Brahms:
Byzantine Resilient Random Membership Sampling

Edward Bortnikov[1] Maxim Gurevich[2] Idit Keidar[2] Gabriel Kliot[3]
Alexander Shraer[2]

September 12, 2008

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 is an attack-resilient gossip-based membership protocol. The second component extracts independent uniformly random node samples from the stream of node ids gossiped by the first. 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 an independent uniform one over time. To our knowledge, no such properties were proven for gossip protocols in the past.

Keywords: Random sampling, gossip, membership, Byzantine faults.

1 Introduction

We consider the problem of sampling random nodes (sometimes called peers) 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 [23, 32, 35, 37], selection of communication partners in gossip-based protocols [13, 18, 21], data sampling, and choosing locations for data caching, e.g., in unstructured peer-to-peer networks [34].

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 the node's *local view*. We consider a dynamic system, subject to *churn*, whereby the set of active nodes changes over time. Local views in such a system must continuously evolve to incorporate new active nodes and to remove ones that are no longer active.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

[1] Yahoo! Research Israel. Email: `ebortnik@yahoo-inc.com`. The work was done when the author was with the Department of Electrical Engineering, The Technion – Israel Institute of Technology.

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

[3] Department of Computer Science, The Technion – Israel Institute of Technology. Email: `gabik@cs.technion.ac.il`.

[4] A shorter preliminary version of this paper appeared in the 27th ACM Symposium on Principles of Distributed Computing (PODC), August 2008.

be effectively maintained with gossip-based membership protocols [1, 21, 22, 26, 43], 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 [19, 33, 39], or maintained (almost) full local views [9, 28], where faulty nodes cannot push correct ones out of the view (please see Section 2 for more detailed discussion of previous work). In contrast, small local views are susceptible to poisoning with entries (node ids) originating from faulty nodes; this is because in a dynamic system, nodes must inherently accept new ids and store them in place of old ones in their local views. In Section 3, we illustrate that traditional gossip-based membership is highly vulnerable to adversary attacks, which can quickly poison the entire views of correct nodes.

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, 22, 26], 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.

This paper addresses these challenges. In Section 4, we present Brahms, a membership service that stores a sub-linear number of ids (e.g., $\Theta(\sqrt[3]{n})$ in a system of size n) at each node, and provides each node with independent random node 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 susceptible to attacks that may *bias* the views, i.e., cause certain nodes to be over-represented in views while others are under-represented (we precisely quantify the extent of this bias mathematically); and (3) to correct this bias at each node. Specifically, each node maintains, in addition to the gossip-based local view, an unbiased *sample list* of nodes.

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. Sampler uses min-wise independent permutations [14], and stores one element of the stream at a time. In Brahms, the data stream is comprised of gossiped ids, from which Samplers obtain independent uniformly random id samples, and store them in the sample list. By using such *history samples* from the sample list 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, with high probability. 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 history samples, e.g., 10%. Therefore, the protocol can still deal effectively with churn.

In Section 5, we define the attacker's goals and the corresponding attack strategies, under which we evaluate Brahms. 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. This goal is achieved by a *uniform attack*, whereby the attacker equally divides its power among all correct 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 [1]. Since samples help prevent isolation, we analyze the most adverse circumstances, where an attack is 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.

One of the important contributions of this paper is our mathematical analysis, 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 first show (in Section 6) that whenever the set of nodes remains connected, the sample lists converge to independent uniformly random selections from among all nodes. We further show that if views are of size $\Omega(\sqrt[3]{n})$, then the convergence rate is bounded independently of the system size. Section 7 then analyzes the local views generated by the gossip process and shows that under certain circumstances, they preserve the connectivity required for uniform samples.

Specifically, for the attack goal of maximizing the representation of faulty ids (Section 7.1), we show that under certain conditions on the adversary, even without using history samples, the portion of faulty ids in local views generated by Brahms's gossip process 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).

Next, we consider the goal of isolating a node (Section 7.2). The key to proving that Brahms prevents, with high probability, 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 8), 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.

2 Related Work

We are not familiar with any previous work explicitly dealing with random node sampling in a Byzantine setting. We next review previous work on Byzantine membership (Section 2.1), node sampling and sampling from data streams in benign settings (Section 2.2), and on the related problem of Byzantine-resilient overlay construction (Section 2.3).

2.1 Byzantine Membership

Most previous Byzantine-tolerant gossip based protocols have either considered static settings where the full membership is known to all [19, 33, 39] or focused on maintaining full local views [9, 28] rather than partial samples. The only exception we are aware of is the *Secure peer sampling service (SPSS)* [27].

This paper considers an attack on gossip-based membership, whereby the attackers send many faulty ids to correct nodes. The proposed service, SPSS, mitigates such attacks by gathering statistics about over-represented node ids. Over-represented ids are deemed faulty, and are removed from views. However, as the authors show, the effectiveness of this approach is limited to a small number of malicious nodes (in the order of the view size). In contrast, Brahms tolerates $\Theta(n)$ Byzantine failures with views of size $\Theta(\sqrt[3]{n})$. Moreover, SPSS is only evaluated in simulations and no formal proofs of its properties are given.

2.2 Node Sampling and Sampling from Streams

Gossip-based membership [1, 21, 22, 26, 43] is a robust and efficient technique for maintaining small, (typically logarithmic-size) local views in the presence of benign failures, ensuring a low probability for partitions [1], and an eventual uniform average representation of nodes in local views [1, 22, 26]. However, even in benign settings, it does not ensure that every node eventually obtains a uniform random sample as Brahms does. Furthermore, as we show in Section 3, it is vulnerable to Byzantine failures.

Proven near-uniform node samples can be obtained using a Random Walk (RW). Random walks are often used for peer sampling and counting in peer-to-peer networks; their outcome is used for overlay construction and for the maintenance of partial local membership views [23, 32, 36, 10]. RWs have also been recently proposed to combat Sybil attacks [44] (in which malicious nodes forge identities in order to impose as multiple nodes). However, the correctness of RW-based sampling depends on the network topology. If the actual topology is different from the assumed one, then the sample produced by the RW may be far from uniform [23]. In contrast, Brahms does not assume any specific network topology. Its sole assumption is that the graph formed by correct nodes is connected. Moreover, using RWs in a Byzantine setting is problematic, because a faulty node anywhere along the path of a random walk can render the information obtained in this walk useless.

King and Saia [31] present a method for (proven) uniform sampling in a distributed hash table (DHT) like Chord, which is not resilient to Byzantine attacks [15].

Uniform sampling is related to the problem of load-balancing data over nodes in a DHT [29, 30], which strives to achieve the following: given a data item, the node that stores it should be chosen uniformly at random. Typically in DHTs, all nodes use the same hash function for mapping data to nodes, in order to facilitate data location. This approach results in an unbalanced load, which can be improved by creating multiple virtual nodes for each real node [29], or by dynamic re-balancing of the key space [30]. In contrast, our application does not require all nodes to agree on a common hash function. Brahms ensures balanced sampling (i.e., that every correct node appears with the same probability in every sample of a correct node), by using random (or pseudo-random) hash functions, picked independently by each node.

Various previous works have dealt with benign sampling, e.g., from unbiased data streams [42] or from biased data streams with a *known bias* [8, 17]. Other works have focused on unbiasing data samples from a *random access* medium rather than a stream [11], or counting the number of distinct elements in a (possibly biased) stream, e.g., [2, 12]. However, we are not aware of previous work providing uniform samples from a data stream with an unknown bias, as our Sampler component does.

2.3 Byzantine Resilient Overlays

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. As noted above, previous Byzantine-tolerant gossip-based membership solutions have maintained (almost) full local views [9, 28] or withstood only weak attacks [27].

Several recent works have focused explicitly on securing overlays, mostly structured ones, also attempting to ensure that all correct nodes may communicate with each other using the overlay, i.e., to prevent the *eclipse attack* [40, 41], where routing tables of correct nodes are gradually poisoned with links to adversarial nodes. These works typically assume that faulty nodes cannot control their ids, which is implemented by using mechanisms such as a CA [28, 15, 40] or a cryptographic random number generator [5]. Brahms also assumes that the number of ids controlled by faulty nodes is bounded, but does allow faulty nodes to control their own ids.

Singh et al. [40, 41] proposed a defense against eclipse attacks in structured overlays, based on the observation that when an eclipse attack is launched, the in-degree of faulty nodes is likely to be higher than the average in-degree of correct nodes. The idea is, therefore, to audit node degrees, and choose neighbors whose degree is below some threshold. Unlike Brahms, this does not result in a uniform random selection of neighbors. Finally, this approach is not appropriate for unstructured overlays.

Other solutions for Byzantine-tolerant structured overlays maintain *constrained* routing tables, where faulty nodes are not over-represented, in addition to the regular routing tables, in which faulty nodes might by over-represented [15, 16]. This approach resembles our unbiasing of the local views. However, the constrained table is not proven to be a uniform sample of the nodes. Moreover, unlike Brahms, these solutions require either frequent id re-assignment [16] or a secure way of measuring network distances [15].

Awerbuch and Schiedeler propose Byzantine-tolerant structured overlay constructions [4, 6, 7], with logarithmic-size views. However, unlike Brahms, they either require constant re-joining [4] or employ a complex cryptographic random-number generator [5] and need id re-distributions upon every join [6, 7]. Moreover, these solutions are much more complex than Brahms.

Finally, unlike the works mentioned above, we present a *general* sampling technique, one application of which is building Byzantine-resilient unstructured overlays.

3 Model, Goal, and Challenges

We describe the system model, outline our design goal, and illustrate the challenges in achieving it.

3.1 System Model

We consider a collection U of nodes, each identified by a unique id. We do not constrain the way in which node ids are chosen, e.g., we do not assume a certification authority. Nevertheless, nodes are not allowed to use multiple ids, which rules out massive Sybil attacks [20] (where one faulty node can impersonate as many nodes). Individual nodes do not know the entire set of nodes U. Rather, each node has some initial knowledge of a small set of other nodes, so that the graph induced by this knowledge is connected.

