Brahms: Byzantine Resilient Random Membership Sampling

Edward Bortnikov* Maxim Gurevich* Idit Keidar*
Gabriel Kliot† Alexander Shraer*

ABSTRACT

We present Brahms, an algorithm for sampling random nodes in a large dynamic system prone to malicious behavior. Brahms stores small membership views at each node, and yet overcomes Byzantine attacks by a linear portion of the system. Brahms is composed of two components. The first one is a resilient gossip-based membership protocol. The second one uses a novel memory-efficient approach for uniform sampling from a possibly biased stream of ids that traverse the node. We evaluate Brahms using rigorous analysis, backed by simulations, which show that our theoretical model captures the protocol's essentials. We study two representative attacks, and show that with high probability, an attacker cannot create a partition between correct nodes. We further prove that each node's sample converges to a uniform one over time. To our knowledge, no such properties were proven for gossip protocols in the past.

Keywords

Random sampling, gossip, membership, Byzantine failures

1. INTRODUCTION

We consider the problem of sampling a random node (or peer) in a large dynamic system subject to adversarial (Byzantine) attacks. Random node sampling is important for many scalable dynamic applications, including neighbor selection in constructing and maintaining overlay networks [15, 22, 25, 27], selection of communication partners in gossip-based protocols [6, 10, 13], data sampling, and choosing locations for data caching, e.g., in unstructured peer-to-peer networks [24].

Typically, in such applications, each node maintains a set of random node ids that is asymptotically smaller than the system size. This set is called a *local view*. In a dynamic system, where the set of active nodes changes over time (this is called *churn*), the local views must continuously evolve to reflect these changes, adding new active nodes and removing ones that are no longer active. By using small local views, the maintenance overhead is kept small. In the absence of malicious behavior, small local views can be effectively maintained with gossip-based membership protocols [1, 13, 14, 19, 31], which were proven to have a low probability for partitions, including under churn [1].

Nevertheless, adversarial attacks present a major challenge for small local views. Previous Byzantine-tolerant gossip protocols either considered static settings where the full membership is known to all [11, 23, 29], or maintained (almost) full local views [4, 20], where faulty nodes cannot push correct ones out of the view. In contrast, small local views are susceptible to poisoning with entries (node ids) originating from faulty nodes; this is because the system is dynamic, and therefore nodes inherently must accept new ids and store them in place of old ones in their local views. It is even more challenging to provide *independent uniform samples* in such a setting. Even without Byzantine failures, gossip-based membership only ensures that eventually the *average* representation of nodes in local views is uniform [1, 14, 19], and not that *every node* obtains an independent uniform random sample. Faulty nodes may attempt to skew the system-wide distribution, as well as the individual local view of a given node.

In this paper, we address these challenges. We present Brahms (Section 3), a gossip-based membership service that stores a sublinear number of ids (e.g., $\Theta(\sqrt[3]{n})$ in a system of size n) at each node, and provides *each node* with membership samples that converge to uniform ones over time. The main ideas behind Brahms are (1) to use gossip-based membership with some extra defenses to make it viable (in the sense that local views are not solely composed of faulty ids) in an adversarial setting; (2) to recognize that such a solution is bound to produce biased views due to attacks (we precisely quantify the extent of this bias mathematically); and (3) to correct this bias at each node.

To achieve the latter, we introduce *Sampler*, a component that obtains uniform samples out of a data stream in which elements recur with an unknown bias, using min-wise independent permutations [8]. We prove (Section 5) that Sampler obtains independent uniform samples from the biased stream of gossiped node ids. By using such *history samples* of the gossiped ids to update part of the local view, Brahms achieves *self-healing* from partitions that may occur with gossip-based membership. In particular, nodes that have been active for sufficiently long (we quantify how long) cannot be isolated from the rest of the system. The use of history samples is an example of *amplification*, whereby even a small healthy sample of the past can boost the resilience of a constantly evolving view. We note that only a small portion of the view is updated with

[1]Department of Electrical Engineering, The Technion – Israel Institute of Technology. Email: {ebortnik@techunix, gmax@techunix, idish@ee, shralex@techunix}.technion.ac.il.
[2]Department of Computer Science, The Technion – Israel Institute of Technology. Email: gabik@cs.technion.ac.il.

PODC'08, August 18–21, 2008, Toronto, Ontario, Canada.

history samples, e.g., 10%. Therefore, the protocol can still deal effectively with churn.

One of the important contributions of this paper is our mathematical analysis (Section 6), which provides insights to the extent of damage that an attacker can cause and the effectiveness of various mechanisms for dealing with them. Extensive simulations of Brahms with up to 4000 nodes validate the few simplifying assumptions made in the analysis. We consider two possible goals for an attacker. First, we study attacks that attempt to maximize the representation of faulty ids in local views at any given time. We show that as long as faulty nodes comprise less than $\frac{1}{3}$ of the system, even without using history samples, the portion of faulty ids in local views is bounded by a constant smaller than one. (Recall that the over-representation of faulty ids is later fixed by Sampler; the upper bound on faulty ids in local views ensures Sampler has good ids to work with). If the adversary gains control of additional nodes after uniform samples have already been obtained, then Brahms can resist *any* ratio of faulty nodes.

Second, we consider an attacker that aims to partition the network. The easiest way to do so is by isolating one node from the rest. Clearly, once a node has obtained uniform samples of correct nodes, it can no longer be isolated. We therefore study an attack launched on a new node that joins the system when its samples are still empty, and when it does not yet appear in views or samples of other nodes. We further assume that such a *targeted* attack on the new node occurs in tandem with an attack on the entire system, as described above. The key to proving that Brahms prevents, w.h.p., an attacked node's isolation is in comparing how long it takes for two competing processes to complete: on the one hand, we provide a lower bound on the expected time to poison the entire view of the attacked node, assuming there are no history samples at all. On the other hand, we provide an upper bound on how fast history samples are expected to converge, under the same attack. Whenever the former exceeds the latter, the attacked node is expected to become immune to isolation before it is isolated. We prove that with appropriate parameter settings, this is indeed the case.

Finally, we simulate the complete system (Section 7), and measure Brahms's resilience to the combination of both attacks. Our results show that, indeed, Brahms prevents the isolation of attacked nodes, its views never partition, and the membership samples converge to perfectly random ones over time.

Related Work. We are not familiar with prior work dealing with random node sampling in a Byzantine setting. Previous Byzantine-tolerant membership services maintained full local views [20, 4] rather than partial samples. Previous work on gossip-based partial views [1, 13, 14, 19, 31], and on near-uniform node sampling using random walks [15, 22, 26, 5] or DHT overlays [21] was limited to benign settings.

One application of Brahms is Byzantine-tolerant overlay construction. Brahms's sampling allows each node to connect with some random correct nodes, thus constructing an overlay in which the sub-graph of correct nodes is connected. Several recent works, e.g., [30, 9, 3], have focused explicitly on securing overlays, mostly structured ones, attempting to ensure that all correct nodes may communicate with each other using the overlay, i.e., to prevent the *eclipse attack* [30], where routing tables of correct nodes are gradually poisoned with links to adversarial nodes. These works have a different focus than ours, since their goal is to construct (structured) overlay networks, whereas we present a general sampling technique, one application of which is building adversary-resilient unstructured overlays.

2. MODEL, GOAL, AND CHALLENGES

Model. We consider a system of nodes, each identified by a unique id. The way of choosing node ids is non-constrained, e.g., we do not assume a certification authority [20]. The system is subject to churn, i.e., nodes can join and leave it dynamically. The nodes can communicate through a fully connected network with reliable links. A node can determine the source of every message, and cannot intercept messages addressed to other nodes (the standard "unauthenticated" Byzantine model [2]). For simplicity of the further analysis, we assume a synchronous model with a discrete global clock, zero processing times, and message latencies of a single time unit.

