Towards Scalable and Robust Overlay Networks

Baruch Awerbuch*
Department of Computer Science
Johns Hopkins University
Baltimore, MD 21218, USA
baruch@cs.jhu.edu

Christian Scheideler
Department of Computer Science
Technical University of Munich
85748 Garching, Germany
scheideler@in.tum.de

ABSTRACT

Every peer-to-peer system is based on some overlay network connecting its peers. Many of the overlay network concepts proposed in the scientific community are based on the concept of virtual space. These designs are usually highly scalable, but they do not guarantee robustness against adversarial attacks, especially when considering open peer-to-peer systems. In these systems, determined adversaries may start both insider and outsider attacks in order to harm the overlay network as much as this is possible. We will focus on insider attacks in which the adversarial peers in the network perform join-leave attacks, and we will consider outsider attacks in which an adversary can perform a denial-of-service attack against any of the honest peers in the network. Strategies have been proposed that can defend an overlay network against even massive join-leave attacks, and strategies are also known that can defend an overlay network against limited denial-of-service attacks. However, none of these can protect an overlay network against *combinations* of these attacks. We illustrate in this paper why it is not easy to design strategies against these attacks and propose join and leave protocols for overlay networks based on the concept of virtual space that can make them provably robust against these attacks without creating too much overhead.

1. INTRODUCTION

Due to the rise of peer-to-peer systems, dynamic overlay networks have recently received a lot of attention. An overlay network is a logical network formed by its participants across a wired or wireless domain. In open peer-to-peer systems, participants may frequently enter and leave the overlay network. Hence, two operations have to be provided so that the overlay network can be adjusted to these changes:

- Join(v): peer v joins the system
- Leave(v): peer v leaves the system

A central goal has been to find join and leave operations that run as efficiently as possible and that maintain a highly scalable overlay network. However, besides scalability, robustness is also important since in open environments like the Internet adversaries may try to start both insider and outsider attacks on a distributed system.

In this paper, we study the problem of how to protect an overlay network against a certain combination of insider and outsider attacks. The insider attacks we will be focusing on are legal join-leave attacks on the system by adversarial peers. More specifically, we consider the scenario in which there are n honest peers and ϵn adversarial peers in the system for some constant $\epsilon < 1$. The adversary has full control over its adversarial peers and knows the entire overlay network at any point in time. It can use this information to decide in an adaptive manner which of its adversarial peers should leave the system and join it again from scratch. In this way, it can design a sequence of rejoin activities by the adversarial peers in order to harm the overlay network as much as this is possible. (For example, by degrading its scalability or isolating honest peers.)

Besides insider attacks, we also allow certain outsider attacks. We consider outsider attacks in which the adversary can shut down any peer at any point in time by starting a brute-force denial-of-service attack on it that bypasses the overlay network. We assume that an honest peer that is exposed to such an attack will leave the system (in a potentially non-graceful manner) and will rejoin the network from scratch as soon as the denial-of-service attack on it is over.

Our goal is to design *oblivious* join and leave operations that can protect an overlay network against *any* combination of insider and outsider attacks within our model, very high probability. (In the following, "with high probability", or short "w.h.p.", always means a probability of at least $1 - 1/n^c$, where n is the number of peers in the system and the constant c can be made arbitrarily large.) That is, the join and leave operations must not distinguish between the honest and adversarial peers and, even more, must not maintain any kind of state in order to protect the overlay network against these attacks. We will demonstrate that, on a high level, suitable operations indeed exist, and our hope is that they are sufficiently light-weight so that they are useful in practice.

In the following, let n be the number of honest peers in the system and ϵn for some $\epsilon < 1$ be the maximum number of adversarial peers in the system at any time. For simplicity, we are focusing on peers being placed at points in the $[0, 1)$-interval which is, for example, the case in Chord [20], but our results can also be adapted to other spaces, such as the $[0, 1)^2$ space used for the dynamic Gabber-Galil graphs [11] or the $[0, 1)^d$ space used by CAN [13]. We are considering the following game between an adversary and the system.