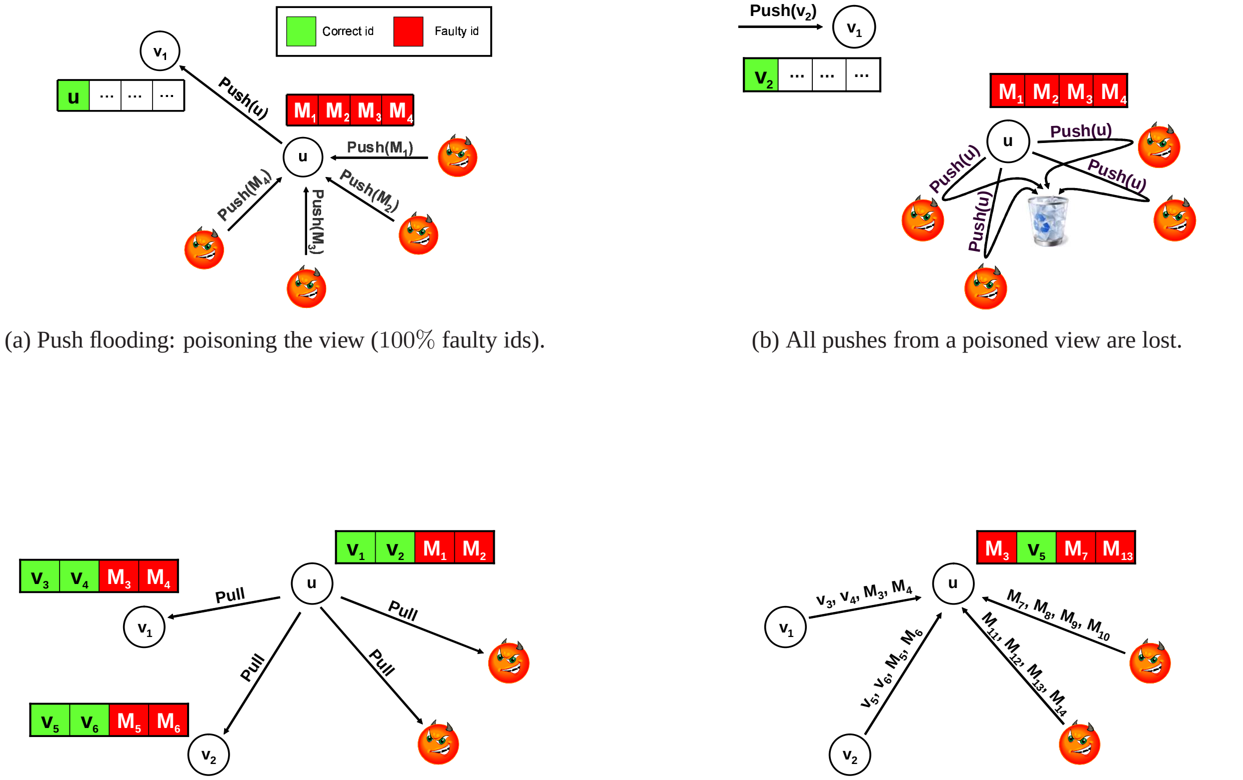
The system is subject to churn, i.e., nodes can join and leave (or crash) dynamically. A node that has joined and did not leave or crash is *active*. A correct active node follows the protocol, whereas faulty active nodes may exploit the protocol to attack other nodes. Every pair of nodes can communicate with each other directly through bidirectional reliable links, provided that they know each other's ids. A node can determine the source of every incoming message, and cannot intercept messages addressed to other nodes (this is the standard ”unauthenticated“ Byzantine model [3]). For simplicity of the analysis, we assume a synchronous model with a discrete global clock, zero processing times, and message latencies of a single time unit.

3.2 Design Goal

Each node maintains a list of node ids called *sample list*. Intuitively, each entry in the sample list should converge to an independent uniform random sample of the active nodes. However, the notion of a uniform sample is only meaningful when applied to a fixed set, and not to an ever-changing one. Therefore, for the sake of specifying our protocol's goal, we assume that there is a time T_0 at which churn ceases, and require each entry in the sample list to converge to an independent uniform random sample of the nodes that are active from time T_0 onward.

3.3 Design Challenges - Vulnerabilities of Gossip-Based Membership

Gossip-based protocols (e.g., [1, 26]) 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



(a) Push flooding: poisoning the view (100% faulty ids).

(b) All pushes from a poisoned view are lost.

(c) Skewed pulls: initially, 50% faulty ids.

(d) Skewed pulls: after one round, 75% faulty ids.

Figure 1: **Malicious attacks on traditional gossip protocols using push and pull requests. (a) Faulty nodes flood a correct node u with pushes, and totally poison its view. (b) Node u with a totally poisoned view sends pushes only to faulty nodes, and ceases being represented in the views of other correct nodes. (c) Node u pulls views from two correct nodes with 50% correct ids, and two faulty nodes. (d) The faulty nodes return only faulty ids, thus poisoning 75% of u's view.**

view. The primary goal of a gossip-based membership service is to preserve connectivity of the overlay induced by the nodes' views; that is, to avoid network partitions. Note that connectivity is also a prerequisite for random sampling, since nodes in distinct connected components have zero probability for learning about each other.

Nodes propagate membership information through two primitives, *push* – unsolicited sending of a node's id to some other node in the sender's view, and *pull* – request-reply retrieval of another node's view. Pushes allow new active nodes to become represented in other nodes' views, whereas pulls spread knowledge about active nodes throughout the system. Allavena et al. [1] have shown that both are needed in order to avoid partitions and star-like topologies with high probability. They have further shown that in benign gossip that uses both pull and push, the expected time until a network partition is exponential in the squares of both the view size and the isolated component's size, under reasonable assumptions. Extensive empirical studies [21, 26] have validated that gossip-based protocols maintain connectivity in benign setting in practice.

We now illustrate that traditional gossip is not resilient to adversarial pushes and pulls. For example, an adversary can choose to over-represent the faulty ids in the views of some correct nodes. We illustrate how both push and pull can be abused so as to lead to rapid poisoning of views at all correct nodes.

Push flooding. The adversary can flood correct nodes with pushes of faulty ids, and thus to cause all views of correct nodes to quickly become poisoned with faulty ids. To mitigate push flooding, we assume that the sending rate of Byzantine nodes can be constrained. That is, there is a mechanism that makes it costly for nodes to send designated messages, which we call *limited send*, and use for push messages. This mechanism can be implemented in different ways, e.g., computational challenges like Merkle's puzzles [38], virtual currency, etc. Note that employing limited send is necessary but not sufficient: while such rate-limiting prevents the adversary from flooding all correct nodes in parallel, an attacker can still target correct nodes one by one. This process is illustrated in Figure 1: first, the attacker focuses on one node u, and leads to complete poisoning of its view (Figure 1(a)). For simplicity, Figure 1(a) shows the effect of this attack on a push-only protocol; when pull and push are combined, a similar degradation occurs, although it might take longer, as we show in Section 7.

Once the attacker succeeds in poisoning u's entire view, all of u's pushes are sent to faulty nodes (Figure 1(b)), and consequently, u disappears from the views of other correct nodes. Note that even in push-pull gossip, the representation of ids that are not pushed to correct nodes is exponentially decaying, since a constant part of each view is updated with new pushes. Once this occurs, u is isolated from the system, and the attacker can proceed to attack additional nodes.

Skewed pull responses. 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 in the system.

The effect of this attack on a purely pull-based gossip protocol is illustrated in Figures 1(c) and 1(d). In this example, the system begins a gossip round in a state where 50% of ids in all views are faulty, (Figure 1(c)), and we see that at the end of the same round, 75% of the ids in a typical node's view are faulty (Figure 1(d)).

These 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, an attacker can disintegrate the entire overlay, thus precluding peer sampling completely. Brahms adopts a two-layer approach to this problem. As a first step, we guarantee, with high probability, that the attacker cannot isolate correct nodes, that is, the maximum bias to their views is bounded. As a second step, we correct the incurred bias through local uniform sampling.

4 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 traversed the node (Section 4.1). The *gossip* component is a distributed protocol that spreads ids across the network (Section 4.2), and maintains a dynamic *view* $\mathcal{V}$. We denote the size of $\mathcal{V}$ by ℓ_1 and the size of $\mathcal{S}$ by ℓ_2. 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 undefined (denoted $\perp$).

4.1 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 a time. The output is a uniform random choice of one of the unique inputs witnessed thus far. For example, assume some id, id_1, appears only once in a certain input stream, while another id, id_2, appears 1000 times in the same stream; Sampler's output on this stream has an equal probability of being id_1 as for being id_2.

```
function Sampler.init()
    $h \leftarrow$ randomPRF(); $q \leftarrow \perp$
function Sampler.next($elem$)
    if $q = \perp \vee h(elem) < h(q)$ then
        $q \leftarrow elem$
function Sampler.sample()
    return $q$
```

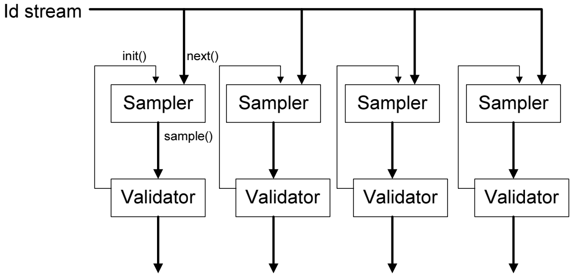


Figure 2: **Uniform sampling from an id stream in Brahms. (a) Sampler's pseudo-code. (b) Sampling and validation of ℓ_2 ids.**

Sampler uses *min-wise independent* permutations [14]. 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., [24]) are considered an excellent practical approximation of min-wise independent permutations, provided that $|U|$ is large, e.g., 2^{160}.

The pseudo-code of Sampler appears in Figure 2(a). It selects a random min-wise independent function h upon initialization, and applies it to all input values (in the `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 (see Figure 2(b)), which select hashes independently. The same stream of ids observed by the node 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). Thus, when churn ceases, each sample converges to an independent uniform random selection from among the active nodes.

4.2 Gossip

Brahms's view is maintained by a gossip protocol as shown in 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].

Brahms uses parameters $\alpha > 0$, $\beta > 0$, and $\gamma > 0$ that satisfy $\alpha + \beta + \gamma = 1$, to control the portion of pushed ids, pulled ids, and history samples in the new view, respectively. 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 (Lines 22-26). At the end of each round, $\mathcal{V}$ and $\mathcal{S}$ are updated with fresh ids. While all received ids are streamed to $\mathcal{S}$ (Line 37), re-computing $\mathcal{V}$ requires extra care, to protect against poisoning of the views with faulty ids. Brahms offers the following set of techniques to 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

```
 1: $\mathcal{V}$ : tuple[$\ell_1$] of Id
 2: $\mathcal{S}$ : tuple[$\ell_2$] of Sampler
 3: Initialization($\mathcal{V}_0$):
 4:   $\mathcal{V} \leftarrow \mathcal{V}_0$
 5:   for all $1 \leq i \leq \ell_2$ do
 6:     $\mathcal{S}[i]$.init()
 7:   updateSample($\mathcal{V}_0$)
 8: {Stale sample invalidation}
 9: periodically do
10:   for all $1 \leq i \leq \ell_2$ do
11:     if probe($\mathcal{S}[i]$.sample()) fails then
12:       $\mathcal{S}[i]$.init()
13: {Auxiliary functions}
14: function updateSample($\mathcal{V}$)
15:   for all $id \in \mathcal{V}$, $1 \leq i \leq \ell_2$ do
16:     $\mathcal{S}[i]$.next($id$)
17: function rand($\mathcal{V}, n$)
18:   return $n$ random choices from $\mathcal{V}$
19: {Gossip}
20: while true do
21:   $\mathcal{V}_{push} \leftarrow \mathcal{V}_{pull} \leftarrow \emptyset$
22:   for all $1 \leq i \leq \alpha\ell_1$ do
23:     {Limited push}
24:     send_lim $\langle$“$push\_request$“$\rangle$ to rand($\mathcal{V}, 1$)
25:   for all $1 \leq i \leq \beta\ell_1$ do
26:     send $\langle$“$pull\_request$“$\rangle$ to rand($\mathcal{V}, 1$)
27:   wait(1)
28:   for all received $\langle$“$push\_request$“$\rangle$ from $id$ do
29:     $\mathcal{V}_{push} \leftarrow \mathcal{V}_{push} \circ \{id\}$
30:   for all received $\langle$“$pull\_request$“$\rangle$ from $id$ do
31:     send $\langle$“$pull\_reply$“$, \mathcal{V}\rangle$ to $id$
32:   for all received $\langle$“$pull\_reply$“$, \mathcal{V}'\rangle$ from $id$ do
33:     if I sent the request, and this is the first reply then
34:       $\mathcal{V}_{pull} \leftarrow \mathcal{V}_{pull} \circ \mathcal{V}'$
35:   if $(|\mathcal{V}_{push}| \leq \alpha\ell_1 \ \wedge\ \mathcal{V}_{push} \neq \emptyset \ \wedge\ \mathcal{V}_{pull} \neq \emptyset)$ then
36:     $\mathcal{V} \leftarrow$ rand($\mathcal{V}_{push}, \alpha\ell_1$) $\circ$ rand($\mathcal{V}_{pull}, \beta\ell_1$) $\circ$ rand($\mathcal{S}, \gamma\ell_1$)
37:   updateSample($\mathcal{V}_{push} \circ \mathcal{V}_{pull}$)
```

Figure 3: **The pseudo-code of Brahms.**

representation would decay exponentially [1]. The protocol employs limited sending of push messages (performed by send_lim), hence the system-wide 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 such a *targeted attack* by blocking the update of $\mathcal{V}$. Namely, if more than the expected $\alpha\ell_1$ pushes are received in a round, Brahms does not update $\mathcal{V}$ in that round (Line 35). 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, the adversary cannot block all correct nodes simultaneously, i.e., some nodes make progress even under an attack.