Design goal. We pursue maintaining at each node a small set of node ids, which converges to a stable uniform membership *sample* after the churn ceases.

Gossip-based protocols (e.g., [1]) are a well-known mechanism for membership information dissemination in the presence of churn. These protocols maintain at each node a small subset of active node ids, called *view*, which facilitates communication between nodes. The gossiping nodes propagate information through two main primitives, *push* – unsolicited sending the node's id to some other node, and *pull* – request-reply retrieval of the other node's view. Part of the received information is used to update the views. This way, the knowledge about newborn nodes spreads through the system, and the information about defunct nodes disappears.

Computing a uniform sample at all nodes requires the overlay induced by the union of their views to remain perpetually connected (otherwise, two nodes in different connected components have zero probability to learn about each other). Allavena et al. [1] have demonstrated a benign gossip protocol for which the expected time until a network partition is exponential in the squares of the view size and the partition size, under reasonable assumptions. Extensive empirical studies [13, 18] validated that gossip maintains connectivity in practice.

However, traditional gossip is not resilient to malicious use of communication primitives. For example, an adversary can choose to over-represent the faulty ids in samples of some correct nodes. Note that our model rules out massive Sybil attacks, which allow the faulty nodes to use multiple ids [12]. We illustrate two kinds of behavior, both leading to rapid poisoning of views at all correct nodes.

Push flooding. The adversary can flood the correct nodes with pushes of faulty ids. A solely push-based gossip is clearly vulnerable to this attack (Figure 2(a)). Moreover, push flooding cannot be resisted unless the sending rate of Byzantine nodes is constrained. Hence, we assume a mechanism that makes it costly for nodes to send designated (push) messages, which we call *limited* messages. This mechanism can be implemented in different ways, e.g., computational challenges like Merkle's puzzles [28], virtual currency, etc. Note that employing limited messages is necessary but not sufficient: while they prevent the adversary from flooding all correct nodes in parallel, the latter can still target them one by one.

Skewed pull responses. The faulty nodes can return only faulty ids in response to pull requests. Since pulls from correct nodes return faulty ids as well as correct ids, this behavior leads to exponential decay in the representation of correct nodes. A purely pull-based dissemination fails to handle this attack (Figure 2(b)).

The described scenarios demonstrate that an adversary can exploit traditional gossip to bias the distribution of ids in the views of correct nodes. In the long run, it can disintegrate the entire overlay, thus precluding uniform sampling completely. Brahms adopts a two-layer approach to this problem. As a first step, we guaran-

Figure 1: Malicious attacks on traditional gossip protocols. (a) Push attack. (b) Pull attack.

tee, w.h.p., that the attacker cannot isolate correct nodes, that is, the maximum damage to their views is bounded. As a second step, we correct the incurred bias through local uniform sampling.

3. BRAHMS

Brahms has two components. The local *sampling* component maintains a *sample list* $\mathcal{S}$ – a tuple of uniform samples from the set of ids that traverse the node (Section 3.1). The *gossip* component is a distributed protocol that spreads locally known ids across the network (Section 3.2), and maintains a dynamic *view* $\mathcal{V}$. Each node has some initial $\mathcal{V}$ (e.g., received from some bootstrap server or peer node). $\mathcal{V}$ and $\mathcal{S}$ may contain duplicates, and some entries in $\mathcal{S}$ may be non-defined (denoted $\perp$).

3.1 Sampling

Sampler is a building block for uniform sampling of unique elements from a data stream. The input stream may be biased, that is, some values may appear in it more than others. Sampler accepts one element at a time as input, produces one output, and stores a single element at any time. The output is a uniform random choice of one of the unique inputs witnessed thus far.

Sampler uses *min-wise independent* permutations [8]. A family of permutations $\mathcal{H}$ over a range $[1 \ldots |U|]$ is min-wise independent if for any set $X \subset [1 \ldots |U|]$ and any $x \in X$, if h is chosen at random from $\mathcal{H}$, then $\Pr(\min\{h(X)\} = h(x)) = \frac{1}{|X|}$. That is, all the elements of any fixed set X have an equal chance to have the minimum image under h. Pseudo-random functions (e.g., [16]) are considered an excellent practical approximation of min-wise independent permutations, provided that $|U|$ is large, e.g., 2^{160}.

Sampler (Figure 2(a)) selects a random min-wise independent function h upon initialization, and applies it to all input values (`next()` function). The input with the smallest image value encountered thus far becomes the output returned by the `sample()` function. The property of uniform sampling from the set of unique observed ids follows directly from the definition of a min-wise independent permutation family.

Brahms maintains a tuple of ℓ_2 sampled elements in a vector of ℓ_2 Sampler blocks (Figure 2(b)), which select hashes independently. The same id stream is input to all Samplers. Sampled ids are periodically probed (e.g., using pings), and a Sampler that holds an inactive node is invalidated (re-initialized).

3.2 Gossip

Brahms's view is maintained by a gossip protocol (Figure 3). We denote list concatenation by $\circ$. By slight abuse of notation, we denote both the vector of samplers and their outputs (the sample list) by $\mathcal{S}$. Brahms executes in (unsynchronized) rounds. It uses two means for propagation: (1) *push* – sending the node's id to some other node, and (2) *pull* – retrieving the view from another node. These operations serve two different purposes: pushes are required to reinforce knowledge about nodes that are under-represented in other nodes' views (e.g., newborn nodes), whereas pulls are needed to spread existing knowledge within the network [1]. A combination of pulls and pushes is required because the representation of ids propagated solely by pulls decays over time, whereas the representation of push-propagated ids increases.

Brahms uses parameters $\alpha > 0$, $\beta > 0$ and $\gamma > 0$ that satisfy $\alpha + \beta + \gamma = 1$. In a single round, a correct node issues $\alpha\ell_1$ push requests and $\beta\ell_1$ pull requests to destinations randomly selected from its view (possibly with repetitions). At the end of each round, $\mathcal{V}$ and $\mathcal{S}$ are updated with fresh ids. While all received ids are streamed to $\mathcal{S}$ (Figure 2, Line 37), re-computing $\mathcal{V}$ requires extra care, to protect against poisoning of the views with faulty ids. Brahms offers a set of techniques sto mitigate this problem.

Limited pushes. Since pushes arrive unsolicited, an adversary with an unlimited capacity could swamp the system with push requests. Then, correct ids would be propagated mainly through pulls, and their representation would decay exponentially [1]. The protocol employs limited push messages, hence the fraction of faulty pushes is constrained.

Attack detection and blocking. While using limited pushes prevents a simultaneous attack on all correct nodes, it provides no solace against an adversary that floods a specific node. Brahms protects against this *targeted attack* by blocking the update of $\mathcal{V}$. Namely, if more than the expected $\alpha\ell_1$ pushes are received, it does not update $\mathcal{V}$. Although this policy slows down progress, its expected impact in the absence of attacks is bounded (nodes recompute $\mathcal{V}$ in most rounds). Thanks to limited pushes, some nodes make progress even under an attack (Line 35).

Controlling the contribution of pushes vs pulls. As most correct nodes do not suffer from targeted attacks (due to limited pushes), their views are threatened by pulls from neighbors more than by adversarial pushes. This is because whereas all pushes from correct nodes are correct, a pull from a random correct node may contribute some faulty ids. Hence, the contribution of pushes and pulls to $\mathcal{V}$ must be balanced: pushes must be constrained to protect the targeted nodes, while pulls must be constrained to protect the rest. Brahms updates $\mathcal{V}$ with randomly chosen $\alpha\ell_1$ pushed ids and $\beta\ell_1$ pulled ids (Line 36).