1.1 Join-leave game

The join-leave game proceeds in rounds. In each round, the adversary has complete knowledge of the entire system and can ask whatever peer it likes to leave and join the system again from scratch (which we will also call a rejoin request in the following). Rejoin requests of adversarial peers cover insider attacks and rejoin requests of honest peers cover outsider attacks (the adversary performs a DoS-attack on the honest peer so that it is excluded from the network, but it rejoins as soon as the attack has ended).

*Supported by NSF-ANIR 0240551, NSF-CCF 0515080, and NSF-CCR 0311795.

Our goal is to find *oblivious* join and leave strategies, i.e., strategies that have no memory of the history of the system and that cannot distinguish between the honest and adversarial peers, that assign positions in $[0,1)$ to the peers so that for *any* adversarial strategy above the following two conditions can be preserved for every interval $I \subseteq [0,1)$ of size at least $(c \log n)/n$ for a constant $c > 0$ and any polynomial number of rounds in n, with high probability:

- *Balancing condition:* I contains $\Theta(|I| \cdot n)$ peers.
- *Majority condition:* the honest peers in I are in the majority.

It is not difficult to show that if these conditions are kept, structured overlay network concepts together with quorum-based decision rules can be used to wash out adversarial behavior (e.g., [4]). Certainly, the brute-force strategy of giving every peer a new random place whenever a peer rejoins will achieve the stated goal, with high probability, but this would be a very expensive strategy. The challenge is to find join and leave operations that need as little randomness and as few rearrangements as possible to satisfy the two conditions. (Notice that generating random bits in a system with adversarial peers (see, e.g., [3]) as well as rearranging peers is expensive and should therefore be kept at a minimum.) It turns out that this is not easy. We first review some prior work (Section 1.2). Then we give a list of approaches that do not work (Section 1.3), which is followed by two approaches for which we do not know yet whether they work (Section 1.4), and finally we present an approach that does work (Section 1.5). This approach will be analyzed in Section 2.

1.2 Prior work on robust overlay networks

Solutions against the insider and outsider attacks covered in this paper have already been found if just *one* kind of attack is allowed. We will give a quick review of the literature below. However, *none* of these strategies work if the adversary can use any *combination* of these attacks. Hence, a new strategy is needed.

Outsider attacks

Oblivious outsider attacks on peer-to-peer systems in which peers are placed at random positions in some virtual space can be modeled as random faults or churn. Random faults and churn has been heavily investigated in the peer-to-peer community (e.g., [1, 9, 14, 15, 20]), and it is known that certain structured peer-to-peer systems like Chord can tolerate any constant fault probability. Much more difficult to handle are adaptive outsider attacks, and the best current result is a structured overlay network that can recover from any sequence of adaptive outsider attacks in which at most $\log n$ many peers can be removed from the system at any point in time [9]. The basic idea behind this approach is that the peers perform local load balancing in order to fill any holes the adversary may have created in the network. This approach works well if all peers in the network are assumed to be honest, but it does not work any more if some of the peers are adversarial. All the adversary would have to do in this case is to focus on a particular corner of the network and force all honest peers in it to leave until sufficiently many adversarial peers have accumulated in it. Once the adversarial peers have gained the majority in this corner, it is not hard to imagine that they can start serious application-layer attacks since it will not be possible any more to wash out adversarial behavior efficiently in a proactive manner.

Insider attacks

There are also some results on join-leave attacks by adversarial peers. People in the peer-to-peer community have been aware of the danger of these attacks since quite a while [7, 8] and various solutions have been proposed that may help thwart these attacks in practice [5, 6, 12, 17, 18, 19] but until recently no mechanism was known that can *provably* cope with these attacks without sacrificing the openness of the system.