Controlling the contribution of pushes versus 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.

[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, is expected to happen soon

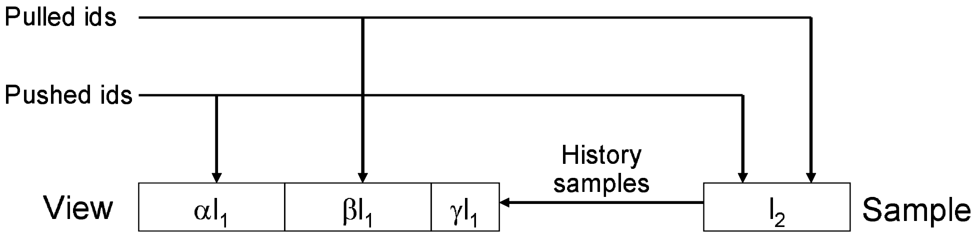


Figure 4: **View re-computation in Brahms.**

Therefore, we use 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.

Parameter settings. Brahms's parameters control 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 so small as to make the feedback ineffective. We show in Section 8 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 attacked node's sample stabilizes. We study their impact in Section 7, where we show that choosing $\ell_1, \ell_2 = \Theta(\sqrt[3]{n})$ suffices to protect even nodes that are attacked immediately upon joining the system.

5 Definitions and Attack Models

In the next two sections we present mathematical analysis, which, like previous studies [1], makes some simplifying assumptions. The theoretical results are then validated through extensive simulations (in Sections 7 and 8).

We study the asymptotical properties of a system of n active nodes, after a point T_0 at which 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 pushes that all faulty nodes can jointly send (using limited send) in a single round (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 as follows:

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

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 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 (though the ratio of faulty ids in the view is not necessarily bounded by f). We further assume that before an attack starts, the in-degrees and out-degrees of all correct nodes are (roughly) equal. This property is an approximation of reality – Jelasity et al. [26] have shown that benign gossip leads to a low variance in in-degrees. Our simulations demonstrate that our results, which use this assumption, are valid.

We assume that Byzantine nodes always prefer to increase their representation rather than decrease it. That is, they push only faulty ids to correct nodes and always return faulty ids in pulls. Likewise, faulty nodes always respond to probe requests, to avoid invalidation.

We study two representative Byzantine attacks. The first, called *balanced attack*, distributes the faulty pushes evenly among the correct nodes. We show in Section 7 that this attack maximizes the system-wide representation of faulty ids at each round. The second attack, called *targeted*, focuses an increased portion of faulty pushes on a small subset of correct nodes, in an attempt to isolate them from the overlay. Section 7 analyzes the dynamics of both attacks, and demonstrates that in both cases, Brahms prevents the overlay's partitioning, with the right choice of parameters.

6 Analysis - Sampling

In this section we analyze the properties of a sample $\mathcal{S}_u$ of a correct node u. Let $s = \mathcal{S}_u[i]$ be a sampler element for some correct u and some i. Recall that s employs a permutation $s.h$, chosen independently at random. Let $s(t)$ denote 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 n ids (we neglect collisions for the sake of the definition). Note that s^* can be either a correct or a faulty id. In Section 6.1 we show that s eventually converges to an independent uniform random sample. In Section 6.2 we analyze how fast a node obtains at least one correct perfect sample, as needed for self-healing. Section 6.3 discusses scalability, namely, how to choose view sizes that ensure a constant convergence time, independent of system size. For readability, some formal proofs are deferred to Appendix A, while this section overviews the proof approach.

6.1 Eventual Convergence to Uniform Sample

Consider a sampler $s \in S_u$ of node u. The perfect id of s, s^*, samples ids uniformly at random by definition of min-wise independent family of hash functions. Thus, our goal is to prove that s eventually holds s^*. Obviously, for s to be able to sample some correct node v, the id of v has to reach u. To allow for such 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 $t \geq T_0$. The following theorem shows that under this assumption each id eventually has the same probability to be sampled by s.

Theorem 6.1 *If $\mathcal{N}(t)$ remains weakly connected for each $t \geq T_0$, then, for all $v \in \mathcal{C}$,*

$$\Pr(s(t) = v) \rightarrow_{t\rightarrow\infty} \frac{1}{n}.$$

Proof : Let u be an active node at time T_0. Then u sends pushes in every round $> T_0$ (recall that we assume this also for faulty nodes, see Section 5). We now consider a time $t > T_0$, and study the spreading of u's id as a random process from time t onward. We denote by $See_u(t)$ the set of nodes that have a nonzero probability of including u in their local views at time t. That is,

$$See_u(t) \triangleq \{v | \Pr(u \in \mathcal{V}_v(t)) > 0\}.$$

We will show that eventually, $See_u(t)$ includes all active nodes.

Proposition 6.2 *Every id $v \in See_u(t)$ is also in $See_u(t')$ at any time $t' > t$.*

Proof : The probability that u is the perfect sample of every sampler is $1/n$, and hence, once u has a nonzero probability of being included in $\mathcal{V}_v(t)$, it has a nonzero probability of being sampled in $\mathcal{S}_v(t)$. Since a perfect id remains in $\mathcal{S}_v(t)$ forever, u has a nonzero probability of being added back to $\mathcal{V}_v$ as a history sample at every later time t'. □

Proposition 6.3 *As long as See_u does not contain all active nodes, See_u grows by at least one id each 3 rounds.*

Proof : Consider a time $t > T_0$. If $See_u(t)$ includes all active nodes, we are done. Otherwise, by Proposition 6.2 See_u never shrinks, so it remains to show that at least one new id is added to it every 3 rounds.

By connectivity of $\mathcal{N}(t)$, there is a path in $\mathcal{N}(t)$ from every node in $See_u(t)$ to every node that is not in $See_u(t)$. Consider an edge between some $u_1 \in See_u(t)$ and some $u_2 \notin See_u(t)$. We now show that u_2 is added to See_u within at most 3 rounds. That is, $u_2 \in See_u(t+3)$. There are 4 possible cases, depending on the type of edge between u_1 and u_2.

1. $\mathbf{u_1 \in \mathcal{V}_{u_2}(t)}$. There is a positive probability for u_2 to pull from u_1 at round t, and since $u_1 \in See_u(t)$, u has a nonzero probability of being in $\mathcal{V}_{u_1}(t)$ and being returned in the pull and included into $\mathcal{V}_{u_2}(t+1)$, and we are done.

2. $\mathbf{u_2 \in \mathcal{V}_{u_1}(t)}$. There is a positive probability for u_1 to push to u_2 at round t, leading to u_1 being in $\mathcal{V}_{u_2}(t+1)$, and the proof continues as case 1.

3. $\mathbf{u_1 \in \mathcal{S}_{u_2}(t)}$. There is a positive probability for u_1 to be added to $\mathcal{V}_{u_2}(t+1)$ as a history sample, and the proof continues as case 1.

4. $\mathbf{u_2 \in \mathcal{S}_{u_1}(t)}$. There is a positive probability for u_2 to be added to $\mathcal{V}_{u_1}(t+1)$ as a history sample, and the proof continues as case 2.

We conclude that for every u and every $t > T_0$, $See_u(t+3)$ contains at least one new id that was not included in $See_u(t)$. □

By Proposition 6.3, for every u and every $t > T_0$, $See_u(t+3n)$ includes all active nodes. That is, for every node u, and at every time t in the run of the protocol after T_0, there is a positive probability for u to appear in every other node's stream by time $t+3n$. Since the event of u being in another node v's stream by time $t+3n$ has nonzero probability of occurring starting from every time $t > T_0$, eventually, with probability 1, there will be some t when this even will occur. □

Recall that we assumed (Section 5) that faulty active nodes always seek to maximize their representation, and therefore, send pushes to correct nodes and respond to invalidation probes. Therefore, they appear in the gossip streams, and are sampled with the same probability as correct nodes. In a system where this

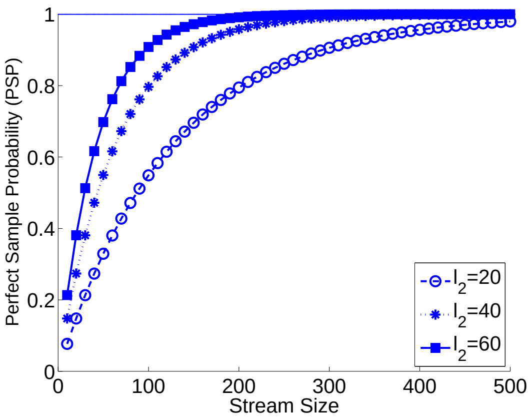


Figure 5: **Growth of the probability to observe at least one correct perfect sample (Perfect Sample Probability - PSP) with the stream size, for 1000 nodes, $f = 0.2$, and $\rho = 0.4$.**

assumption does not hold, and faulty active nodes may refrain from responding to pings, the probability that a correct id is sampled converges to the range $[\frac{1}{n}, \frac{1}{(1-f)n}]$ or $[\frac{1}{n}, \frac{1}{|\mathcal{C}|}]$, instead of exactly to $1/n$ as stated above.

The next lemma discusses the convergence rate of samples.

Lemma 6.4 *From T_0 onward, for each correct node u, the expected fraction of samplers in S_u that output their perfect ids 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, for $t > T_0$. Note that $D(t)$ contains only ids that are active after T_0, since inactive ids are invalidated and no invalidations happen after T_0 (recall that at time T_0 churn ceases). Then, for each u's 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 each $t' \geq t$, the lemma follows. □

6.2 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 set of ids that u 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. For the sake of proving the lower bound, we made worst-case assumption: 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 time t. Our analysis depends on the number of unique ids observed by u, rather than directly on Λ. Obviously, it is unrealistic to expect our gossip protocol to produce independent uniform random samples (cf. [26]). 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 includes as many random unique ids as 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 [26]. We empirically measured ρ to be about 0.4 with our gossip protocol (see Section 7.2).