History samples. The attack detection and blocking technique can slowdown a targeted attack, but not prevent it completely. Note that if the adversary succeeds to increase its representation in a victim's view through targeted pushes, it subsequently causes this victim to pull more data from faulty nodes. As the attacked node's view deteriorates, it sends fewer pushes to correct nodes, causing its system-wide representation to decrease. It then receives fewer correct pushes, opening the door for more faulty pushes[1]. Brahms overcomes such attacks using a self-healing mechanism, whereby a portion (γ) of $\mathcal{V}$ reflects the *history*, i.e., previously observed ids (Line 36). A direct use of history does not help since the latter may also be biased. Therefore, we use a feedback from $\mathcal{S}$ to obtain unbiased history samples. Once some correct id becomes the attacked node's permanent sample (or the node's id becomes a permanent sample of some other correct node), the threat of isolation is eliminated. Figure 4 illustrates the view re-computation procedure.

Brahms's parameters entail a tradeoff between performance in a benign setting and resilience against Byzantine attacks. For example, γ must not be too large since the algorithm needs to deal with churn; on the other hand, it must not be too small to make the feedback effective. We show (Section 7) that $\gamma = 0.1$ is enough for protecting $\mathcal{V}$ from partitions. The choice of ℓ_1 and ℓ_2 is crucial for guaranteeing that a targeted attack can be contained until the

[1]This avalanche process can be started, e.g., by opportunistically sending the target a slightly higher number of pushes than expected. Since correct pushes are random, a round in which sufficiently few correct pushes arrive, such that Brahms does not detect an attack, happens soon w.h.p.

```
1: function Sampler.init()
2:     h ← randomPRF(); q ← ⊥
3: function Sampler.next(elem)
4:     if q = ⊥ ∨ h(elem) < h(q) then
5:         q ← elem
6: function Sampler.sample()
7:     return q
```

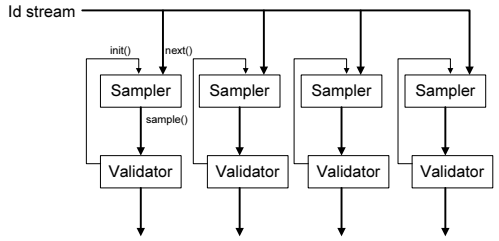


Figure 2: Uniform sampling from an id stream in Brahms. (a) Sampler's pseudo-code. (b) Sampling and validation of ℓ_2 ids.

```
1: V : tuple[ℓ1] of Id
2: S : tuple[ℓ2] of Sampler
3: Initialization(V0):
4:     V ← V0
5:     for all 1 ≤ i ≤ ℓ2 do
6:         S[i].init()
7:     updateSample(V0)
8: {Stale sample invalidation}
9: periodically do
10:    for all 1 ≤ i ≤ ℓ2 do
11:        if probe(S[i].sample()) fails then
12:            S[i].init()
13: {Auxiliary functions}
14: function updateSample(V)
15:    for all id ∈ V, 1 ≤ i ≤ ℓ2 do
16:        S[i].next(id)
17: function rand(V, n)
18:    return n random choices from V
19: {Gossip}
20: while true do
21:    V_push ← V_pull ← ∅
22:    for all 1 ≤ i ≤ αℓ1 do
23:        {Limited push}
24:        send_lim ⟨"push_request"⟩ to rand(V, 1)
25:    for all 1 ≤ i ≤ βℓ1 do
26:        send ⟨"pull_request"⟩ to rand(V, 1)
27:    wait(1)
28:    for all received ⟨"push_request"⟩ from id do
29:        V_push ← V_push ∘ {id}
30:    for all received ⟨"pull_request"⟩ from id do
31:        send ⟨"pull_reply", V⟩ to id
32:    for all received ⟨"pull_reply", V'⟩ from id do
33:        if I sent the request, and this is the first reply then
34:            V_pull ← V_pull ∘ V'
35:    if (|V_push| ≤ αℓ1 ∧ V_push ≠ ∅ ∧ V_pull ≠ ∅) then
36:        V ← rand(V_push, αℓ1) ∘ rand(V_pull, βℓ1) ∘ rand(S, γℓ1)
37:    updateSample(V_push ∘ V_pull)
```

Figure 3: The pseudo-code of Brahms.

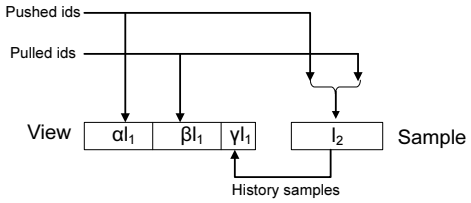


Figure 4: View re-computation in Brahms.

attacked node's sample stabilizes. For example, $\ell_1, \ell_2 = \Theta(\sqrt[3]{n})$ suffice to protect even nodes that are attacked immediately upon joining the system (Section 6.2).

4. DEFINITIONS AND ATTACK MODELS

We study the asymptotical properties of a system of n nodes with unique ids, after a point T_0 at which the churn ceases. The subset of correct nodes is denoted $\mathcal{C}$. The faulty nodes comprise less than some fraction $f < 1$ of n. We assume that the system-wide fraction of limited pushes that all faulty nodes can jointly send in a single time unit is at most p, for some $p < 1$.

We denote the view and the sample list at node u at time t by $\mathcal{V}_u(t)$ and $\mathcal{S}_u(t)$, respectively. We define the *overlay graph* $\mathcal{N}(t)$, induced by the union of $\mathcal{V}$ and $\mathcal{S}$ at all correct nodes, which captures their knowledge about each other at time t:

$$\mathcal{N}(t) \triangleq \{\mathcal{C}, \bigcup_{u \in \mathcal{C}} \{(u,v) | v \in (\mathcal{V}_u(t) \bigcup \mathcal{S}_u(t)) \cap \mathcal{C}\}\}.$$

We also define $\mathcal{V}(t)$, a subgraph of $\mathcal{N}(t)$ induced by $\mathcal{V}$ of correct nodes (edges induced by $\mathcal{S}$ are omitted):

$$\mathcal{V}(t) \triangleq \{\mathcal{C}, \bigcup_{u \in \mathcal{C}} \{(u,v) | v \in \mathcal{V}_u(t) \cap \mathcal{C}\}\}.$$

For a node u, the number of its incoming edges in a graph is called its *in-degree*, and the number of outgoing edges is called its *out-degree*. For example the in-degree of node u in $\mathcal{V}(t)$ is the number of instances of u in views of correct nodes, and its out-degree is the number of correct ids in its view. The *degree* of u is the sum of its in-degree and out-degree.

Brahms's resilience depends on the distribution of in-degrees and out-degrees in $\mathcal{V}(t)$. We assume a necessary condition for initial connectivity, namely, that the view of every joining correct node contains some correct ids (the ratio of faulty ids in the view is not necessarily bounded by f). We further assume that before the attack starts, the in-degrees and out-degrees of all correct nodes are (roughly) equal. This property is a close approximation of reality, since a benign gossip process preserves it [1].

We assume the worst-case behavior of Byzantine nodes, i.e., pushing faulty ids to correct nodes and always return faulty ids to pulls. Faulty nodes always respond to probe requests, to avoid invalidation. We consider two representative Byzantine attacks. The first attack, called *balanced*, maximizes the system-wide representation of faulty ids by distributing the faulty pushes evenly among the correct nodes. The second attack, called *targeted*, focuses an increased portion of faulty pushes on a small subset of correct nodes, in order to isolate them from the overlay. Section 6 analyzes the dynamics of both attacks, and demonstrates that Brahms prevents the overlay's partitioning, with the right choice of parameters.

