end for**
upon event ⟨ receive | [BALLOT, b] ⟩ **do**
11: $t'' = t' + t$

Intermediate Counting phase

procedure local_count(t'',$\mathcal{P}_o$) **is**
12: **for each** $officemate \in \mathcal{P}_o$ **do**
13: send [INDIVIDUALTALLY, t''] ($officemate$)
14: $b = -b$
15: **end for**
upon event ⟨ receive | [INDIVIDUALTALLY, t] ⟩ **do**
16: $t' = t' + t$

Local Tally Broadcasting & Forwarding phase

procedure local_tally_broadcast(i, t'', $\mathcal{P}_p$) **is**
17: **for each** $proxy \in \mathcal{P}_p$ **do**
18: send [LOCALTALLY, i, t'] ($proxy$)
19: **end for**
upon event ⟨ receive | [LOCALTALLY, i_{group}, t] ⟩ **do**
20: $\mathcal{S}[i_{group}] = \mathcal{S}[i_{group}] \cup \{t\}$
21: **if** ($\mathcal{S}[i_{group}] = |\mathcal{P}_c|$) **then**
22: $\mathcal{T}[i_{group}] =$ choose($\mathcal{S}[i_{group}]$)
23: local_tally_broadcast(i,$\mathcal{T}[i_{group}]$)
24: **end if**
function choose($\mathcal{A}$) **returns** local tally **is**
25: **return** the most represented local tally in $\mathcal{A}$



(a) Voting (b) Intermediate counting (c) Individual tally sharing (d) Local tally broadcasting & forwarding

Figure 1: Key phases of the polling algorithm. In Figure 1(a), a node in g_{i-1} generates 3 ($k = 1$) ballots $\{+1, -1, +1\}$ and sends them to its proxies in g_i. Figure 1(b) depicts a node in g_i collecting its received ballots $\{+1, +1, +1\}$, and summing them to +3 (individual tally) that it shares with its officemates in g_i as depicted in Figure 1(c). Figure 1(c) also shows that when a node received every expected individual tallies $\{-1, +3, +2\}$ it computes and sends the local tally (+7) to the next group g_{i+1}. Finally, Figure 1(d) shows that when this node sends its local tally (+7) to its proxies, these proxies forward the local tally to theirs in g_{i+2}.

3.2 Description

3.2.1 Voting

The ballot generation is inspired by the simple secret sharing scheme introduced in [4]. In order to vote for a given value $v \in$ {-1, +1}, a node generates $2k + 1$ *ballots* $\{b_1, \ldots, b_{2k+1}\} \in$ {-1, +1} representing its vote, where k is an integer called privacy parameter. The intuition is to create $k + 1$ ballots of a given tendency (positive or negative) and k opposite ballots, resulting, when summed, in a single vote $v' = v$. Each ballot is defined as $b_j = v$ if j is odd and $b_j = -v$ if j is even, so that $v' = \sum_{j=1}^{2k+1} b_j = v$.

Once a node has generated its $2k + 1$ ballots, it sends each of them to a different proxy. The number of proxies is to be chosen accordingly, $|\mathcal{P}_p| = 2k + 1$ (for all nodes). Lines 6–9 in Algorithm 1 detail the voting phase. Figure 1(a) depicts a node sending its $2k + 1$ ballots (e.g., $\{+1, -1, +1\}$) to its assigned proxies. Once every node in the system has received one ballot from each of its clients, the voting round is over.

3.2.2 Intermediate Counting

A group acts as an office for the preceding group on the ring. The officemates collect ballots from their clients (Figure 1(b)) and share intermediate results (Figure 1(c)). To this end, a proxy sums the ballots it received into an *individual tally* t'', as described in Algorithm 1, line 11. Once a node has received the expected number of ballots from its clients, it broadcasts the computed individual tally to its officemates, as depicted in Figure 1(b) (lines 12–15 in Algorithm 1). The officemates aggregate the received data, i.e., they sum each others' individual tallies and store the result into a *local tally* t' as shown in Figure 1(c) (line 16 in Algorithm 1).

3.2.3 Local Tally Forwarding

Once the intermediate counting phase is over, i.e., all the officemates have computed a local tally, each node in the group sends its local tally to its proxies (lines 17–19 in Algorithm 1). Upon reception of a message containing a local tally, a proxy adds it to the set $\mathcal{S}[i]$ of possible values for g_i. When a proxy has received the expected number $|\mathcal{P}_c|$ of local tallies for a given group g_i, it decides on a local tally by choosing the most represented value in $\mathcal{S}[i]$ and stores it in $\mathcal{T}[i]$. When a local tally $\mathcal{T}[i]$ is assigned, it is further forwarded (Figure 1(d)) to the next group using the proxies (lines 20–24 in Algorithm 1). Local tallies are then forwarded in the system along the ring. When a local tally reaches its source (the group that emitted it), it is no longer forwarded. The global tally is computed at each node by simply summing the local tallies of all groups: $\hat{t} = \sum_{i=1}^{r} \mathcal{T}[i]$ (line 5 of Algorithm 1).

3.3 Analysis

We analyze the correctness of the polling algorithm and its complexity assuming no malicious nodes. We then consider the impact of the malicious nodes on privacy. The impact of malicious attacks on the accuracy of the polling is presented in Section 4 assuming verification procedures.