In the following lemma we study the dependency between the probability of a sampler to output its correct perfect id, the number $\Lambda(t)$ of (non-unique) correct ids streamed into the sampler, and the stream deficiency factor ρ.

Proposition 6.5 *Let s be a sampler. Then, for $|\mathcal{C}| \gg 1$ and for each $t > T_0$,*

$$\Pr(s(t) \neq s^* | s^* \in \mathcal{C}) = e^{-\frac{\rho\Lambda(t)}{|\mathcal{C}|}}.$$

Proof idea. A sampler does not output its correct perfect id only if that id did not occur in the stream. We calculate the probability of this event as a function of the effective number $\rho\Lambda(t)$ of independent and uniformly distributed correct ids in the stream by time t. The full proof appears in Appendix A.1.

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:

Lemma 6.6 *Let u be a random correct node. Then, for $t > T_0$,*

$$PSP_u(t) \geq 1 - \left((1-f)e^{-\frac{\rho\Lambda(t)}{|\mathcal{C}|}} + f\right)^{\ell_2}.$$

Proof : Since u has ℓ_2 independent samplers, the probability of each one to have a correct perfect id is $\Pr(s^* \in \mathcal{C}) \geq 1-f$. Similarly, $\Pr(s^* \notin \mathcal{C}) \leq f$. Based on Proposition 6.5, the probability of $s(t)$ not being a correct perfect id is

$$\begin{aligned}\Pr(s(t) \neq s^* \vee s^* \notin \mathcal{C}) &= \Pr(s(t) \neq s^* | s^* \in \mathcal{C}) \Pr(s^* \in \mathcal{C}) + \Pr(s^* \notin \mathcal{C}) \\ &\leq (1-f)e^{-\frac{\rho\Lambda(t)}{|\mathcal{C}|}} + f.\end{aligned}$$

The perfect sample probability $PSP_u(t)$ equals 1 minus the probability of each of ℓ_2 samplers not outputting a correct perfect id, that is:

$$PSP_u(t) \geq 1 - \left((1-f)e^{-\frac{\rho\Lambda(t)}{|\mathcal{C}|}} + f\right)^{\ell_2}.$$

□

Figure 5 illustrates the dependence of PSP on the stream size $\Lambda(t)$ and on ℓ_2. For example, we see that when the sample size is $40 = 4\sqrt[3]{n}$ (for $n = 1000$, $f = 0.2$), and the portion of unique ids in the stream is $\rho = 0.4$, a correct perfect sample is obtained, with probability close to 1, after 300 ids traverse the node.

6.3 Scalability

From Lemma 6.6, we see that PSP depends on Λ and ℓ_2. To get a higher PSP, we can increase either of them. 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 nodes, n, increases. When a node has ℓ_2 samplers, and f is fixed, $\Omega(\ell_2)$ of them have correct perfect sample, s^*, with high probability. Therefore, by Proposition 6.5, the probability of at least one of these $\Omega(\ell_2)$ samplers outputting its perfect id satisfies, with high probability

$$PSP_u(t) \geq \Omega(1 - (e^{-\frac{\rho\Lambda(t)}{n}})^{\ell_2}) = \Omega(1 - e^{-\frac{\rho\Lambda(t)\ell_2}{n}}).$$

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 when $\ell_1 = \Omega(\sqrt[4]{n})$ 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 $\ell_2 = \Omega(\frac{n}{\log^2 n})$. When choosing parameter values for our simulations later in the paper, we shall use $\ell_2 = \ell_1 = c\sqrt[3]{n}$ for some c.

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 node's view (complement to out-degree)
$y_u(t)/\tilde{y}_u(t)$		occurrences of the node in views of correct nodes (in-degree)
$g_u^{\text{push}}(t)/\tilde{g}_u^{\text{push}}(t)$	$g^{\text{push}}(t)/\tilde{g}^{\text{push}}(t)$	correct ids pushed to the 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 the 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 the 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 the node

Table 1: **Random variable definitions.**

7 Analysis – Overlay Connectivity

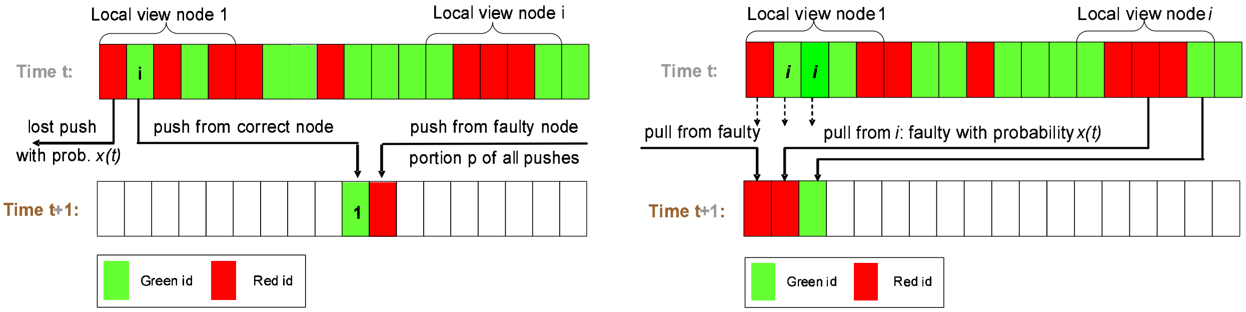
We now prove that Brahms, with appropriate parameter settings, maintains overlay connectivity despite the attacks defined in Section 5, satisfying the prerequisite for Theorem 6.1.

We study two possible adversary targets. The first target, addressed in Section 7.1, is increasing the global representation of faulty ids. 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 at the end of the round, among all strategies. (A similar approach of analyzing the adversary's damage in a single round was taken, e.g., in [33].) We proceed by analyzing the effect of this attack, namely the evolution of the system-wide fraction of faulty ids at the end of each round. We further show that under certain conditions this fraction converges to a value that is strictly smaller than 1. That is, this attack alone can not partition the network.

We next consider an attack that attempts to partition the network (rather than increase the faulty nodes' 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 7.2 shows that had Brahms not used history samples, correct nodes could have been isolated in this manner. However, Brahms withstands such *targeted* attacks, even if they start immediately upon a node's join, when the node is not represented in other views and has no history. The key property we prove is that Brahms's gossip prevents isolation long enough for history samples to become effective. This section employs stochastic analysis backed by simulations.

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. The variable x denotes the number of faulty ids in the node's view ($1 - x$ is the node's out-degree in the overlay of correct nodes) and y denotes the number of occurrences of node's id in the views of correct nodes (the node's in-degree). Their fractions in views are denoted with ˜ above. Correct (resp., faulty) ids propagated through pushes and pulls are denoted g (for green) (resp., r (for red)), with appropriate superscripts for push and pull.

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 always use $p = f$. A different subset of faulty nodes push their ids to a given correct node in each round, using a round-robin schedule.



(a) Impact of push

(b) Impact of pull

Figure 6: **Fixed point analysis illustration.**

7.1 Balanced Attack - Increasing Global Representation of Faulty Ids

We study the balanced attack, which shares the adversarial pushes evenly among all correct nodes. Lemma B.1 in Appendix B.1 shows that this attack is most efficient in maximizing the system-wide representation of faulty ids in a single round. Intuitively, this result is explained as follows. The probability for an adversary's message to be accepted, (i.e., selected in `rand` on line 36 in Figure 3), in a given round is maximized when the message reaches a node that receives a minimal number of pushes in that round; over-loading nodes only reduces the adversary's messages chances of being accepted. However, the adversary has no information about the number of correct pushes received by any particular correct node. Moreover, the expected number of received pushes is the same at all the correct nodes. Therefore, the adversary maximizes the number of faulty pushes expected to be accepted in a round by distributing them evenly among correct nodes.

We now proceed and study the system dynamics when a balanced attack is applied over multiple rounds. The analysis makes two simplifying assumptions. First, we ignore the effect of view blocking (Figure 3, Line 35). Note that this is a worst-case assumption, which only accelerates the deterioration of correct views. Second, we assume that the balanced attack preserves the in-degrees and out-degrees of all correct nodes equal over time, since it does not distinguish between correct nodes. Formally,

Assumption 7.1 *For all $u \in \mathcal{C}$ and all $t \geq T_0$: $x_u(t) = x(t)$, and $y_u(t) = \ell_1 - x_u(t)$.*

Our extensive simulations closely validate the theoretical results obtained using these two assumptions. Throughout this section we assume $0 < p < 1$. The other cases (all pushes are faulty or no faulty pushes at all) are not interesting.

7.1.1 The evolution of $\tilde{x}(t)$

We study the evolution of the ratio of faulty node ids in views, $\tilde{x}(t)$, over time. 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 $t \gg T_0$.

Definition 7.1 *Assuming a fixed $p \in (0,1)$ we define:*

$$\psi(x) \triangleq \alpha \frac{p}{p + (1-p)(1-x)} + \beta((1-x)x + x) + \gamma f.$$

Lemma 7.2 *For $t \gg T_0$, and $p \in (0,1)$, the expected system-wide fraction of faulty ids evolves as*

$$\mathrm{E}(\tilde{x}(t+1)) = \mathrm{E}(\psi(\tilde{x}(t))).$$

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 (Figure 3, Lines 36 and 17–18) 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 $\tilde{r}^{\text{push}}(t)$, the probability of a received push to be faulty. Each correct node wastes an expected fraction $\tilde{x}(t)$ of its $\alpha\ell_1$ 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 (Assumption 7.1), 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 (i.e., nodes v s.t. $u \in \mathcal{V}_v(t)$). Since this number is large, the number of received correct pushes is approximately equal to its expectation at 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 a portion $1-p$ out of all pushes (by definition of p). Hence, the total number of pushes is $\frac{\alpha\ell_1}{1-p}|\mathcal{C}|$, and the number of faulty pushes is $\frac{p\alpha\ell_1}{1-p}|\mathcal{C}|$. Since faulty pushes are perfectly balanced among the correct nodes, u receives exactly $r_u^{\text{push}}(t) = \frac{p}{1-p}\alpha\ell_1$ faulty pushes, and their fraction among all received pushes is:

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

Out of all push-received ids u stores a fraction of α in its view. 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 (Assumption 7.1), 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 $\beta((1-\tilde{x}(t))\tilde{x}(t)+\tilde{x}(t))$.

Finally, since $t \gg T_0$, all history samples are perfect (the ratio of faulty ids in them is f). Hence, their expected contribution to $\tilde{x}(t+1)$ is γf, and the claim follows. □

7.1.2 Fixed point existence

We now show that the system has a stable state. A value $\hat{x}$ is called a *fixed point* of $\tilde{x}(t)$ if $\psi(\tilde{x}(t)) = \tilde{x}(t) = \hat{x}$. To find the potential fixed points, we substitute this into the equation from Lemma 7.2. The following Claim immediately follows from our definitions.