5. ANALYSIS - SAMPLING

In this section we analyze the properties of $\mathcal{S}_u$ of a correct node u. Let $s = \mathcal{S}_u[i]$ be a sampler block for some correct u and some i. Recall that s employs a min-wise independent permutation $s.h$, chosen independently at random. Let let $s(t)$ be the output of s at time t. We define the *perfect* id corresponding to s, s^*, to be the id with the minimal value of $s.h$ among all ids (we neglect collisions for the sake of the definition). Note that s^* can be either a correct or a faulty id. In Section 5.1 we show that the subset of correct ids in $\mathcal{S}_u$ eventually converges to a uniform random sample from $\mathcal{C}$. In Section 5.2 we analyze how fast a node obtains at least one correct perfect sample, as needed for self-healing. Section 5.3 discusses scalability, namely, how to choose view sizes that ensure a constant convergence time, independent of system size.

5.1 Eventual Convergence to Uniform Sample

Consider sampler s of node u. If s^* is correct, s samples correct ids uniformly at random. Obviously, for s to be able to sample some correct node v, the id of v has to reach u. To guarantee such a reachability between all the correct nodes, we require the overlay graph $\mathcal{N}(t)$ to remain *weakly connected* after T_0. That is, the undirected graph, obtained from $\mathcal{N}(t)$ by replacing all of its directed edges with undirected ones, is connected for all $t \geq t_0$. If s^* is faulty, it may remain "silent", thus preventing its id from being known to correct nodes. The following theorem shows that each correct id has roughly the same probability to be sampled by s.

THEOREM 5.1. *If $\mathcal{N}(t)$ remains weakly connected for each $t \geq T_1$, for some $T_1 \geq T_0$, then, for all $v \in \mathcal{C}$, and $\varepsilon > 0$, there exists $T_\varepsilon \geq T_1$ such that for all $t \geq T_\varepsilon$*

$$\frac{1}{n} - \varepsilon \leq \Pr(s(t) = v) \leq \frac{1}{(1-f)n} + \varepsilon.$$

Proof idea. The key to the theorem is to show that whenever $\mathcal{N}(t)$ remains weakly connected, the id of each correct node eventually reaches every other correct node with probability 1. This is because the id has a non-zero probability to traverse a path to every correct node in the system. Thus, each sampler will eventually settle on its perfect id, provided that its perfect id is correct. Therefore, $\Pr(s(t) = s^* | s^* \in \mathcal{C}) \rightarrow_{t\rightarrow\infty} 1$. Since the probability for s^* to be faulty is at most f, $\Pr(s(t) = s^*)$ approaches the range $[1-f, 1]$. The theorem follows since $\forall v \in \mathcal{C}, \Pr(s(t) = v | s^* \in \mathcal{C}) = \frac{1}{|\mathcal{C}|} \leq \frac{1}{(1-f)n}$, and since we assume that when s^* is faulty, $0 \leq \Pr(s(t) = v | s^* \notin \mathcal{C}) \leq \frac{1}{(1-f)n}$.

The next lemma discusses the convergence rate of samples.

LEMMA 5.2. *If no invalidations happen, for each correct node u, the expected fraction of samplers that output their perfect id grows linearly with the fraction of unique ids observed by u.*

PROOF. Let $D(t)$ be the set of ids observed by u until time t. Then, for each sampler s, $\Pr(s^* \in D(t)) = \frac{|D(t)|}{n}$. Since for each s such that $s^* \in D(t)$, $s(t') = s^*$ for $t' \geq t$, the lemma follows. □

5.2 Convergence to First Perfect Sample

We show a lower bound on the probability that $\mathcal{S}_u$ contains *at least one* perfect id of an active correct node, as a function of the ids it observes and system parameters. This provides an upper bound on the time it takes $\mathcal{S}_u$ to ensure self-healing and prevent u's isolation. We assume that u joins the system at time T_0, with an empty sample. Let $\Lambda(t)$ be the number of correct ids observed by u from time T_0 to t. Our analysis depends on the number of unique ids observed by u, rather than directly on Λ. Obviously, one can expect the observed stream to include many repetitions, as it is unrealistic to expect our gossip protocol to produce independent uniform random samples (cf. [19]). Indeed, achieving this property is the goal of sampler. In order to capture the bias in Λ, we define a *stream deficiency factor*, $0 \leq \rho \leq 1$, so that a stream of length $\Lambda(t)$ produced by our gossip mechanism is roughly equivalent, for the purposes of sampler, to a stream of length $\rho\Lambda(t)$ in which correct ids are independent and distributed uniformly at random. This is akin to the clustering coefficient of gossip-based overlays [19]. We empirically measured ρ to be about 0.4 with our gossip protocol (see Section 6.2).

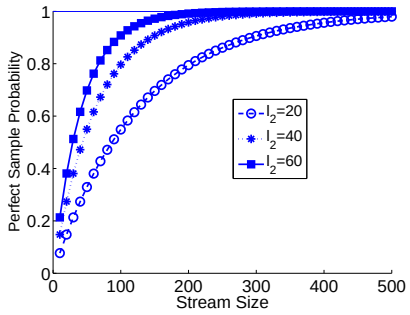


Figure 5: Growth of the probability to observe at least one correct perfect sample (Perfect Sample Probability - PSP) with the stream size, for 1000 nodes, $f = 0.2$, and $\rho = 0.4$.

We define the *perfect sample probability* $PSP_u(t)$ as the probability that $\mathcal{S}_u(t)$ contains at least one correct perfect id. The convergence rate of PSP is captured by the following:

LEMMA 5.3. *Let u be a random correct node. Then, for $t > T_0$, $PSP_u(t) \geq 1 - ((1-f)e^{-\frac{\rho\Lambda(t)}{|\mathcal{C}|}} + f)^{\ell_2}$.*

Proof idea. A sampler outputs a correct perfect id if (1) its perfect id is correct, and (2) this id is observed by the sampler in the stream. PSP is the probability that at least one of ℓ_2 samplers outputs a correct perfect id.

Figure 5 illustrates the dependence of PSP on the stream size $\Lambda(t)$ and on ℓ_2. When the sample size is $40 = 4\sqrt[3]{n}$, and the portion of unique ids in the stream is $\rho = 0.4$, a perfect sample is obtained, with probability close to 1, after 300 ids traverse the node.

5.3 Scalability

From Lemma 5.3 we see that PSP depends on Λ and ℓ_2. To get a higher PSP, we can increase either one. While increasing Λ is achieved by increasing ℓ_1, and consequently the network traffic, increasing ℓ_2 has only a memory cost. We now study the asymptotic behavior of $PSP_u(t)$ as the number of the nodes, n, increases. When a node has ℓ_2 samplers, $\Omega(\ell_2)$ of them have correct s^* w.h.p. Therefore, w.h.p., $PSP_u(t) \geq \Omega(1 - (e^{-\frac{\rho\Lambda(t)}{n}})^{\ell_2}) = \Omega(1 - e^{-\frac{\rho\Lambda(t)\ell_2}{n}})$. For a constant t, $\Lambda(t) = \Omega(\ell_1^2)$ since there are $\Omega(\ell_1)$ pulls, obtaining $\Omega(\ell_1)$ ids each. Thus, $PSP_u(t) \geq \Omega(1 - e^{-\frac{\ell_1^2 \cdot \ell_2}{n}})$. For scalability, it is important that for a given t, $PSP_u(t)$ will be bounded by a constant independent of the system size. This condition is satisfied when $\ell_1^2 \cdot \ell_2 = \Omega(n)$, e.g., when $\ell_2 = \ell_1 = \Omega(\sqrt[3]{n})$, or $\ell_1 = \Omega(\sqrt[4]{n})$ and $\ell_2 = \Omega(\sqrt[2]{n})$. To reduce network traffic at the cost of a higher memory consumption, one can set $\ell_1 = \Omega(\log n)$ and $\ell_2 = \Omega(\frac{n}{\log^2 n})$.