Theorem 1 (Correctness)**.** *Assume a system where each node p starts with a binary value v_p. The polling algorithm terminates and each node eventually outputs $\sum_p v_p$.*

Proof. (Accuracy) We first prove that the local tally computed in every group g_i reflects the vote of all nodes in g_{i-1}. The local tally computed in a group is the sum of the ballots received by its members. Each node p in g_{i-1} sends each of its ballots $b_{1,p}, \ldots, b_{2k+1,p}$ to one distinct proxy in g_i. Similarly, each proxy p' in g_i receives a set of ballots $\mathcal{B}_{p'}$ from its clients. Since we assume no malicious nodes, the set of ballots sent by the nodes in g_{i-1} equals the set of ballots received in g_i. Therefore, the local tally t' computed by each member of g_i is

$$t' = \sum_{p' \in g_i} \sum_{b \in \mathcal{B}_{p'}} b = \sum_{p \in g_{i-1}} \sum_{j=1}^{2k+1} b_{j,p} = \sum_{p \in g_{i-1}} [(k+1) \cdot v_p + k \cdot (-v_p)] = \sum_{p \in g_{i-1}} v_p \ .$$

Note that this follows from the homomorphic property of the simple secret sharing scheme [4]. Since nodes do honestly forward the local tallies along the ring and the messages are eventually received, each node ends up with the correct values for the local tallies of every group, thus the correct global tally.

(Termination) A node knows the number of messages it is supposed to receive during each phase. Since every node sends the required number of messages and every message eventually reaches its destination, each phase completes. As the polling algorithm is a finite sequence of phases, it is guaranteed to eventually terminate. □

In a realistic scenario, crashes and message losses do occur and may affect the correctness and the termination. Failures of nodes sending ballots or individual tallies and message losses may *(i)* affect the accuracy of the global tally, and *(ii)* prevent node from detecting the end of a phase or deciding on a local tally (and thus computing the output). More details are given in Appendix B.

Proposition (Spatial complexity)**.** *The asymptotic size S of the state maintained at each node in group g_i is $\mathcal{O}(r \cdot k + |g_i|)$.*

Proof. Every node maintains a set of proxies ($2k+1$), the set of its officemates ($|g_i|$) and the list of its clients ($2k+1$ clients on average). Additionally, every node stores a set of $2k+1$ possible values for each of the r local tallies to perform the global counting, that is $S = \mathcal{O}(k) + \mathcal{O}(|g_i|) + \mathcal{O}(k) + \mathcal{O}(r \cdot k) = \mathcal{O}(r \cdot k + |g_i|)$. □

Proposition (Message complexity)**.** *The asymptotic average number of messages M sent by a node in group g_i is $\mathcal{O}(r \cdot k + |g_i|)$.*

Proof. A node sends messages during the voting phase ($2k+1$ ballots), the intermediate counting phase ($|g_i| - 1$ individual tallies), and the global counting phase which involves the dissemination of r local tallies along the ring using its $2k+1$ proxies. That is $M = \mathcal{O}(k) + \mathcal{O}(|g_i|) + \mathcal{O}(r \cdot k) = \mathcal{O}(r \cdot k + |g_i|)$. □

Note that the parameters are not independent: the sizes of the groups are related and bound to the number of groups by the relation $\sum_{i=1}^{r} |g_i| = N$. Those two quantities are minimized when $r = |g_i|$, $\forall i \in \{1, \ldots, r\}$. That is: $\forall i \in \{1, \ldots, r\}\ |g_i| = r = \sqrt{N}$, $S = \mathcal{O}(k\sqrt{N})$ and $M = \mathcal{O}(k\sqrt{N})$.

Theorem 2 (Privacy)**.** *Considering a coalition of B malicious nodes, the probability of violating a node's privacy is bounded by $(B/N)^{k+1}$.*

Proof. The vote of a node is revealed if and only if the $k+1$ proxies that received the $k+1$ ballots containing the most represented value belong to the malicious coalition. This event occurs with probability $\binom{B}{k+1}/\binom{N}{k+1}$. For all k, B and N, this probability is bounded by $(B/N)^{k+1}$. The bound is tight for $B, N \gg k$. □

4 Confining the impact of malicious nodes

In this section, we analyze the impact of malicious nodes on the polling protocol, assuming companion verification schemes to detect attacks and identify guilty nodes. Detection is performed by the nodes themselves relying on some of them being honest, and verifications are performed by an external entity (e.g., the social network infrastructure) polling nodes to identify the guilty ones. These verification procedures are either sporadic, following real-life tax verification schemes, or triggered by nodes. We distinguish two types of verifications: *(i)* public verifications that leverage only information that does not compromise the nodes' privacy (i.e., the content of the ballots they sent), such as the individual tallies received from their officemates, and *(ii)* private verifications that may leverage every information logged by the nodes including the content of the ballots.

To dissuade nodes from misbehaving, verifications must affect the reputation of the involved nodes. Initially, every node in the system is assigned the same reputation score (given in *reputation units*, ru). When an attack is detected and reported, the neighbors (i.e., the nodes it communicates with, typically clients and proxies) of the accused nodes are polled for the messages they exchanged. The reputation of the involved nodes is affected as follows: if the inspected node confesses the action that was reported as suspect (i.e., confirms the testimonies of its neighbors), then only its reputation is decreased by 1 ru. Otherwise, the reputation of both involved nodes is decreased by 1 ru.