Claim 7.3 *For $\alpha, \beta, \gamma, f \in [0,1]$ and $p \in (0,1)$, every real root $0 \le \hat{x} \le 1$ of the equation $\psi(\tilde{x}(t)) = \tilde{x}(t)$ is a fixed point of $\tilde{x}(t)$.*

To shed more light on the balanced attack's dynamics, we study the fixed point values under specific combinations of α, β and γ. Simplifying the equation $\psi(x) = x$, we get $h(x) = 0$, where:

$$h(x) = \beta(1-p)x^3 + (2\beta p - 3\beta - p + 1)x^2 + (\gamma f p - \gamma f + 2\beta - 1)x + \alpha p + \gamma f.$$

By Claim 7.3, the fixed point $\hat{x}$ is a root of $h(\tilde{x})$. We first establish a number of useful observations regarding the functions $\psi(x)$ and $h(x)$ that will be used throughout our analysis, here and in Appendix B.2. They can be shown by straightforward calculus.

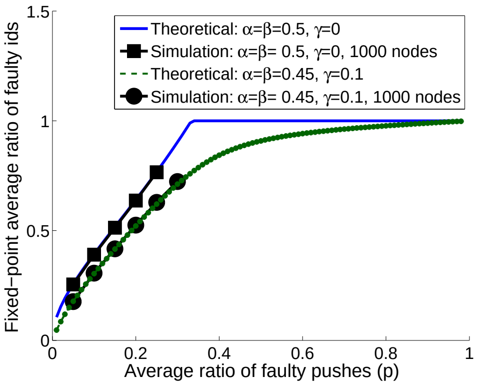


Figure 7: **Fixed point $\hat{x}$ of the system-wide fraction of faulty ids in local views, as a function of p, under a balanced attack.**

Observations:

O.1 $\psi(\tilde{x})$ is monotonically increasing for $\tilde{x} \in [0,1]$, since both $\frac{p}{p+(1-p)(1-\tilde{x})}$ and $\tilde{x}+(1-\tilde{x})\tilde{x} = 2\tilde{x}-\tilde{x}^2$ are monotonically increasing in this interval.

O.2 The absolute value of the first derivative of $\psi(\tilde{x})$ for $x \in [0,1]$ is bounded by a constant K.

O.3 $\lim_{x\to-\infty} h(x) = -\infty$, $h(0) = \alpha p + \gamma f \geq 0$, $h(\hat{x}) = 0$, $h(1) = p(\alpha+\beta+\gamma f - 1) \leq 0$, and $\lim_{x\to+\infty} h(x) = +\infty$. $h(x)$ has a single feasible root $0 < \hat{x} < 1$ (since $h(x)$ is continuous and the other two roots lie outside $[0,1]$). In addition, $h(x)$ is increasing in $(0,\hat{x})$ and decreasing in $(\hat{x},1)$.

O.4 $\psi(x) > x$ for $x \in (0,\hat{x})$ and $\psi(x) < x$ for $x \in (\hat{x},1)$. This is a straightforward application of the previous observation.

We focus on valid roots $0 \leq \hat{x} \leq 1$. A fixed point $\hat{x} = 1$ is called *trivial* (any other value is nontrivial). The existence of a nontrivial fixed point means that there is a stable system state in which the representation of correct ids is nonzero. On the other hand, if the system is at the trivial fixed point $\hat{x} = 1$, it means the views of all correct nodes hold only faulty ids.

Fixed points with history samples. If $\gamma > 0$ (i.e., history samples are used), a trivial fixed point does not exist (1 is not a root) and a single nontrivial fixed point always exists. This is since $h(0) \geq 0$ and $h(1) < 0$ and by Observation O.3 a single feasible root lies in $0 \leq \hat{x} < 1$.

Fixed points without history samples. If $\gamma = 0$ (no history samples), then $\hat{x} = 1$ is a root, i.e., a trivial fixed point exists. This is easily explainable, since if the views of all the correct nodes are totally poisoned, then neither pulls nor pushes can help. By Observation O.3 there is also a single feasible root $0 \leq \hat{x} < 1$. 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.

Two more parameter combinations deserve special interest:

1. $\beta = 1, \alpha = \gamma = 0$ (pull only, no history samples). Both roots $\hat{x} = 0$ and $\hat{x} = 1$ exist, for all p. This can be easily explained by considering the initial conditions. Since faulty nodes cannot push their own ids, if none of the views initially contain a faulty id, correct nodes pull only from correct

nodes and the faulty nodes will remain unrepresented. On the other hand, as shown in Figure 1(c,d), if $\tilde{x}(T_0) > 0$ (faulty nodes are initially represented) the views collapse to $\hat{x} = 1$.

2. $\alpha = 1, \beta = \gamma = 0$ (push only, no history samples). The only valid root is $\hat{x} = \frac{p}{1-p}$, for $p \leq \frac{1}{2}$ (recall that $p > 0$). That is, a nonzero fraction of correct ids can be maintained iff the majority of pushes are correct. This follows from the fact that a single correct push and a single faulty push equally contribute to the view.

These results highlight the importance of using history samples. Figure 7 depicts a fixed point of $\tilde{x}(t)$ for two combinations of α, β, and γ and for various values of p. We see a perfect match between theoretical analysis and simulations.

7.1.3 Convergence to the fixed point

We conclude the analysis by showing convergence to a nontrivial fixed point, if one exists.

The idea. We show (Appendix B.2) that the sequence of expected values of $\tilde{x}(t)$, $\{\mathrm{E}(\tilde{x}(T_0+k))\}$ for $k \geq 0$, can be approximated by an auxiliary sequence $\{\psi^k(\tilde{x}(T_0))\}$. The latter sequence converges to $\hat{x}$, i.e., so does the expected value of $\tilde{x}(t)$. We exploit the properties of ψ and use well-known calculus techniques.

7.2 Targeted Attack

We study a targeted attack on a single correct node u, which starts upon u's join, which occurs at time 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 the upper bound on the time to a perfect correct sample shown in Section 6 (a sufficient condition for non-isolation).

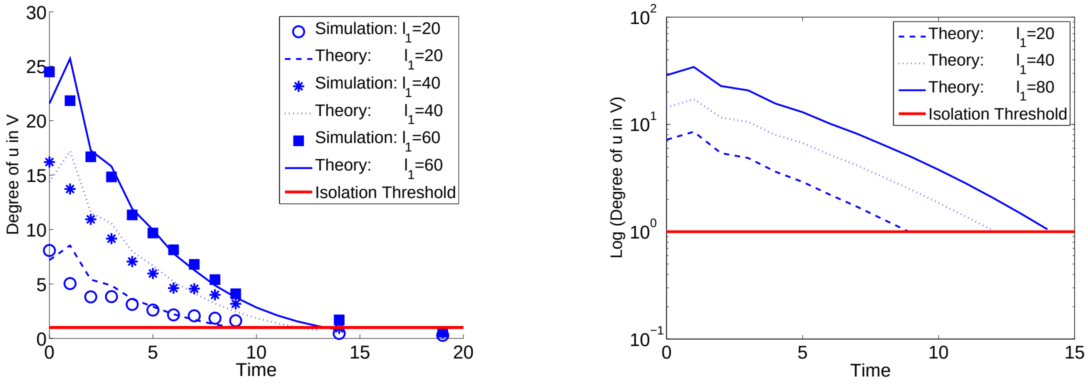
Lower bound on expected isolation time. As we seek a lower bound, we make a number of worst-case assumptions (formally stated in Appendix C). First, we assume that the protocol does not employ history samples (i.e., $\gamma = 0$), so that $\mathcal{S}$ does not correct $\mathcal{V}$'s bias. Next, we 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. At the same time 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. Hence, the ratio of faulty ids in this view is at the fixed point, i.e., higher than p (Section 7.1).

Clearly, the time to isolation in $\mathcal{V}(t)$ is a lower bound on that in $\mathcal{N}(t)$. We study the dynamics of u's degree in $\mathcal{V}(t)$, i.e., the sum of the out-degree (the number of correct ids in view), $\ell_1 - x_u(t)$, and the in-degree, $y_u(t)$. We show in Appendix C.2 that for every two 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

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



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

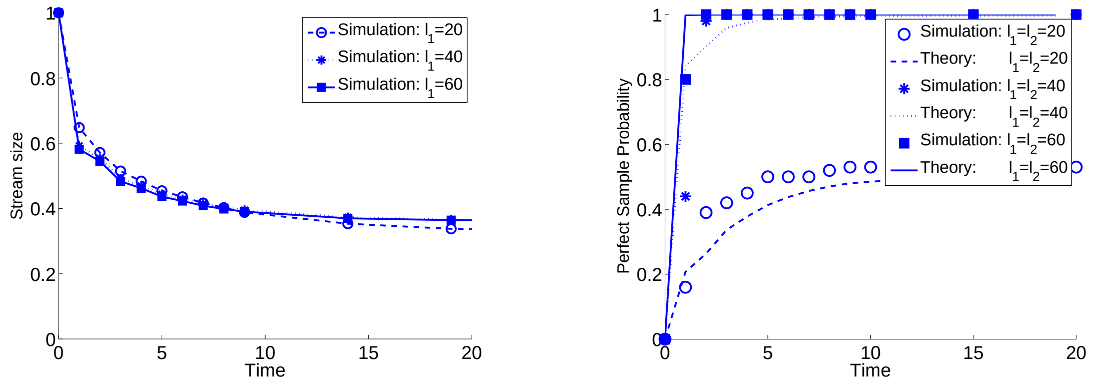
(b) Logarithmic scale (theory only), independent of n

Figure 8: **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**.**

Note that the coefficient matrix does not depend on $x_u(t)$ or $y_u(t)$, 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., $y_u(T_0) = 0$, but within a few rounds, u becomes represented and $\ell_1 - x_u$ and y_u are close). 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 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 poisoned the push-originated entries in $\mathcal{V}_u$. The decay rate would have been much larger, leading to almost immediate isolation.

Figure 8(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. When the degree becomes 1, the node is isolated. For example, the average time to isolation for $\ell_1 = 2\sqrt[3]{n}$ is 10 rounds. Figure 8(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 6), Lemma 6.6 bounds $PSP_u(t)$, the probability for a perfect sample at time t, from below. The expected number of correct ids observed by u till the end of round T is $\Lambda(t) = \sum_{t=T_0}^{T_0+T-1}(\mathrm{E}(g_u^{\text{push}}(t)) + \mathrm{E}(g_u^{\text{pull}}(t)))$; the expected values of $g_u^{\text{push}}(t)$ and $g_u^{\text{pull}}(t)$ are by-products of the analysis in Appendix C. Figure 9(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 9(b) depicts the progress of the upper bound of Lemma 6.6 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 8(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 ceases observing new correct ids. 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 (see 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



(a) Deficiency factor.

(b) Perfect sample probability, $\rho = 0.4$.

Figure 9: **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.**

sampled by other correct nodes, with high probability. These factors coupled with history samples ($\gamma > 0$) completely prevent u's isolation, as shown in Section 8.