Correct node u	Random correct node	Semantics
number/fraction	number/fraction	
$x_u(t)/\tilde{x}_u(t)$	$x(t)/\tilde{x}(t)$	faulty ids in the view
$y_u(t)/\tilde{y}_u(t)$		instances among the views of correct nodes
$g_u^{\text{push}}(t)/\tilde{g}_u^{\text{push}}(t)$	$g^{\text{push}}(t)/\tilde{g}^{\text{push}}(t)$	correct ids pushed to node
$r_u^{\text{push}}(t)/\tilde{r}_u^{\text{push}}(t)$	$r^{\text{push}}(t)/\tilde{r}^{\text{push}}(t)$	faulty ids pushed to node
$g_u^{\text{pull}}(t)/\tilde{g}_u^{\text{pull}}(t)$	$g^{\text{pull}}(t)/\tilde{g}^{\text{pull}}(t)$	correct ids pulled by node
$r_u^{\text{pull}}(t)/\tilde{r}_u^{\text{pull}}(t)$	$r^{\text{pull}}(t)/\tilde{r}^{\text{pull}}(t)$	faulty ids pulled by node

Table 1: Definition of common random variables.

6. ANALYSIS – OVERLAY CONNECTIVITY

We prove that Brahms, with appropriate parameter settings, maintains overlay connectivity despite malicious behavior. Our methodology is mathematical analysis, which, like previous studies [1], makes some simplifying assumptions. The theoretical results are validated through extensive simulations.

We first bound the damage that can be caused within a *single* round (a similar approach was taken, e.g., in [23]). In the full version of this paper [7], we prove that in any single round, a *balanced* attack, which spreads faulty pushes evenly among correct nodes, maximizes the expected system-wide fraction of faulty ids, $\tilde{x}(t)$, among all strategies. In Section 6.1, we prove that if this attack persists, the ratio of faulty ids in the system eventually stabilizes at a fixed point. We study the convergence process, and show that for certain parameter choices, this fixed point is strictly smaller than 1.

Alternatively, an adversary can try to partition the network (rather than increase its representation) by targeting a subset of nodes with more pushes than in a balanced attack. Without prior information about the overlay's topology, attacking a single node can be most damaging, since the sets of edges adjacent to single nodes are likely to be the sparsest cuts in the overlay. Section 6.2 shows that had Brahms not used history samples, correct nodes could have been isolated in this manner. However, Brahms sustains such *targeted* attacks, even if they start immediately upon a node's join, when it is not represented in other views and has no history. The key property is that Brahms's gossip prevents isolation long enough for history samples to become effective.

Notation. We study time-varying random variables, listed in Table 1. A local variable at a specific correct node u is subscripted by $_u$. When used without subscript, a variable corresponds to a random correct node. Correct (resp., faulty) ids propagated through pushes and pulls are denoted g (green) (resp., r (red)).

Simulation setup. We validate our assumptions using simulations with $n = 1000$ nodes or more. Each data point is averaged over 100 runs. For simplicity, we assume $p = f$. A different subset of faulty nodes push their ids to a given correct node in each round, using a round-robin schedule.

6.1 Balanced Attack

In the analysis of a balanced attack we ignore blocking since its only effect is to slow the convergence rate. Simulations show that this assumption has little effect on the results. Since a balanced attack does not distinguish between correct nodes, we assume that it preserves the in-degrees and out-degrees of all correct nodes equal over time:

ASSUMPTION 6.1. *For all $u \in \mathcal{C}$ and all $t \geq T_0$: $x_u(t) = x(t)$, and $y_u(t) = \ell_1 - x_u(t)$.*

We show the existence of a parameter-dependent fixed point of $\tilde{x}(t)$ and the system's convergence to it. Since the focus is on asymptotic behavior, we assume $t \gg T_0$.

LEMMA 6.1. *For $t \gg T_0$, if $p \neq 0$ or $\tilde{x}(t) \neq 1$, the expected system-wide fraction of faulty ids evolves as*

$$\begin{aligned}\mathrm{E}(\tilde{x}(t+1)) \quad &= \alpha\tfrac{p}{p+(1-p)(1-\tilde{x}(t))} \\ &+\beta(\tilde{x}(t)+(1-\tilde{x}(t))\tilde{x}(t)) \\ &+\gamma f.\end{aligned}$$

PROOF. Consider the re-computation of $\mathcal{V}$ at a correct node u at time t. The weights of pushes, pulls, and history samples in the recomputed view are α, β and γ, respectively. Since the random selection process preserves the distribution of faulty ids in each data source, the probability of a push- (resp., pull)-originated entry being faulty is equal to the probability of receiving a faulty push (resp., pulling a faulty id).

Figure 6(a) illustrates the analysis of $\tilde{r}^{\text{push}}(t)$. Each correct node wastes an expected fraction $\tilde{x}(t)$ of its pushes because they are sent to faulty nodes. The rest are sent with an equal probability over each outgoing edge in $\mathcal{V}(t)$. Since out-degrees and in-degrees are equal among all correct nodes, each correct node u receives the same expected number of correct pushes: $\mathrm{E}(g_u^{\text{push}}(t)) = (1-\tilde{x}(t))\alpha\ell_1$. The variable $g_u^{\text{push}}(t)$ is binomially distributed, with the number of trials equal to the total number of pushes among all nodes with an outgoing edge to u. Since this number is large, the number of received correct pushes is approximately equal among all correct nodes, i.e., $g_u^{\text{push}}(t) \approx (1-\tilde{x}(t))\alpha\ell_1$, for all u.

The total number of correct pushes is $\alpha\ell_1|\mathcal{C}|$, which is $1-p$ out of all pushes, hence the total number of faulty pushes is $\frac{p\alpha\ell_1}{1-p}|\mathcal{C}|$. Therefore, u receives exactly $r_u^{\text{push}}(t) = \frac{p}{1-p}\alpha\ell_1$ faulty pushes, i.e., their fraction among all received pushes is:

$$\tilde{r}_u^{\text{push}}(t) = \frac{\frac{p}{1-p}\alpha\ell_1}{\frac{p}{1-p}\alpha\ell_1 + (1-\tilde{x}(t))\alpha\ell_1} = \frac{p}{p+(1-p)(1-\tilde{x}(t))}.$$

Hence, the expected ratio of push-originated faulty ids in $\mathcal{V}_u$ is $\alpha\frac{p}{p+(1-p)(1-\tilde{x}(t))}$.

Figure 6(b) depicts the evolution of pull-originated faulty ids. Since all correct nodes have an equal out-degree, a correct node is pulled with probability $1-\tilde{x}(t)$, while a faulty node is pulled with probability $\tilde{x}(t)$. A pulled id is faulty with probability $\tilde{x}(t)$ if it comes from a correct node, and otherwise, it is always faulty. Hence, the expected fraction of pull-originated faulty ids is $\beta(\tilde{x}(t)+(1-\tilde{x}(t))\tilde{x}(t))$.