We consider an overlay of $\sqrt{N}$ groups of size $\sqrt{N}$ ($\sqrt{N} \in \mathbb{N}$) and a perfect client/proxy matching, i.e., each node has exactly the same number ($2k+1$) of clients and proxies.

Assuming honest nodes give priority to privacy over accuracy, we focus in a first step on bounding the impact of colluding malicious nodes when using only public verifications. Then, considering also private verifications, we prove that colluding nodes compromising the global tally within the bound may be caught with a non-null probability. We then discuss independent malicious nodes.

4.1 Impact of a malicious coalition under public verifications

Theorem 3. *For $B < \sqrt{N}$, each member of a malicious coalition may affect the global tally up to $6k+2$.*

Proof (structure). The proof relies on the facts that *(i)* the honest nodes always tell the truth and strictly follow the protocol (including detections and verifications), and *(ii)* malicious nodes do not behave in such a way that their reputation is decreased with probability one. Effectively, showing that the attacks with

unbounded impact are detected by the honest nodes with probability 1 proves the theorem. A malicious node may bias the protocol at all three phases. Lemmas 3.1-3.4 encompass all possible attacks, propose a detection scheme relying on honest node, and bound the impact of those that cannot be detected with probability 1. In addition, if an attack is detected, we prove that the guilty node is exposed by a public verification. Summing the maximum impacts of all these attacks (Lemmas 3.1 and 3.2) for each malicious node gives a maximum impact $(2k + 2(2k + 1)) = (6k + 2)$ on the global tally.

Note that the proof relies on the assumption $B < \sqrt{N}$ to ensure that the malicious can neither "control" (there is at least one honest node in each group) nor "fool" an entire group without being detected (i.e. there are not enough malicious node to both perform and cover malicious actions). In fact, the weakest assumption needed is $\forall i, |g_i \cap \mathcal{B}| + |g_{i+1} \cap \mathcal{B}| < \sqrt{N}$. □

Lemma 3.1 (Voting)**.** *A malicious node can affect the global tally up to* $2k$*, when voting.*

Proof. Due to the overlay structure, a node can only send ballots to the proxies it is assigned (otherwise the ballots are discarded), i.e., a maximum of $2k + 1$ ballots. Therefore a malicious node may affect the global tally by either *(i)* sending less ballots than it is supposed to or by *(ii)* sending more than $k + 1$ -1-ballots. In the worst case, the malicious node sends $2k + 1$ -1-ballots. Since a node is allowed to send only a set of ballots that sums to -1 or $+1$, its maximum impact is $2k$. Note that this attack can be only detected by inspecting the content of the ballots sent by the malicious node, this is a private verification. □

Lemma 3.2 (Computing individual tallies)**.** *There exists a public verification scheme so that, if a malicious node modifies its individual tally by more than* $2(2k+1)$*, the attack is detected with probability* 1 *and the node is exposed.*

Proof. The considered overlay structure ensures that any node has exactly $2k+1$ clients and thus receives $2k + 1$ ballots during the voting phase. A malicious node can modify its individual tally by inverting the +1-ballots it received, i.e., it turns them in -1-ballots, decreasing its individual tally accordingly. In addition, a malicious node can also forge ballots. While the first attack cannot be detected without inspecting the content of the ballots themselves, the latter is identified by its honest officemates with probability 1 if the individual tally is out of the range $[-(2k + 1), 2k + 1]$. Therefore, not to be publicly detected, a node corrupting or forging ballots must output an individual tally in that range. Consequently, the worst case occurs when a malicious node receives $2k + 1$ +1-ballots and inverts them all when summing them, leading to a maximum impact of $2(2k + 1)$. When verified, a malicious node outputting an individual tally out of the range is automatically exposed. □

Lemma 3.3 (Broadcasting individual tallies)**.** *There exists a public verification scheme so that, a malicious node broadcasting inconsistent copies of its individual tally to honest nodes, i.e., sending different values to its honest officemates, is detected with probability* 1 *and the node is exposed.*

Proof. Before deciding on a local tally, every node broadcasts to its officemates the set of individual tallies it received. This way, a honest officemate trivially detects the inconsistency. Malicious nodes are exposed when their neighbors are asked for the individual tallies they received from these nodes. □

Note that a malicious node has no incentive not to broadcast its individual tally since it would be publicly detected, whereas sending an individual tally of 0 would have the same impact on the computed local tally while not being publicly detectable. However, broadcasting different individual tallies may help a malicious node to impose an arbitrary value for the local tally. For instance, assume a proxy has 5 clients, i.e., $k = 2$, only two of them being malicious. In that case, there is a majority of honest nodes. Consequently, if the honest nodes send the same local tally, it will be the one chosen by the proxy. However, if the malicious nodes send different values as their individual tallies then honest nodes will compute different local tallies. The proxy will then decide on the arbitrary local tally sent by the malicious nodes.