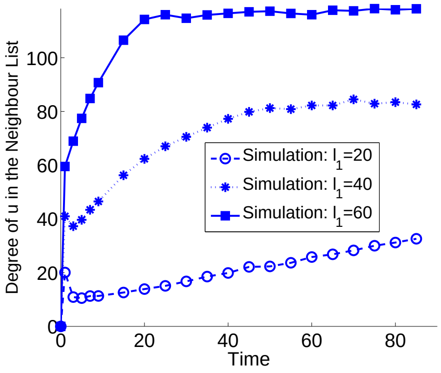
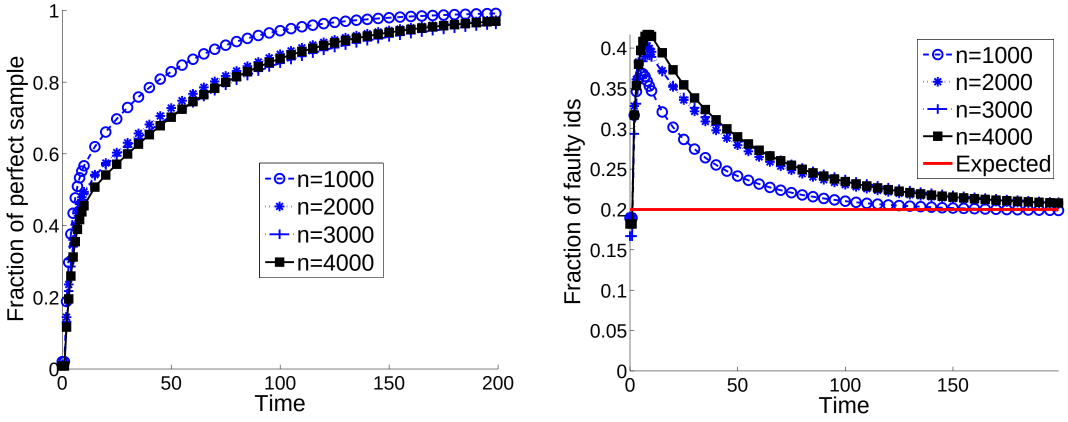


Figure 10: **Targeted attack: degree dynamics of an attacked node in** $\mathcal{N}(t)$**,** $n = 1000$**,** $p = 0.2$**,** $\alpha = \beta = 0.45$ **and** $\gamma = 0.1$**.**

8 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 7.2, 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 in Figure 11(a). Note that this criterion is



(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 $\ell_2 = 2\sqrt[3]{n}$.**

conservative, since missing a perfect sample does not automatically lead to a biased choice. More than 50% of perfect samples are achieved 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 $n \geq 2000$.

9 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 and Udi Wieder for their 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.

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] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In *Proc. of the 28th annual ACM symposium on Theory of computing (STOC)*, pages 20–29, 1996.

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

[4] B. Awerbuch and C. Scheideler. Group Spreading: A Protocol for Provably Secure Distributed Name Service. In *ICALP*, pages 183–195, 2004.

[5] B. Awerbuch and C. Scheideler. Robust Random Number Generation for Peer-to-Peer Systems. In *OPODIS*, pages 275–289, 2006.

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

[7] B. Awerbuch and C. Scheideler. Towards Scalable and Robust Overlay Networks. In *IPTPS*, 2006.

[8] B. Babcock, M. Datar, and R. Motwani. Sampling from a moving window over streaming data. In *Proc. of the 13th annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 633–634, 2002.

[9] 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.

[10] 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.

[11] Z. Bar-Yossef and M. Gurevich. Random Sampling from a Search Engine's Index. In *Proc. of 15th WWW*, pages 367–376, 2006. Full version available as CCIT Report #598, Department of Electrical Engineering, Technion.

[12] Z. Bar-Yossef, T. S. Jayram, R. Kumar, D. Sivakumar, and L. Trevisan. Counting distinct elements in a data stream. In *Proc. of the 6th International Workshop on Randomization and Approximation Techniques (RANDOM)*, pages 1–10, 2002.

[13] 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.

[14] 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.

[15] M. Castro, P. Druschel, A. J. Ganesh, A. I. T. Rowstron, and D. S. Wallach. Secure Routing for Structured Peer-to-Peer Overlay Networks. In *OSDI*, 2002.

[16] T. Condie, V. Kacholia, S. Sankararaman, J. Hellerstein, and P. Maniatis. Induced Churn as Shelter from Routing-Table Poisoning. In *Proc. of the 13th Annual Network and Distributed System Security Symposium (NDSS)*, 2006.

[17] M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining Stream Statistics over Sliding Windows. *SIAM J. Comput.*, 31(6):1794–1813, 2002.

[18] 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.

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

[20] J. R. Douceur. The Sybil Attack. In *Proc. of the 1st International Workshop on Peer-to-Peer Systems (IPTPS)*, pages 251–260, 2002.

[21] 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.

[22] 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.

[23] C. Gkantsidis, M. Mihail, and A. Saberi. Random walks in peer-to-peer networks. In *IEEE INFOCOM*, pages 130–140, 2004.

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

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

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

[27] G.-P. Jesi and M. van Steen D. Hales. Identifying Malicious Peers Before It's Too Late: A Decentralized Secure Peer Sampling Service. In *Proc. of the 1st IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, June 2007.

[28] 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.

[29] D. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, and R. Panigrahy. Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web. In *Proc. of the ACM STOC*, pages 654–663, 1997.

[30] D. R. Karger and M. Ruhl. Simple Efficient Load Balancing Algorithms for Peer-to-Peer Systems. In *SPAA*, pages 36–43, 2004.

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

[32] C. Law and K. Siu. Distributed construction of random expander networks. In *IEEE INFOCOM*, pages 2133– 2143, April 2003.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

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

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

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

[40] A. Singh, M. Castro, P. Druschel, and A. Rowstron. Defending against eclipse attacks on overlay networks. In *ACM SIGOPS European Workshop*, 2004.

[41] A. Singh, T.-W. Ngan, P. Druschel, and D. S. Wallach. Eclipse Attacks on Overlay Networks: Threats and Defenses. In *IEEE INFOCOM*, 2006.

[42] J. S. Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, 1985.

[43] 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.

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

A Analysis - sampling

A.1 Convergence to First Perfect Sample

Proposition 6.5 (restated) *Let s be a sampler. Then, for $|\mathcal{C}| \gg 1$ and for each $t > T_0$,*

$$\Pr(s(t) \neq s^*|s^* \in \mathcal{C}) = e^{-\frac{\rho\Lambda(t)}{|\mathcal{C}|}}.$$

Proof : A sampler outputs its perfect id s^* once that id occurs in the sampler's input stream. So the probability of $s(t) \neq s^*$ is the probability that s^* did not appear in the stream of during the rounds $T_0 \leq t' \leq t$. Recall that $\Lambda(t)$ is the number of correct ids observed by the sampler from time T_0 to t, and that a stream of length $\Lambda(t)$ includes as many random unique ids as a stream of length $\rho\Lambda(t)$ in which correct ids are independent and distributed uniformly at random. Let G denote a random correct id observed by the sampler, and note that for each $v \in \mathcal{C}$, $\Pr(G = v) = \frac{1}{|\mathcal{C}|}$. Then,

$$\begin{aligned}\Pr(s(t) \neq s^*|s^* \in \mathcal{C}) \quad &= \quad \Pr(G \neq s^*|s^* \in \mathcal{C})^{\rho\Lambda(t)} = \\ &\quad\; (1 - \Pr(G = s^*|s^* \in \mathcal{C}))^{\rho\Lambda(t)} = \\ &\quad\; \left(1 - \frac{1}{|\mathcal{C}|}\right)^{\rho\Lambda(t)}.\end{aligned}$$

Since $\frac{1}{|\mathcal{C}|} \ll 1$, we use $1 - x \approx e^{-x}$ ($1 - x$ is the first order Taylor expansion of e^{-x}, and is a good approximation for a small x), and approximate the above as follows:

$$\Pr(s(t) \neq s^*|s^* \in \mathcal{C}) \approx \left(e^{-\frac{1}{|\mathcal{C}|}}\right)^{\rho\Lambda(t)} = e^{-\frac{\rho\Lambda(t)}{|\mathcal{C}|}}.$$

From now on, we assume $\frac{1}{|\mathcal{C}|}$ is small enough, so we use equality. That is,

$$\Pr(s(t) \neq s^* | s^* \in \mathcal{C}) = e^{-\frac{\rho\Lambda(t)}{|\mathcal{C}|}}.$$

□

B Balanced Attack Analysis

B.1 Short-term Optimality

We now prove that in any single round, a balanced attack maximizes the expected system-wide fraction of faulty ids, $\tilde{x}(t)$, among all strategies. Consider a schedule $R : \mathcal{C} \rightarrow \mathbb{N}$ that assigns a number of faulty pushes to each correct node at round t. A schedule is *balanced* if for every two correct nodes u and v, it holds that $|R(u) - R(v)| \leq 1$. Otherwise, the schedule is *unbalanced*. We prove that every unbalanced schedule is suboptimal. All balanced schedules are equally optimal, for symmetry considerations.

Lemma B.1 *If schedule R is unbalanced, then there exists another schedule that leads to a larger expected ratio of faulty ids than R in round $t+1$.*

Proof : Since a schedule of faulty pushes in round t does not affect the pulls or history samples in this round, it is enough to prove the claim for the push-originated ids. Consider two nodes, u and v, such that $R(u) > R(v) + 1$. Consider an alternative schedule R' that differs from R in moving a single push from u to v. Consider the change in the expected cumulative fraction of push-originated faulty ids in $\mathcal{V}_u(t+1)$ and $\mathcal{V}_v(t+1)$ following this shift (in the other nodes, the ratio of faulty ids does not change).

The probability of a push-originated view entry at node u being faulty, provided that $R(u)$ faulty pushes were received, is equal to the expected fraction of $R(u)$ among all pushes received by u. Note that $R(u)$ is set in advance, i.e., without knowing the number of received correct pushes, $g_u^{\text{push}}(t)$. The expected number of faulty pushes accepted depends on the latter as follows:

$$\mathrm{E}(\tilde{r}_u^{\text{push}} | r_u^{\text{push}} = R(u)) = \sum_{G=1}^{|\mathcal{C}|} \Pr[g_u^{\text{push}}(t) = G] \cdot \frac{R(u)}{R(u)+G}.$$

We need to show that

$$\mathrm{E}(\tilde{r}_u^{\text{push}} | r_u^{\text{push}} = R(u) - 1) + \mathrm{E}(\tilde{r}_v^{\text{push}} | r_v^{\text{push}} = R(v) + 1) > \mathrm{E}(\tilde{r}_u^{\text{push}} | r_u^{\text{push}} = R(u)) + \mathrm{E}(\tilde{r}_v^{\text{push}} | r_v^{\text{push}} = R(v)),$$

i.e.,

$$\begin{aligned}
&\sum_{G=1}^{|\mathcal{C}|} \Pr[g_u^{\text{push}}(t) = G] \cdot \frac{R(u)-1}{R(u)-1+G} + \sum_{G=1}^{|\mathcal{C}|} \Pr[g_v^{\text{push}}(t) = G] \cdot \frac{R(v)+1}{R(v)+1+G} \\
&\geq \sum_{G=1}^{|\mathcal{C}|} \Pr[g_u^{\text{push}}(t) = G] \cdot \frac{R(u)}{R(u)+G} + \sum_{G=1}^{|\mathcal{C}|} \Pr[g_v^{\text{push}}(t) = G] \cdot \frac{R(v)}{R(v)+G}.
\end{aligned}$$