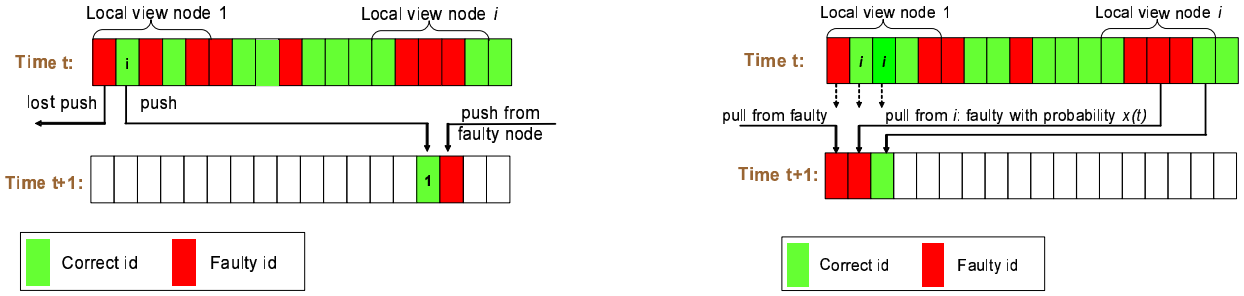
Finally, since $t \gg T_0$, the history sample is perfect (the ratio of faulty ids in it is f). Hence, its expected contribution is γf, and the claim follows. □

We now show that the system converges to a stable state. A value $\hat{x}$ is called a *fixed point* of $\tilde{x}(t)$ if $\mathrm{E}(\tilde{x}(t+1)) = \tilde{x}(t) = \hat{x}$. Substituting this into the equation from Lemma 6.1, we get:

LEMMA 6.2. *For $\alpha, \beta, \gamma, p, f \in [0,1]$, every real root $0 \leq \hat{x} \leq 1$ of the following cubic equation is a fixed point of $\tilde{x}(t)$, except for the root $x = 1$ for $p = 0$:*

$$\begin{aligned}&\beta(1-p)\tilde{x}^3 &&+ \\ &(2\beta p - 3\beta - p + 1)\tilde{x}^2 &&+ \\ &(\gamma f p - \gamma f + 2\beta - 1)\tilde{x} &&+ \\ &(\alpha p + \gamma f) &&= \quad 0.\end{aligned}$$

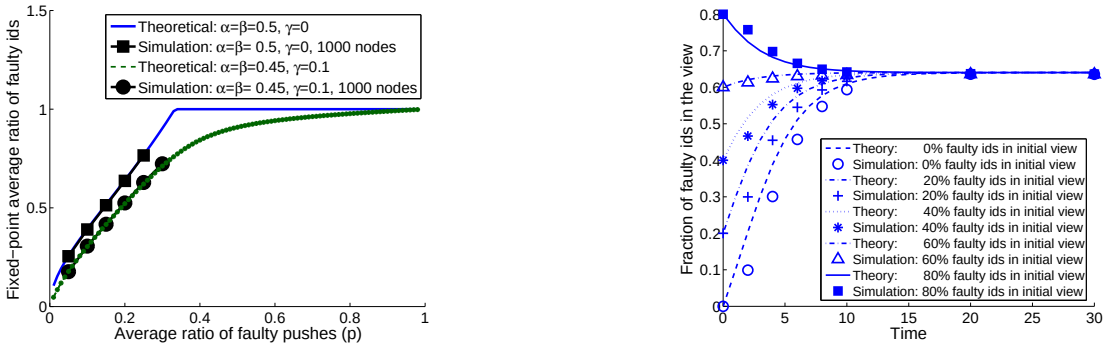
If $\gamma = 0$ (no history samples), $\hat{x} = 1$ is always a root. We call it a *trivial* fixed point. This is easily explainable, since if the views of all the correct nodes are totally poisoned, then neither pulls nor pushes help. In the full version of this paper [7], we show that if



(a) Impact of push

(b) Impact of pull

Figure 6: Fixed point analysis illustration.



(a) Fixed point $\hat{x}$ as a function of p, for $\gamma = 0$ and $\gamma > 0$

(b) Convergence to $\hat{x}$: $n = 1000$, $p = 0.2$, $\alpha = \beta = 0.5$ and $\gamma = 0$.

Figure 7: System-wide fraction of faulty ids in local views, under a balanced attack: (a) Fixed points (b) Convergence process.

$\gamma = 0$, there can exist at most one nontrivial fixed point $0 \leq \hat{x} < 1$. For example, if $\alpha = \beta = \frac{1}{2}$ and $\gamma = 0$, then $\hat{x} = \frac{p+\sqrt{4p-3p^2}}{2(1-p)}$, for $0 \leq p \leq \frac{1}{3}$. In contrast, if the fraction of faulty pushes exceeds $\frac{1}{3}$, the only fixed point is 1, causing isolation of all correct nodes.

If $\gamma > 0$, there exists a single nontrivial fixed point for all p. This highlights the importance of history samples. Figure 7(a) depicts the analysis results, perfectly matched by simulations.

We conclude the analysis by proving convergence to a nontrivial fixed point (the complete proof appears in [7]).

LEMMA 6.3. *If there exists a fixed point $\hat{x} < 1$ of $\tilde{x}(t)$, and $\tilde{x}(T_0) < 1$, then $\tilde{x}(t)$ converges to $\hat{x}$.*

Proof idea. We show that for all t, the sequence of $\tilde{x}(t)$ is trapped between $\hat{x}$ and another sequence, $\phi(t)$, that converges to $\hat{x}$. Hillam's theorem [17] is then used to prove sequence convergence.

Since the balanced attack does not distinguish between correct nodes, the same result holds for $\tilde{x}_u(t)$, for each correct node u. Figure 7(b) depicts the convergence to the nontrivial fixed point from various initial values of $\tilde{x}(t)$. The analytical and simulation results are similar. The latter's convergence is slightly slower because the analysis ignores blocking.

6.2 Targeted Attack

We study a targeted attack on a single correct node u, which starts upon u's join at T_0. We prove that u is not isolated from the overlay by showing a lower bound on the expected time to isolation, which exceeds an upper bound on the time to a perfect correct sample (a sufficient condition for non-isolation, Section 5).

Lower bound on expected isolation time. As we seek a lower bound, we make a number of worst-case assumptions (formally stated in [7]). First, we analyze a simplified protocol that does not employ history samples (i.e., $\gamma = 0$), so that $\mathcal{S}$ does not correct $\mathcal{V}$'s bias. Next, we assume an unrealistic adaptive adversary that observes the exact number of correct pushes to u, $g_u^{\text{push}}(t)$, and complements them with $\alpha\ell_1 - g_u^{\text{push}}(t)$ faulty pushes – the most that can be accepted without blocking. The adversary maximizes its global representation through a balanced attack on all correct nodes $v \neq u$, thus minimizing the fraction of correct ids that u pulls from correct nodes. Finally, we assume that u is not represented in the system initially, and it derives its initial view from a random set of correct nodes, where the ratio of faulty ids is at the fixed point (Section 6.1).

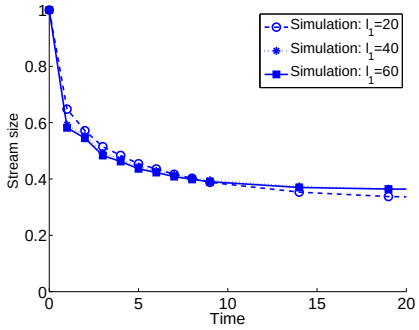
Clearly, the time to isolation in $\mathcal{V}(t)$ is a lower bound on that in $\mathcal{N}(t)$. We study the dynamics of the number of correct ids in u's out-degree in $\mathcal{V}(t)$, $\ell_1 - x_u(t)$, and u's in-degree, $y_u(t)$. We show in [7] that for any two specific values of $x_u(t)$ and $y_u(t)$, the expected out-degree and in-degree values at $t+1$ are

$$\begin{pmatrix} \ell_1 - \mathrm{E}(x_u(t+1)) \\ \mathrm{E}(y_u(t+1)) \end{pmatrix} = A_{2\times 2} \times \begin{pmatrix} \ell_1 - x_u(t) \\ y_u(t) \end{pmatrix},$$