Lemma 3.4 (Forwarding local tallies)**.** *There exists a public verification scheme so that, a group forwarding inconsistent copies of a local tally, i.e., nodes sending different values to their proxies, is detected with probability* 1 *and the malicious nodes are exposed.*

Proof. An inconsistent local tally forwarding is detected assuming the following: before deciding on a local tally, a node broadcasts the set of received local tallies to its officemates. An inconsistency is detected if at least one of the following conditions is satisfied: (C1) a honest node received different local tallies from its clients, (C2) a honest node received different local tallies than its officemates. Consider j malicious nodes concentrated in a group g_i forwarding an incorrect local tally to their proxies. Because of (C1), the clients of a honest node in g_{i+1} must all be malicious. Since the number of clients of all nodes equals their number of proxies $(2k+1)$, a coalition of j malicious nodes can corrupt a maximum of j proxies. Therefore, the $\sqrt{N}-j$ remaining proxies in g_{i+1} must collude with the coalition in g_i to circumvent (C2). To conclude, not to be detected, such an attack requires j malicious nodes in g_i and $\sqrt{N}-j$ malicious nodes in g_{i+1}, that is a minimum number of $\sqrt{N}$ malicious nodes. A malicious node in g_i can be exposed by a public verification scheme since it broadcasted a local tally that does not correspond to the sum of individual tallies it received. A malicious node in g_{i+1} can be exposed by a public verification scheme since it has broadcasted a different local tally from the one it received. □

Theorem 3 shows that the protocol trades accuracy for privacy. Effectively, the privacy parameter k adjusts accuracy and privacy in opposite directions. Increasing the number of messages to represent a vote, i.e., increasing k, gives an opportunity $6k+2$ to each malicious node to corrupt the outcome of the result. Conversely, the probability of a vote to be disclosed decreases following $(B/N)^{k+1}$.

4.2 Private verifications

So far we only considered public verifications, i.e., where the content of the ballots is never disclosed. Assume now that the nodes accept, with a non-null probability, to relax privacy for the sake of verifications and reveal *a subset* of the ballots they sent and/or received. Then, this partial information can be leveraged to detect the malicious behaviors described in Lemmas 3.1 and 3.2.

Theorem 4. *There is a non-null probability to expose a malicious node when misbehaving, even if its impact is less than* $6k+2$.

Proof. Regarding the voting phase, a malicious node that sent $k+1+j$ +1-ballots and only $k-j$ -1-ballots $(1 \leq j \leq k)$ is unable to provide, for both kinds, the id of $k-j+1$ proxies to which it sent a -1-ballot. Therefore, a simple verification is to ask the suspected node to provide a list of proxies that

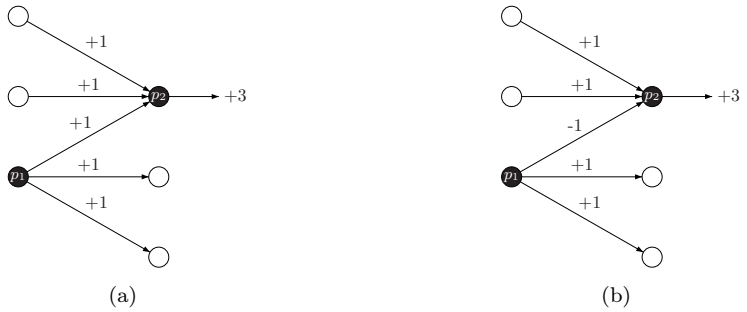


Figure 2: Malicious nodes have no gain in covering up each other.

can testify it sent at least j' ballots of each kind, for a random value j' ranging from 1 to k. Note that an inspected node can disclose $j'=k$ without revealing its vote. Regarding the ballot corruption attack, partial information about the ballots received by the inspected node can be leveraged to refine the bound on its individual tally: assume the inspected node received n_b ballots, if we further know that it received at least n_b^+ +1-ballots and n_b^- -1-ballots, then the range is refined from $[-n_b, n_b]$ to $[-n_b+2n_b^+, n_b-2n_b^-]$. In both the aforementioned verifications, a malicious node has no interest in covering up another one. Consider the examples depicted in Figure 2 where a malicious node p_1 is the client of a malicious node p_2. In the case represented in Figure 2(a), if p_1's vote is verified and p_2 covers p_1 up, i.e., it testifies p_1 sent a -1-ballot, it exposes itself to a private verification on its individual tally (note that a node has to be consistent from one verification to another, thus, if the vote of p_2 is further verified, p_2 must stick to its first version about the ballots it received). The same situation occurs in the case depicted in

Figure 2(b): if p_2's individual tally is verified and p_2 covers p_1 up, i.e., it testifies it sent a -1-ballot to p_2, it puts itself at risk, should it be subject to a private verification on its vote.