Since all correct nodes have the same in-degree in $\mathcal{V}(t)$ (Assumption 7.1), $g_u^{\text{push}}(t)$ and $g_v^{\text{push}}(t)$ have identical (binomial) distributions. Hence, it is enough to show that for all $G \geq 0$ and all $R(u) > R(v) + 1 > 0$:

$$\frac{R(u)-1}{R(u)-1+G} + \frac{R(v)+1}{R(v)+1+G} \geq \frac{R(u)}{R(u)+G} + \frac{R(v)}{R(v)+G}.$$

We simplify by switching sides:

$$\left(\frac{R(u)-1}{R(u)-1+G}-\frac{R(u)}{R(u)+G}\right)+\left(\frac{R(v)+1}{R(v)+1+G}-\frac{R(v)}{R(v)+G}\right)\geq 0.$$

$$\frac{-G}{(R(u)+G)(R(u)-1+G)}+\frac{G}{(R(v)+G)(R(v)+1+G)}\geq 0.$$

Since $R(u)-1\geq R(v)+1>0$ and $R(u)-2\geq R(v)>0$, indeed

$$\frac{-G}{(R(u)+G)(R(u)-1+G)}+\frac{G}{(R(v)+G)(R(v)+1+G)}$$

$$\geq\frac{-G}{(R(u)+G)(R(u)-1+G)}+\frac{G}{(R(u)-2+G)(R(u)-1+G)}$$

$$\geq\frac{G}{R(u)-1+G}\cdot\left(\frac{1}{R(u)-2+G}-\frac{1}{R(u)+G}\right)=\frac{G}{R(u)-1+G}\cdot\frac{2}{(R(u)+G)(R(u)-2+G)}>0.$$

As needed. □

We conclude by showing that all balanced schedules are equally optimal for the adversary.

Proposition B.2 *Every two balanced schedules lead to the same expected fraction of faulty ids in round* $t+1$.

Proof : Consider two balanced schedule R and R'. R can be transformed into R' by a sequence of moves of a single push message from node u to node v, such that $R(u)=R(v)+1$ whereas $R'(v)=R'(u)+1$. For symmetry reasons, neither of these moves alters the expected cumulative fraction of faulty ids received by u and v. Hence, each transformation produces a schedule that implies the same $\tilde{x}(t+1)$ as the previous one. □

B.2 Convergence to the fixed point

To capture the dynamics of $\tilde{x}(t)$, we define the sequence $\{a_k\}$, the expected system-wide fractions of faulty ids at time T_0+k, as follows:

$$a_k=\begin{cases}\tilde{x}(T_0) & k=0,\\ E(\tilde{x}(T_0+k))=E(\psi(\tilde{x}(T_0+k-1))) & k>0\end{cases}$$

We next define $\{b_k\}$, which we use to approximate $\{a_k\}$. $\{b_k\}$ is defined as follows:

$$b_k=\psi^k(\tilde{x}(T_0)),\forall k\geq 0.$$

Equivalently, $b_k=\psi^k(b_{k-1})$. That is, $\{b_k\}$ is a sequence of applying ψ on the expected system-wide fractions of faulty ids in every cycle.

In order to prove convergence of $\{b_k\}$, we define an auxiliary sequence $\{c_k\}$ below. We prove that $\{b_k\}$ is bounded between $\hat{x}$ and $\{c_k\}$. Finally, we show that the latter sequence converges to $\hat{x}$, implying that so does $\{b_k\}$. Since $\{b_k\}$ approximates $\{a_k\}$, $\{a_k\}$ converges to $\hat{x}$ as well.

We now explain why $\{b_k\}$ can be used to approximate $\{a_k\}$. Consider an element a_k of $\{a_k\}$. Since a_k is the expectation of a random variable (namely $\tilde{x}(T_0+k)$), it can be written as $a_k=\sum p_i x_i$, where $\forall i: p_i=\Pr[\tilde{x}(T_0+k)=x_i]$. By Lemma 7.2, a_{k+1} can be written as $a_{k+1}=\sum p_i\psi(x_i)$.

Since $\tilde{x}$ is obtained as a combination of binomial distributions with many trials (we assume n to be very large), it has a small variance, and therefore all the significant contributors to this sum are very close to each other, i.e., they all lie within a small segment. Moreover, since ψ is continuous, monotonic, and has a bounded derivative in $(0, 1)$, in small segments, it can be approximated by a linear function. Therefore, $a_{k+1} = \sum p_i\psi(x_i) \approx \psi(\sum p_i x_i) = \psi(a_k)$.

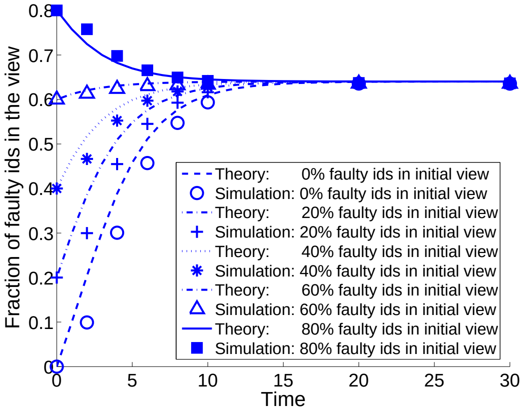


Figure 12: **System-wide fraction of faulty ids in local views, under a balanced attack. Convergence to a fixed point $\hat{x}$: $n = 1000$, $p = 0.2$, $\alpha = \beta = 0.5$ and $\gamma = 0$. The theory depicts the sequence $\{b_k\}$.**

Figure 12 depicts the evolution of $\{b_k\}$ as a function of time for various initial values of $\tilde{x}(T_0)$. The figure also depicts the actual ratio of the faulty ids in the views in the simulation study. We can see that $\{b_k\}$ well approximates the actual faulty ids fraction. $\{b_k\}$'s convergence is slightly faster because the analysis ignores blocking.

We next prove that $\{c_k\}$ converges to $\hat{x}$. This is done by applying Hillam's theorem [25]. We conclude by showing that $\{b_k\}$ is bounded between $\hat{x}$ and $\{c_k\}$, thus proving the convergence of $\{b_k\}$.

Preliminary B.3 *Liphschitz condition (simplified) [25]:*
The function $f : [a, b] \to [a, b]$ satisfies the Liphschitz condition with constant K iff for all $x, y \in [a, b]$ it holds that $|f(x) - f(y)| \leq K|x - y|$.

Preliminary B.4 *Hillam's theorem [25]:*
If $f : [a, b] \to [a, b]$ satisfies the Liphschitz condition with constant K, then the iteration scheme $\{x_{n+1} = \lambda x_n + (1 - \lambda)f(x_n)\}$, where $\lambda = \frac{1}{K+1}$, converges to a fixed point of f.

Lemma B.5 *The sequence $\{c_k\}$ converges to the fixed point of $\psi(\tilde{x})$.*

Proof : Based on Observations O.1 and O.2 and by the mean value theorem, for all $\tilde{x}_1, \tilde{x}_2 \in [0, 1]$ ($\tilde{x}_1 \leq \tilde{x}_2$), there exists $\tilde{x}' \in [\tilde{x}_1, \tilde{x}_2]$ such that

$$\psi(\tilde{x}_2) - \psi(\tilde{x}_1) = \frac{\delta\psi}{\delta x}(x') \cdot (\tilde{x}_2 - \tilde{x}_1).$$

We can therefore find a constant K satisfying the Lipschitz condition for ψ in $[0, 1]$. Let K be such a constant, and let $\lambda = \frac{1}{K+1}$. We are now ready to define $\{c_k\}$, as follows:

$$c_k = \begin{cases} \tilde{x}(T_0) & k = 0, \\ \lambda c_{k-1} + (1 - \lambda)\psi(c_{k-1}) & k > 0 \end{cases}$$

Therefore, by Hillam's theorem (Preliminary B.4), the iteration scheme $\{c_k = \lambda c_{k-1} + (1-\lambda)\psi(c_{k-1})\}$, where $\lambda = \frac{1}{K+1}$, converges to a fixed point of $\psi(x)$ for each $c_0 = \tilde{x}(T_0) \in [0, 1]$.

□

From now on, we separate the proof into 2 cases:

1. $\hat{x} \leq \tilde{x}(T_0) = a_0 = b_0 = c_0 < 1$.
2. $0 \leq \tilde{x}(T_0) = a_0 = b_0 = c_0 < \hat{x}$.

Lemma B.6 *If $\tilde{x}(T_0) < 1$, then $\{c_k\}$ converges to $\hat{x}$ (and not to $\hat{x} = 1$).*

Proof : For the first case, recall that $\hat{x}$ is a single nontrivial fixed point. By Observation O.4, $\psi(x) < x$ for $x \in (\hat{x}, 1)$. For an arbitrary $x \in (\hat{x}, 1)$, it holds that $\lambda x + (1 - \lambda)\psi(x) < x$, i.e, the sequence $\{c_k\}$ is monotonically decreasing with t. Hence, this sequence cannot converge to the trivial fixed point (if one exists), i.e., it converges to $\hat{x}$. The proof for the second case is symmetrical. □

Lemma B.7 *$\{b_k\}$ is bounded between $\hat{x}$ and $\{c_k\}$.*

Proof : For the first part of the claim we need to prove that $\hat{x} \leq b_k \leq c_k$ (the second part's proof is symmetrical). We prove by induction on t. The basis is immediate by definition of b_0 and c_0. Assume that $\hat{x} \leq b_k \leq c_k$ for $k > 0$. Consider the following statements:

1. $\psi(c_k) < c_{k+1}$. We know that $c_{k+1} = \lambda c_k + (1-\lambda)\psi(c_k) > \psi(c_k)$ since $\psi(c_k) < c_k$ (by Observation O.4, $\psi(x) < x$ for $x \in (\hat{x}, 1)$ and indeed $c_k \in (\hat{x}, 1)$).
2. $\psi(b_k) \leq \psi(c_k)$, since ψ is monotonically increasing for $x \in [0, 1]$ (Observation O.1) and based on the induction hypothesis ($b_k \leq c_k$).
3. $\psi(b_k) = b_{k+1}$ by definition of b_{k+1}.
4. $\psi(\hat{x}) \leq \psi(b_k)$, since ψ is monotonically increasing for $x \in [0, 1]$ (Observation O.1) and based on the induction hypothesis ($\hat{x} \leq b_k$).
5. $\hat{x} = \psi(\hat{x})$ by definition of $\hat{x}$.

Combining the above statements we get $\hat{x} = \psi(\hat{x}) \leq b_{k+1} = \psi(b_k) \leq \psi(c_k) < c_{k+1}$, thus concluding the induction step.

□

Since the balanced attack does not distinguish between correct nodes, the same result holds for $\tilde{x}_u(t)$, for each correct node u.

C Targeted Attack Analysis

This section analyzes the dynamics of a targeted attack on a single correct node.

C.1 Assumptions

We use the following assumptions on the environment in order to bound the time to isolation from below.