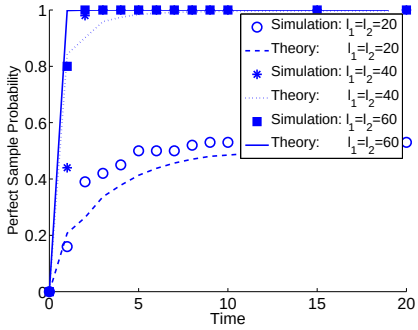
where

$$A_{2\times 2} = \begin{pmatrix} \beta(1-\hat{x}) & \alpha \\ \alpha\frac{1-p}{p+(1-p)(1-\hat{x})} & \beta(1-\hat{x}) \end{pmatrix}.$$

Note that the coefficient matrix does not depend on $x_u(t)$ and $y_u(t)$, and the sum of entries in each row is smaller than 1. This implies that once the in-degree and the out-degree are close, they both decay exponentially. (Initially, this does not hold because u is not represented, i.e., $y_u(T_0) = 0$.) Hence, the expected time to isolation is logarithmic with ℓ_1. Note that this process does not depend on the number of nodes, since blocking bounds the potential attacks on

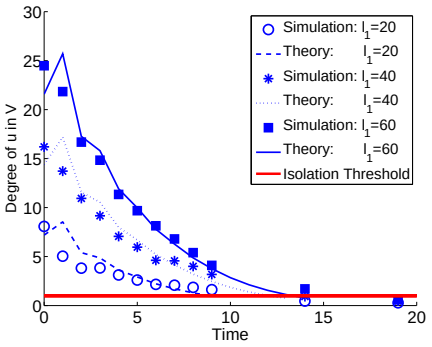


(a) Deficiency factor

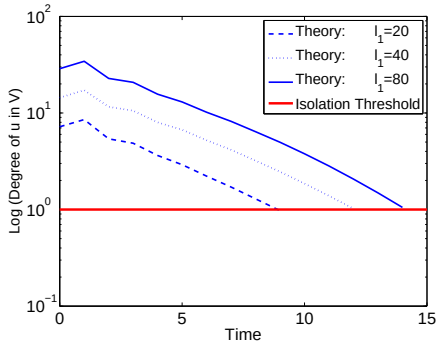


(b) Perfect sample probability

Figure 8: Dynamics within a targeted node ($n = 1000$, $p = 0.2$, $\alpha = \beta = 0.5$ and $\gamma = 0$): (a) Fraction of unique ids in the stream of correct ids, ρ. (b) Growth of Perfect Sample Probability (PSP) with time, $\rho = 0.4$. PSP becomes high quickly enough to prevent isolation.



(a) Normal scale (theory and simulation), $n = 1000$



(b) Logarithmic scale (theory only), independent of n

Figure 9: Targeted attack without history samples: node degree dynamics. $n = 1000$, $p = 0.2$, $\alpha = \beta = 0.5$, $\gamma = 0$. Without history samples a targeted attack isolates u in logarithmic time in ℓ_1.

u independently of the system-wide budget of faulty pushes. Had blocking not been employed, the top right coefficient would have been 0 instead of α, because the adversary would have completely hijacked the push-originated entries in $\mathcal{V}_u$. The decay factor would have been much larger, leading to almost immediate isolation.

Figure 9(a) depicts the dynamics of u's expected degree (the sum of u's in- and out-degrees) until it becomes smaller than 1. Simulation results closely follow our analysis. The temporary growth in u's degree at $t = 1$ occurs because u becomes represented in the system after the first round. For example, the average time to isolation for $\ell_1 = 2\sqrt[3]{n}$ is 10 rounds. Figure 9(b) depicts the same results in log-scale, emphasizing the exponential decay of u's degree and the logarithmic dependency between ℓ_1 and time to isolation.

Upper bound on expected time to perfect correct sample. For given values of the non-unique stream size $\Lambda(t)$ and the deficiency factor ρ (Section 5), Lemma 5.3 bounds $PSP_u(t)$. The expected number of correct ids observed by u till the end of round T is $\Lambda(t) = \sum_{t=T_0}^{T_0+T-1}(\mathrm{E}(g_u^{\mathrm{push}}(t)) + \mathrm{E}(g_u^{\mathrm{pull}}(t)))$; the expected values of $g_u^{\mathrm{push}}(t)$ and $g_u^{\mathrm{pull}}(t)$ are by-products of the analysis in [7], for $\gamma = 0$. Figure 8(a) depicts the deficiency factor ρ measured by our simulations, which behaves similarly for all values of ℓ_1: $\rho \geq 0.4$ for all t. Figure 8(b) depicts the progress of the upper bound of Lemma 5.3 with time, with $\Lambda(t)$ computed as explained above and $\rho = 0.4$. The corresponding simulation results show, for each time t, the fraction of runs in which at least one correct id in $\mathcal{S}_u$ is perfect. For $\ell_2 \geq 40$, the PSP becomes close to 1 in a few rounds, much faster than isolation happens (Figure 9(b)). For $\ell_1 = 20$, it stabilizes at 0.5. The growth stops because we run the protocol without history samples, thus u becomes isolated, and the id stream ceases. A higher PSP can be achieved by independently increasing ℓ_2, e.g., if ℓ_2 is 40, then the PSP grows to 0.8 (Figure 5). Note that perfect samples only provide an upper bound on self-healing time, as $\mathcal{S}_u$ contains imperfect correct ids, and u also becomes sampled by other correct nodes, w.h.p. These factors coupled with history samples ($\gamma > 0$) completely prevent u's isolation, as shown in Section 7.

7. PUTTING IT ALL TOGETHER

In previous sections we analyzed each of Brahms's mechanisms separately. We now simulate the entire system. Figure 10 depicts the degree of node u in $\mathcal{N}(t)$ under a targeted attack. Node u remains connected to the overlay, thanks to history samples ($\gamma = 0.1$). The actual degree of u in $\mathcal{N}(t)$ is higher than the lower bound shown in Section 6.2, due to the pessimistic assumptions made in the analysis (no history samples, no imperfect correct ids, etc.).

We now demonstrate the convergence of $\mathcal{S}$ in the correct nodes. We simulate systems with up to $n = 4000$ nodes; ℓ_1 and ℓ_2 are set to $2\sqrt[3]{n}$. To measure the quality of sample $\mathcal{S}$ under a balanced attack, we depict the fraction of ids in $\mathcal{S}$ that are indeed the perfect sample over time (Figure 11(a)). Note that this criterion is conservative, since missing a perfect sample does not automatically lead to a biased choice. More than 50% of perfect samples are achieved

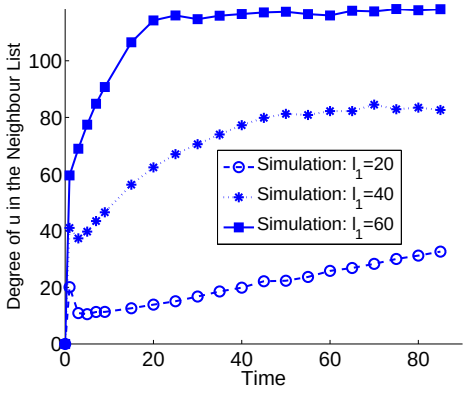
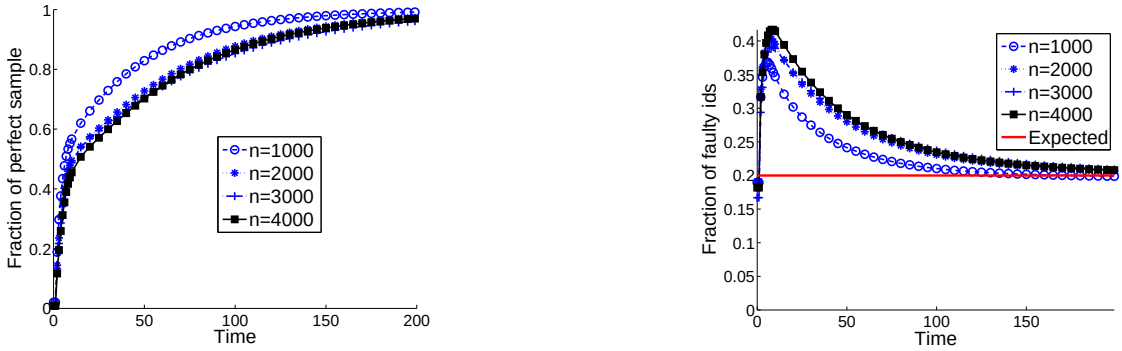


Figure 10: Targeted attack: degree dynamics of an attacked node in $\mathcal{N}(t)$, $n = 1000$, $p = 0.2$, $\alpha = \beta = 0.45$ and $\gamma = 0.1$.