Since malicious node are selfish, they never cover up malicious nodes when privately verified. In conclusion, relaxing privacy ensures that every malicious node has a non-null probability to be exposed. □

Theorem 5. *The reputation of honest nodes is never tarnished.*

Proof. The reputation of a honest node p_1 can only be decreased by two means: *(i)* a node p_2 falsely accuses p_1, or *(ii)* a node p_2 accused by p_1 does not confess its illegal action. Since a honest node always tells the truth, both cases occur only if p_2 is malicious. The first situation neither increases the interest function of p_2 nor decreases the final result of the poll. The second situation decreases the reputation of p_2 by 1 ru with probability 1 and p_2 is not cleared for any other verifications even so. Since p_2 is rational, it never behaves in such a way. Therefore, the reputation of a honest node is never tarnished. □

4.3 Impact of independent malicious nodes

So far we considered that malicious nodes collude to achieve their common goal. However, when malicious nodes are independent, the proposed protocol can be refined as follows: the members of each group g_i share a binary secret $m_i \in \{-1, 1\}$. When voting, a node *modulates* the ballots it sends by multiplying each of them by m_i. The secret of each group is revealed only when the poll is over. This limits the interest of corrupting ballots/tallies, the impact being unpredictable. Nodes determine the global tally by computing $\sum_i m_i \mathcal{T}[i]$ (note that the outcome of the poll remains correct since the sum of the sum of modulated ballots is equal to the modulated sum of the original ballots). As an example, a malicious node in group g_i being a proxy for nodes in g_{i-1} knows the modulation m_i but ignores the modulation m_{i-1} of its clients, therefore, when corrupting the ballots, it cannot know what impact it will have on the final result. A malicious node can only predict the impact of the ballots it sends, i.e., when the node votes. Its impact is therefore limited to $2k$. This mechanism cannot work when malicious nodes collude since they could tell each other the shared secret of their respective groups.

5 Implementation

The goal of the evaluation is to assess the algorithm against the presented theoretical bounds, and the impact of relaxing the assumptions made for the sake of the analysis. Our experiments show that, in a practical setting, *(i)* our polling protocol suffers only as low as 10% in accuracy, and *(ii)* the average impact measured for given sizes of coalition is around $4k \cdot B$.

5.1 Experimental setup

In this section we report on the deployment of our polling algorithm on a 400 PlanetLab nodes testbed. This enables to stretch the algorithm in a real world setting, *(i)* in the presence of message losses, crashes and asynchrony inherent in PlanetLab, and *(ii)* when attacking the protocol by introducing malicious nodes. We evaluate our algorithm with two different privacy parameter values $k = 1$ and $k = 2$.

5.1.1 Overlay

The cluster-ring-based overlay is built using a centralized bootstrapping entity that keeps track of the whole set of nodes, assigning each node to a random group using the hashes of their ids. Nodes have exactly $2k + 1$ proxies in the next group and the number of clients of a node is $(2k + 1)\,|g_{i-1}|\,/\,|g_i|$ on average. Note that although we opted for a centralized overlay construction algorithm (e.g., the social network infrastructure could handle this task), we believe it can be achieved with the help of Fireflies building blocks [11] to prevent malicious nodes to concentrate in groups (which we have shown is of the utmost importance for privacy) and Kelips [8] in order to create groups of similar sizes. Other DHT-based approaches could also provide similar properties [18, 15, 20].

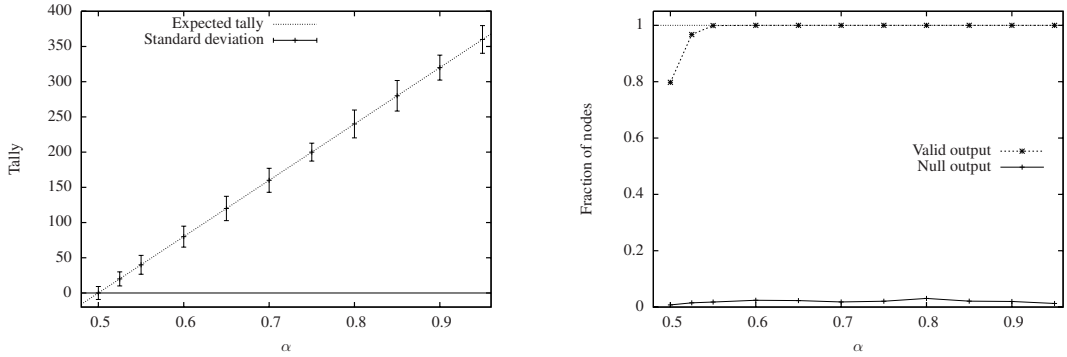
5.1.2 Communication and Asynchrony

Nodes communicate through UDP which may lead to message losses on the communication channels (on the PlanetLab testbed, we observed a loss rate ranging from 5% to 15%). In addition, PlanetLab nodes are unreliable leading to expected messages to be lost due to senders crashes. Therefore, in practice, phase terminations cannot be detected by simply counting the number of received messages. To address this issue, all phases are bounded in time, except the local tally forwarding phase. In this phase, when the number of possible values for a local tally of a group grows beyond a given threshold $\gamma \cdot |\mathcal{P}_c|$, the node schedules the decision on this particular local tally in Δt seconds. In our implementation, γ is set to 0.5 and Δt to 5 seconds.

5.2 Polling in practice

5.2.1 Accuracy

Figure 3 depicts the accuracy of the polling algorithm among 400 PlanetLab nodes with $k = 2$. Figure 3(a) considers the value of the tally while Figure 3(b) considers the sign of the tally. Without loss of generality, we consider a proportion α of node voting +1 ranging from 0.5 to 1. In Figure 3(a), we plot the average standard deviation on the computed tally for α in that range. For each run, we compute the average of the error when computing the tally (this is the difference between the tally on each node and the real one) over all nodes. Each point represents the average of this value over 20 independent runs. Note that the accuracy increases when α is close to 0.5. This is due to the fact that the closer the tally to 0.5, the less message losses impact the outcome: the closer the number of -1-ballots and +1-ballots, the closer to 0 the individual and local tallies.