Assumption C.1 *(no history samples) $\gamma = 0$, which is equivalent to the worst-case assumption that the expected ratio of faulty ids in $\mathcal{S}$ at all times is equal to that in the id stream observed by the node (i.e., history samples are ineffective).*

Assumption C.2 *(unrealistically strong adversary) In each round $t \geq T_0$, the adversary observes the exact number of correct pushes received by u, $g_u^{\text{push}}(t)$, and complements it with faulty pushes to $\alpha\ell_1$ (i.e., the maximal number of faulty ids that can be accepted without blocking). Formally, $r_u^{\text{push}}(t) \triangleq \max(\alpha\ell_1 - g_u^{\text{push}}(t), 0)$.*

Assumption C.3 *(background attack on the rest of the system) The adversary maximizes its global representation through a balanced attack on all correct nodes $v \neq u$. At time T_0, the system-wide expected fraction of faulty ids is at the fixed point $\hat{x}$. (Note that this attack minimizes the fraction of correct ids that u can pull from correct nodes).*

Assumption C.4 *(fresh attacked node) u joins the system at T_0. It is initially not represented in any correct node's view and u's initial view is taken from a random correct node.*

We assume that the effect of u on the entire system's dynamics is negligible. Hence, we assume that the out-degrees and the in-degrees of all correct nodes except u are equal at all times (Assumption 7.1), and these nodes do not block (Section 7.1 showed that the system-wide effect of blocking is marginal).

C.2 Node Degree Dynamics

We study the dynamics of the degree of the attacked node u $\mathcal{V}(t)$. Consider a set of triples $\{(X, Y, t)\}$, each standing for a state $\{x_u(t) = X \ \wedge \ y_u(t) = Y\}$, for $X \in \{0, \ldots, \ell_1\}, Y \in \{0, \ldots, |\mathcal{C}|\ell_1\}$. Each t defines a probability space, i.e., $\sum_{X,Y} \Pr[(X, Y, t)] = 1$. Since u is initially not represented, the only states that have non-zero probability for $t = T_0$ are those for which $Y = 0$. The probability distribution over these states is identical to the distribution of $x_u(T_0)$. Since u borrows its initial view from a random collection of correct nodes, $x_u(T_0) \sim$ Bin$(\ell_1, \hat{x})$.

We now develop probability spaces for each $t > T_0$. The notation $\Pr[(X', Y', t+1)|(X, Y, t)]$ stands for the probability of transition from state (X, Y, t) to state (X', Y', t). That is, $\Pr[(X', Y', t+1)] = \sum_{X,Y} \Pr[(X', Y', t+1)|(X, Y, t)] \cdot \Pr[(X, Y, t)]$. To analyze $\Pr[(X', Y', t+1)|(X, Y, t)]$ we separately consider four independent random variables: the number of push- and pull-originated entries in $\mathcal{V}_u$, (denoted $x_u^{\texttt{push}}(t)$ and $x_u^{\texttt{pull}}(t)$), and the number of push- and pull-propagated instances of u in the views of correct nodes (denoted $y_u^{\texttt{push}}(t)$ and $y_u^{\texttt{pull}}(t)$). The first two affect X' whereas the last two affect Y'. We now demonstrate how conditional probability distributions for these variables are computed. For convenience, we omit the conditioning on (X, Y, t) from further notation.

$y_u^{\texttt{pull}}(t)$: Since the system is at the fixed point, the probability of pulling from some other correct node is $(1 - \hat{x})$. Hence, $y_u^{\texttt{pull}}(t+1)$ is a binomially distributed variable, with the number of trials equal to the total number of correct pulls, $(1 - \hat{x})\beta\ell_1|\mathcal{C}|$, and the probability of success equal to the chance of an entry in a random node's view being u, namely $\frac{Y}{\ell_1|\mathcal{C}|}$: $y_u^{\texttt{pull}}(t+1) \sim$ Bin$((1 - \hat{x})\beta\ell_1|\mathcal{C}|, \frac{Y}{\ell_1|\mathcal{C}|})$. Note that $\mathrm{E}(y_u^{\texttt{pull}}(t+1)) = \beta(1 - \hat{x})Y$.

$y_u^{\texttt{push}}(t)$: By Lemma 7.2, the number of pushes that reach correct nodes is $\alpha\ell_1|\mathcal{C}|\frac{(1-\hat{x})(1-p)+p}{1-p}$. Denote the number of pushes from u to correct nodes in round t by $z_u(t)$. This is a binomially distributed variable with $\alpha\ell_1$ trials and probability of success equal to $1-\frac{X}{\ell_1}$: $z_u(t) \sim \mathrm{Bin}(\alpha\ell_1, 1-\frac{X}{\ell_1})$. For a given $z_u(t) = Z$, since the total number of push-originated entries is $\alpha\ell_1|\mathcal{C}|$, the number of push-propagated instances of u is $y_u^{\texttt{push}}(t+1|Z) \sim \mathrm{Bin}\left(\alpha\ell_1|\mathcal{C}|, \frac{Z}{\alpha\ell_1|\mathcal{C}|((1-\hat{x})+\frac{p}{1-p})}\right)$. Note that $\mathrm{E}(y_u^{\texttt{push}}(t+1|Z)) = Z\frac{1-p}{p+(1-p)(1-\hat{x})}$. Hence, since Z is independent on p and $\hat{x}$,

$$\mathrm{E}(y_u^{\texttt{push}}(t+1)) = \mathrm{E}(Z)\frac{1-p}{p+(1-p)(1-\hat{x})} = \alpha(\ell_1 - X)\cdot\frac{1-p}{p+(1-p)(1-\hat{x})}.$$

$x_u^{\texttt{pull}}(t)$: A pull from a faulty node (which happens with probability $\frac{X}{\ell_1}$) produces a faulty id with probability 1, otherwise the probability to receive a faulty id is $\hat{x}$. Hence, the probability of pulling a faulty id is $\frac{X}{\ell_1} + (1-\frac{X}{\ell_1})\hat{x}$. That is, the number of pull-originated faulty ids in u's view is $x_u^{\texttt{pull}}(t+1) \sim \mathrm{Bin}(\beta\ell_1, \frac{X}{\ell_1} + (1-\frac{X}{\ell_1})\hat{x})$ (i.e., $\mathrm{E}(x_u^{\texttt{pull}}(t+1)) = \beta(X + (\ell_1 - X)\hat{x})$).

We also compute the expected number of correct ids (with duplicates) pulled by u, which we need for estimating the size of the id stream that traverses this node (Section 7.2). Since u performs $\beta\ell_1$ pulls, and the expected number of correct ids pulled from a random node is $(1-\hat{x})\ell_1$,

$$\mathrm{E}(g_u^{\mathrm{pull}}(t)) = (1-\frac{X}{\ell_1})\cdot\beta\ell_1\cdot(1-\hat{x})\ell_1 = (1-\hat{x})\ell_1(\ell_1 - X).$$

$x_u^{\texttt{push}}(t)$: The number of push-originated ids, $x_u^{\texttt{push}}(t+1)$, depends on the number of correct pushes received by u, $g_u^{\mathrm{push}}(t)$. The latter is a binomially distributed variable, with the number of trials equal to the total number of correct pushes, $\alpha\ell_1|\mathcal{C}|$, and the probability of success equal to the chance of an entry in a random node's view being u, namely $\frac{Y}{\ell_1|\mathcal{C}|}$: $g_u^{\mathrm{push}}(t) \sim \mathrm{Bin}(\alpha\ell_1|\mathcal{C}|, \frac{Y}{\ell_1|\mathcal{C}|})$ (Note that $\mathrm{E}(g_u^{\mathrm{push}}(t)) = \alpha Y$. This value is of independent use for evaluating the size of the id stream that traverses u (Section 7.2)).

An expected representation of a correct node different from u in the system is $(1-\hat{x})\ell_1$. Since u is under-represented ($Y < (1-\hat{x})\ell_1$ with high probability), the probability of receiving above $\alpha\ell_1$ correct pushes is low, and hence, we ignore the case of u being blocked by exceedingly many correct pushes. On the other hand, faulty pushes cannot block u either (Assumption C.2), and therefore, we assume that u never blocks. If $G \le \alpha\ell_1$ correct pushes are received, the adversary complements the number of pushes to the maximum allowed (Assumption C.2), i.e., the fraction of faulty pushes to u is $1-\frac{G}{\alpha\ell_1}$. Hence, the number of push-originated faulty ids in u's view is $x_u^{\texttt{push}}(t+1|G) \sim \mathrm{Bin}(\alpha\ell_1, 1-\frac{G}{\alpha\ell_1})$. In other words,

$$\mathrm{E}(x_u^{\texttt{push}}(t+1)) = \alpha\ell_1(1-\frac{\mathrm{E}(g_u^{\mathrm{push}}(t))}{\alpha\ell_1}) = \alpha\ell_1(1-\frac{\alpha Y}{\alpha\ell_1}) = \alpha(\ell_1 - Y).$$

Putting it all together. Summing up, the expected values of in-degree and out-degree can be written as

$$\begin{pmatrix}\ell_1 - \mathrm{E}(x_u(t+1)) \\ \mathrm{E}(y_u(t+1))\end{pmatrix} = \begin{pmatrix}\ell_1 - (\mathrm{E}(x_u^{\texttt{push}}(t+1)) + \mathrm{E}(x_u^{\texttt{pull}}(t+1))) \\ \mathrm{E}(y_u^{\texttt{push}}(t+1)) + \mathrm{E}(y_u^{\texttt{pull}}(t+1))\end{pmatrix} =$$

$$= \begin{pmatrix}\ell_1 - (\alpha(\ell_1 - Y) + \beta(X + (\ell_1 - X)\hat{x})) \\ \alpha(\ell_1 - X)\frac{1-p}{p+(1-p)(1-\hat{x})} + \beta(1-\hat{x})\end{pmatrix} =$$

$$= \begin{pmatrix}\beta(1-\hat{x}) & \alpha \\ \alpha\frac{1-p}{p+(1-p)(1-\hat{x})} & \beta(1-\hat{x})\end{pmatrix}\cdot\begin{pmatrix}\ell_1 - x_u(t) \\ y_u(t)\end{pmatrix}$$

Since we have shown that u does not block with high probability, and Section 7.1 demonstrated that the effect of blocking on the rest of correct nodes is negligible, we assume that all views are recomputed in each round. That is,

$$\Pr[x_u(t+1) = X' | (X,Y,t)] = \sum_{X'_1 + X'_2 = X'} \Pr[x_u^{\texttt{push}}(t) = X'_1 | (X,Y,t)] \cdot \Pr[x_u^{\texttt{pull}}(t) = X'_2 | (X,Y,t)],$$

and

$$\Pr[y_u(t+1) = Y' | (X,Y,t)] = \sum_{Y'_1 + Y'_2 = Y'} \Pr[y_u^{\texttt{push}}(t) = Y'_1 | (X,Y,t)] \cdot \Pr[y_u^{\texttt{pull}}(t) = Y'_2 | (X,Y,t)].$$

Since the computations of X' and Y' are independent, we conclude:

$$\Pr[(X',Y',t) | (X,Y,t)] = \Pr[x_u(t+1) = X' | (X,Y,t)] \cdot \Pr[y_u(t+1) = Y' | (X,Y,t)].$$