(a) Perfect samples vs time

(b) Faulty ids vs. time

Figure 11: Balanced attack: fraction of perfect samples (a) and faulty nodes (b) in $\mathcal{S}$, for $f = 0.2$, $n = 1000, \ldots, 4000$, and $\ell_2 = 2\sqrt[3]{n}$.

within less than 15 rounds; for $\ell_2 = \ell_1 = 3\sqrt[3]{n}$, the convergence is twice as fast. Figure 11(b) depicts the evolution of the fraction of faulty ids in $\mathcal{S}$. Initially, this fraction equals f, and at first increases, up to approximately the fixed point's value. This is to be expected, since the first observed samples are distributed like the original (biased) data stream. Subsequently, as the node encounters more unique ids, the quality of $\mathcal{S}$ improves, and the fraction of faulty ids drops fast to f. The protocol exhibits almost perfect scalability, as the convergence rate is the same for $n \geq 2000$.

8. CONCLUSIONS

We presented Brahms, a Byzantine-resilient membership sampling algorithm. Brahms stores small views, and yet resists the failure of a linear portion of the nodes. It ensures that every node's sample converges to a uniform one, which was not achieved before by gossip-based membership even in benign settings. We presented extensive analysis and simulations explaining the impact of various attacks on the membership, as well as the effectiveness of the different mechanisms Brahms employs.

Acknowledgments

We thank Christian Cachin for his valuable suggestions that helped to improve our paper. We are grateful to Roie Melamed and Igor Yanover for stimulating discussions of a random walk overlay-based solution.

9. REFERENCES

[1] A. Allavena, A. Demers, and J. E. Hopcroft. Correctness of a gossip based membership protocol. In *ACM PODC*, pages 292–301, 2005.

[2] H. Attiya and J. Welch. *Distributed Computing Fundamentals, Simulations, and Advanced Topics*. John Wiley and Sons, Inc., 2004.

[3] B. Awerbuch and C. Scheideler. Towards a Scalable and Robust DHT. In *SPAA*, pages 318–327, 2006.

[4] G. Badishi, I. Keidar, and A. Sasson. Exposing and Eliminating Vulnerabilities to Denial of Service Attacks in Secure Gossip-Based Multicast. In *DSN*, pages 201–210, June – July 2004.

[5] Z. Bar-Yossef, R. Friedman, and G. Kliot. RaWMS - Random Walk based Lightweight Membership Service for Wireless Ad Hoc Networks. In *ACM MobiHoc*, pages 238–249, 2006.

[6] K. P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal multicast. *ACM Transactions on Computer Systems*, 17(2):41–88, 1999.

[7] E. Bortnikov, M. Gurevich, I. Keidar, G. Kliot, and A. Shraer. Brahms: Byzantine Resilient Random Membership Sampling. Technical Report CCIT Report #688, Technion, Feb 2008.

[8] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations. *J. Computer and System Sciences*, 60(3):630–659, 2000.

[9] T. Condie, V. Kacholia, S. Sankararaman, J. Hellerstein, and P. Maniatis. Induced Churn as Shelter from RoutingTable Poisoning. In *NDSS*, 2006.

[10] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic Algorithms for Replicated Database Management. In *ACM PODC*, pages 1–12, August 1987.

[11] D.Malkhi, Y. Mansour, and M. K. Reiter. On Diffusing Updates in a Byzantine Environment. In *SRDS*, pages 134–143, 1999.

[12] J.R. Douceur. The Sybil Attack. In *IPTPS*, 2002.

[13] P. Th. Eugster, R. Guerraoui, S. B. Handurukande, P. Kouznetsov, and A.-M. Kermarrec. Lightweight probabilistic broadcast. *ACM Transactions on Computer Systems (TOCS)*, 21(4):341–374, 2003.

[14] A. J. Ganesh, A.-M. Kermarrec, and L. Massoulie. Peer-to-Peer Membership Management for Gossip-Based Protocols. *IEEE Trans. Comput.*, 52(2):139–149, 2003.

[15] C. Gkantsidis, M. Mihail, and A. Saberi. Random walks in peer-to-peer networks. In *IEEE INFOCOM*, 2004.

[16] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *JACM*, 33(4):792–807, 1986.

[17] B. P. Hillam. A Generalization of Krasnoselski's Theorem on the Real Line. *Math. Mag.*, 48:167–168, 1975.

[18] M. Jelasity and O. Babaoglu. T-Man: Gossip-Based Overlay Topology Management. In *Proc. of the 3rd International Workshop on Engineering Self-Organising Systems (ESOA)*, July 2005.

[19] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. Gossip-based peer sampling. *ACM TOCS*, 25(3):8, 2007.

[20] H. Johansen, A. Allavena, and R. van Renesse. Fireflies: scalable support for intrusion-tolerant network overlays. In *Proc. of the 2006 EuroSys conference (EuroSys)*, pages 3–13, 2006.

[21] V. King and J. Saia. Choosing a random peer. In *ACM PODC*, pages 125–130, 2004.

[22] C. Law and K. Siu. Distributed construction of random expander networks. In *IEEE INFOCOM*, April 2003.

[23] H. C. Li, A. Clement, E. L. Wong, J. Napper, I. Roy, L. Alvisi, and M. Dahlin. BAR Gossip. In *Proc. of the 7th USENIX Symp. on Oper. Systems Design and Impl. (OSDI)*, pages 45–58, Nov. 2006.

[24] C. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *Proc. of the 16th Intr. Conference on Supercomputing (ICS)*, pages 84–95, 2002.

[25] G. Manku, M. Bawa, and P. Raghavan. Symphony: Distributed hashing in a small world. In *Proc. of the 4th USENIX Symposium on Internet Technologies and Systems (USITS)*, 2003.

[26] L. Massoulie, E. Le Merrer, A.-M. Kermarrec, and A. J. Ganesh. Peer Counting and Sampling in Overlay Networks: Random Walk Methods. In *ACM PODC*, pages 123–132, 2006.

[27] R. Melamed and I. Keidar. Araneola: A Scalable Reliable Multicast System for Dynamic Environments. In *IEEE NCA*, pages 5–14, 2004.

[28] R. C. Merkle. Secure Communications over Insecure Channels. *CACM*, 21:294–299, April 1978.

[29] Y. M. Minsky and F. B. Schneider. Tolerating Malicious Gossip. *Dist. Computing*, 16(1):49–68, February 2003.

[30] Atul Singh, T.-W. Ngan, Peter Druschel, and Dan S. Wallach. Eclipse Attacks on Overlay Networks: Threats and Defenses. In *IEEE INFOCOM*, 2006.

[31] S. Voulgaris, D. Gavidia, and M. van Steen. CYCLON: Inexpensive Membership Management for Unstructured P2P Overlays. *Journal of Network and Systems Management*, 13(2):197–217, July 2005.