(a) (b)

Figure 3: Accuracy of the algorithm in presence of asynchrony, message losses and failures.

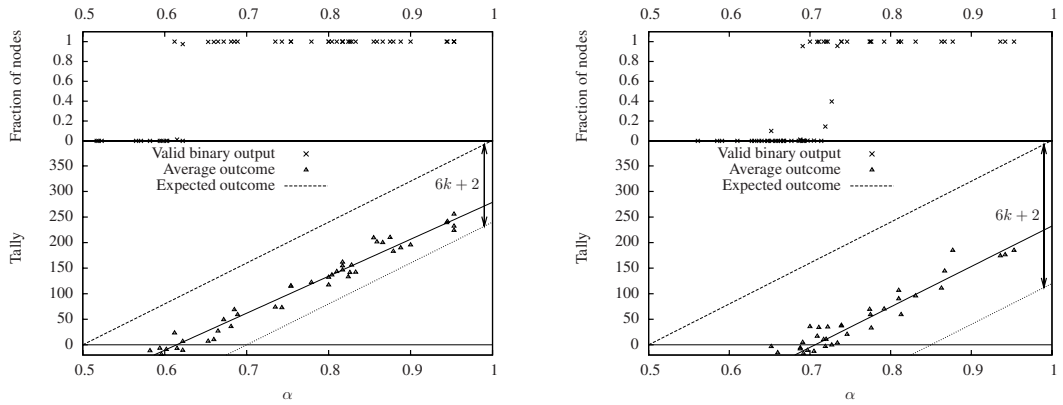
Figure 3(b) displays the fraction of nodes deciding on the correct *sign* of the tally depending on α (dashed line). Effectively, even if the standard deviation is relatively small, some nodes may decide incorrectly on the sign of the outcome. Consider the organizers of a Saturday night party asking their friends in a social network whether partners should be excluded. As depicted in Figure 3(b), for $\alpha = 52.5\%$, some nodes would compute a different answer than the majority. This means that a minority of participants computing a negative result would come with their partners... Figure 3(b) (plain line) also shows the proportion of nodes that are unable to decide on a global tally (because their set of possible values never reach the threshold γ). We observe that this fraction remains very low (less than 4%) and is independent from α.

The impact of message loss varies depending on the phase of the polling protocol. During a voting phase, there is no recovery of ballots *a priori*. Note that the impact on the final outcome of a single ballot is limited to 1. From the intermediate counting phase onwards, the impact may be greater as ballots are aggregated. Yet, the impact is limited by the natural replication inherent in broadcast within and between groups. For this reason, the proposed algorithm intuitively suffers less than raw population protocols [2] or tree-based protocols for aggregation.

5.2.2 Malicious Attacks

We evaluate the protocol in the presence of a coalition of B malicious nodes that bias the outcome of the polling result in an attempt to decrease the final outcome. We consider the worst case: malicious nodes do every possible attack that does not compromise their reputation with probability 1, i.e., every malicious node *(i)* sends $2k+1$ -1-ballots, and *(ii)* inverts every +1-ballot it receives. Figure 4 displays for $k=1$ and $k=2$, the resulting tally (sign on the upper part of the figure and value on the lower part), compared to the real one (dashed line), for $B=19$ malicious nodes ($B=\lceil\sqrt{N}\rceil-1$) in a 400 node system.

We observe that, as expected, the impact of the malicious nodes is greater for $k=2$ than for $k=1$. Effectively the higher k, the better privacy can be preserved but the higher the impact of malicious nodes. We observe that the malicious coalition affects the outcome of the poll within the theoretical bound derived in the analysis (dotted lines in Figure 4). However, the average impact of the coalition is less than $2k+2(2k+1)=6k+2$ (considering the worst case where the malicious proxy receives only +1-ballots and inverts them all). The theoretical bound is never reached as the average impact of a malicious node depends on the actual number of ballots it can invert, this, in turn, depends on α. Effectively, a malicious proxy receives $k+\alpha$ +1-ballots out of $2k+1$ on average. Therefore, its impact is $2k+2(k+\alpha)=4k+2\alpha$ on average. For $k=2$, fitting our 55 data point cloud with a least-squares regression line (plain line in Figure 4) $a(\alpha-0.5)+b$ gives $a=791$ and $b=-163$. This is close to the expected parameter values $a=2(N-B)=760$ and $b=-B(4k+1)=-180$. See Appendix A for more details. We use this analysis to make a projection on larger scale systems. For $k=1$ (Figure 4(a)), every node of the poll outputs a valid binary results when $\alpha>0.62$, which is to be compared to $\alpha>0.55$ observed in Figure 3(b) (without malicious nodes). Note that the threshold decreases with N. On average, we can analytically derive that with $N=10,000$ and $B=99$, the proportion α for which the nodes decides correctly is $\alpha>0.52$.



(a) $k=1$ (b) $k=2$

Figure 4: Accuracy of the poll in presence of malicious nodes: with $N=400$ and $B=19$ malicious nodes manage to confuse the majority of the nodes for *(a)* $\alpha<0.62$ when $k=1$, and *(b)* $\alpha<0.73$ when $k=2$.

6 Related Work

We discuss here related distributed protocols which involve neither cryptography nor security scheme based on intractability of mathematical computations. Similarly to most protocols without cryptography, our work ensures privacy via secret sharing techniques. Also, our solution distances itself from related work in the sense that no participant has a special role (following the peer-to-peer paradigm), resulting in an increased scalability and robustness.

A large amount of work on secret sharing schemes (introduced by Rivest *et al.* in [16]) has been published in the late 80's. In [3], Benaloh proposed a scheme for sharing secrets privately based on polynomials. Benaloh's scheme relies on the homorphic property of addition, to implement polling.

However, a malicious participant can easily corrupt the shares, thus potentially significantly impacting the final outcome.

Assuming a majority of honest participants, Rabin and Ben-Or extended Benaloh's secret sharing and proposed *verifiable secret sharing scheme* (VSS) [14]. Based on VSS, they proposed a private multiparty protocol to compute the sum of the participants' inputs. Beyond the fact that these techniques assume a fully connected network, synchronous links and broadcast channels, they involve higher mathematics. In addition they present a major drawback when applied to distributed polling. Effectively, although VSS allows to compute privately the verifiable sum of the participants' inputs with an exponentially small error on the output, there is no control over the inputs themselves. More specifically, a malicious participant may share an arbitrarily high value and can thus affect the outcome in a potentially unbounded way (more precisely, the inputs are in a finite field Z_p where p is a prime number greater than N, thus they are bounded but can be greater than N). The strength of the protocol is to ensure privacy to participants, including the malicious ones. This makes such a scheme hardly suitable for polling applications. Note that this also applies to complex secret sharing scheme and private multiparty computation such as AMPC [13].

In [12], Malkhi *et al.* proposed an e-voting protocol based on AMPC and enhanced check vectors. While powerful, this protocol differs from our work since participants have distinct and predefined roles (dealers, talliers, and receivers). This may result in decreased scalability and robustness if specific nodes fail. Nonetheless, these design choices are fully justified by the requirements inherent in E-Voting applications, such as democracy, verifiability, and unconditional accuracy. Instead polling applications can relax such constraints for the sake of simplicity.

In a sense, our work also relates to distributed ranking schemes. The principle of ranking is similar to polling, in that a participant evaluates the quality of one of her peers by *(i)* locally grading its behavior (input value), and *(ii)* collecting the local grades assigned by the rest of the system. However, to the best of our knowledge, most published work [9, 17] has focused on designing accurate grading mechanism rather than providing efficient polling schemes. In [6], Dutta *et al.* evoke grading freeriders and take into account possible collusion. However, none of the proposed ranking schemes provide a global polling, as grading generally relies on polling only a subset of nodes (peers usually collaborate with a small part of the network). In addition, privacy is usually not addressed in such schemes.

7 Conclusion

In this paper, we considered a simple decentralized polling protocol and proved that it can be made private and accurate in presence of malicious nodes concerned over their reputation (as members of social networks do) by means of verification procedures. This opens the way to a novel and promising way to perform secure distributed computations.

In addition we believe that our work can be extended to distributed applications that are not critical (i.e., that are not sensitive to small deviation on their outcome). We plan to demonstrate the feasibility of decentralized polling of respectable participants by deploying our system to a large scale social network. A natural and interesting perspective is the extension of our polling protocol to n-ary inputs providing doodle-like services[5].

References

[1] L. Alvisi, A. Clement, and M. Dahlin. Making Byzantine Fault Tolerant Systems Tolerate Byzantine Faults. In *NSDI'09*, 2009.

[2] D. Angluin, J. Aspnes, D. Eisenstat, and E. Ruppert. The Computational Power of Population Protocols. *Distributed Computing*, 20(4):279–304, 2007.

[3] J. Benaloh. Secret Sharing Homomorphisms: Keeping Shares of a Secret Secret. In *CRYPTO'86*, pages 251–260, 1986.

[4] C. Delporte-Gallet, H. Fauconnier, R. Guerraoui, and E. Ruppert. Secretive Birds: Privacy in Population Protocols. In *OPODIS'07*, pages 329–342, 2007.

[5] Doodle: Easy Scheduling. http://www.doodle.com.

[6] D. Dutta, A. Goel, R. Govindan, and H. Zhang. The Design of a Distributed Rating Scheme for Peer-to-Peer Systems. In *Workshop on Economics of Peer-to-Peer Systems*, 2003.

[7] Facebook. http://www.facebook.com.

[8] I. Gupta, K. Birman, P. Linga, A. Demers, and R. van Renesse. Kelips: Building an Efficient and Stable P2P DHT through Increased Memory and Background Overhead. In *IPTPS'03*, pages 160–169, 2003.

[9] M. Gupta, P. Judge, and M. Ammar. A Reputation System for Peer-to-Peer Networks. In *NOSSDAV'03*, pages 144–152, 2003.

[10] A. Haeberlen, P. Kouznetsov, and P. Druschel. PeerReview: Practical Accountability for Distributed Systems. In *SOSP'07*, pages 175–188, 2007.

[11] H. Johansen, A. Allavena, and R. van Renesse. Fireflies: Scalable Support for Intrusion-Tolerant Network Overlays. *SIGOPS Operating System Review*, 40:3–13, 2006.

[12] D. Malkhi, O. Margo, and E. Pavlov. E-Voting without 'Cryptography'. In *FC'02*, pages 1–15, 2002.

[13] D. Malkhi and E. Pavlov. Anonymity without 'Cryptography'. In *FC'01*, pages 108–126, 2001.

[14] T. Rabin and M. Ben-Or. Verifiable Secret Sharing and Multiparty Protocols with Honest Majority. In *STOC'89*, pages 73–85, 1989.

[15] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A Scalable Content-Addressable Network. In *SIGCOMM'01*, pages 161–172, 2001.

[16] R. Rivest, A. Shamir, and Y. Tauman. How to Share a Secret. *Communications of the ACM*, 22:612–613, 1979.

[17] M. Rodriguez-Perez, O. Esparza, and J. L. Muñoz. Analysis of Peer-to-Peer Distributed Reputation Schemes. In *COLCOM'05*, pages 1–6, 2005.

[18] A. Rowstron and P. Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In *Middleware'01*, pages 329–350, 2001.

[19] Second Life. http://www.secondlife.com.

[20] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *SIGCOMM'01*, pages 149–160, 2001.

[21] D. Weitzner, H. Abelson, T. Berners-Lee, J. Feigenbaum, J. Hendler, and G. J. Sussman. Information accountability. *Communications of the ACM*, 51:82–87, 2008.

[22] H. Yu, P. Gibbons, M. Kaminsky, and F. Xiao. SybilLimit: A Near-Optimal Social Network Defense against Sybil Attacks. In *SP'08*, pages 3–17, 2008.

[23] H. Yu, M. Kaminsky, P. Gibbons, and A. Flaxman. SybilGuard: Defending Against Sybil Attacks via Social Networks. *Transactions on Networking*, 16(3):576–589, 2008.

A Average impact of malicious nodes

There is a proportion α of nodes voting +1. Therefore the expected final tally is $\alpha N - (1-\alpha)N = N(2\alpha - 1)$. When voting, nodes voting +1 (resp. -1) share their vote in $k+1$ +1-ballots and k -1-ballots (resp. $k+1$ -1-ballots and k +1-ballots). Therefore, proxies receive a +1-ballot with probability $\alpha \cdot (k+1)/(2k+1) + (1-\alpha) \cdot k/(2k+1) = (k+\alpha)/(2k+1)$. Since a proxy receives on average $2k+1$ ballots, the average impact of a malicious proxy promoting -1 is $2(2k+1) \cdot (k+\alpha)/(2k+1) = 2k + 2\alpha$. In addition, a malicious node can affect the tally by $2k$ when voting. Therefore, a malicious node can bias the protocol by $2k + 2k + 2\alpha = 4k + 2\alpha$ on average. Considering a coalition of B malicious node, the expected tally is thus $N(2\alpha - 1) - B(4k + 2\alpha) = 2(N - B)(\alpha - 0.5) - B(4k + 1)$.

B Impact of crashes

Crashes can have two very different impacts on the system. First, nodes may crash independently before sending *unique* information. This refers to data they received, that did not yet get replicated, typically initial shares of votes. Loosing such data affects the global tally. Second, several nodes may be crashed at given time. This may result in other nodes unable to decide on a local tally and thus on the global tally from a lack of corroborating pieces of information. In the following, we analyze the impact of the first type of crashes on the outcome. We then give the probability that the second situation arises.

Impact of individual crashes Consider a node that crashes before broadcasting its individual tally to its officemates. This individual tally is lost, and is the sum of the $|\mathcal{P}_c| = 2k+1$ ballots sent by its clients (assuming perfect client/proxy matching). The impact of such a crash is bounded by $2k+1$ and is $(2\alpha - 1)$ on average (assuming all groups have an equal size). Moreover, if a node crashes while sending its ballots, it affects the global tally by up to k (if the node crashes after sending its $k+1$ identical ballots). The maximum impact of an individual crash is thus $2k + 1 + k = 3k + 1$.

Probability of simultaneous crashes Assuming nodes crash with probability c, there is a probability $\overline{d}_i$ for a node $p \in g_i$ not to decide on the local tally of group r. This can happen if more than $(1-\gamma)\,|\mathcal{P}_c| = (1-\gamma)(2k+1)$ clients do not send the local tally because they either crash or have not decided on the local tally. We define $\overline{e}_i$ as the probability for a node in g_i not to send a local tally. We have $\overline{e}_i = c + (1-c)\overline{d}_i$ and

$$\overline{d}_i = \sum_{j=0}^{\gamma(2k+1)-1} \binom{2k+1}{j} (1-\overline{e}_{i-1})^j \cdot \overline{e}_{i-1}^{2k+1-j} \quad \text{and} \quad \overline{d}_1 = 0 \ .$$

A node does not decide on the global tally if it has not decided on at least one local tally. This happens with probability $\overline{d}_r$ (if the further tally can be forwarded, so does the others). For $k = 2$, $r = 20$ and $c = 5\%$, the probability for a node not to decide on the global tally is 2%.

Centre de recherche INRIA Rennes – Bretagne Atlantique
IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399