The first mechanism that was shown to preserve randomness for a polynomial number of adversarial rejoin requests uses *random* peer IDs and enforces a *limited* lifetime on every peer in the system, i.e., every peer *has* to reinject itself after a certain amount of time steps [2]. However, this leaves the system in a hyperactive mode that may unnecessarily consume resources that could be better used for other purposes. Ideally, one would like to use *competitive* strategies. That is, the resources consumed by the mixing mechanism should scale with the join-leave activity of the system. Such a strategy was first presented for a pebble-shuffling game on a ring [16]. However, the join rule proposed there cannot be directly applied to overlay networks based on the concept of virtual space since it has no control over the distribution of the peers in the virtual space, i.e., the balancing condition can be violated.

The first rule that was able to satisfy the balancing and majority conditions for a polynomial number of rejoin requests of adversarial peers is the cuckoo rule [4]. We first introduce some notation, and then we describe this rule.

In the following, a *region* is an interval of size $1/2^r$ in $[0,1)$ for some positive integer r that starts at an integer multiple of $1/2^r$. Hence, there are exactly 2^r regions of size $1/2^r$. A *k-region* is a region of size (closest from above to) k/n, and for any point $x \in [0,1)$, the k-region $R_k(x)$ is the unique k-region containing x. Whenever a peer leaves, it just leaves, but when it joins, the following rule is used:

Cuckoo rule: If a new peer v wants to join the system, pick a random $x \in [0,1)$. Place v into x and move all peers in $R_k(x)$ to points in $[0,1)$ chosen uniformly and independently at random (without replacing any further peers).

It was shown [4] that this rule works for arbitrary join-leave attacks of adversarial peers as long as $\epsilon < 1 - 1/k$, and this bound it tight. However, if also honest peers can be forced to leave, it does not work any more. To see why, focus on a region R of size $(c \log n)/n$ in $[0,1)$. Ask any peer still in R to leave until there are no peers left in R. Even if these peers will immediately rejoin afterwards, the probability that an application of the cuckoo rule will move at least one peer back to R is just $O(|R|)$. Hence, on expectation, within $O(\log n)$ departures R will lose all of its peers, which violates the balancing condition.

1.3 Join and leave operations that do not work

Is it possible to find a proper modification of the cuckoo rule so that the class of rejoin attacks considered in this paper can be handled? We will show that several simple variants of the cuckoo rule do not work.

Cuckoo rule with random peer transposition

We have seen that just using the cuckoo rule alone is not sufficient. So how about filling the position given up by the departing peer by a random peer in the system? This does not work either due to the following attack. Again, focus on a region R of size $(c \log n)/n$ in $[0, 1)$. Ask any honest peer in R to leave until there are no more honest peers in R. Since each position of a departing honest peer is filled with an adversarial peer with constant (or more precisely, $\epsilon/(1+\epsilon)$) probability, it just takes a constant number of departures, on average, to turn a position occupied by an honest peer into a position occupied by an adversarial peer. Since it is quite unlikely that R will be affected by $O(\log n)$ applications of the cuckoo rule, it follows that, on expectation, within $O(\log n)$ departures R will only have adversarial peers in it, which violates the majority condition.

Cuckoo rule with random transposition of k-region

Suppose that whenever a peer departs some k-region R, R is flipped with some other k-region R' chosen uniformly at random in $[0, 1)$, that is, R is moved with all of its peers to R' and R' is moved with all of its peers to R. Also for this rule the majority condition can easily be violated.

Suppose that for n many rejoin operations, the adversary picks any adversarial peer p in the system that has at least one other (adversarial or honest) peer in its k-region to rejoin the system (if there is no such peer, we would have ϵn many k-regions with just one peer, and this peer is adversarial, which would be perfect for our attack to work). According to the cuckoo rule, p is moved to a random k-region, and all peers previously in it are replaced. Hence, initially, p is alone in this k-region. The probability that none of r further rejoin operations of the adversary will affect p's k-region is approximately $(1-k/n)^{(1+\epsilon)k \cdot r} \approx e^{-(1+\epsilon)k^2 r/n}$ since, on expectation, $(1+\epsilon)k$ peers are replaced in each rejoin operation and the probability for any of them not be thrown into p's k-region is $1 - k/n$. If k is a constant then it takes $\Theta(n)$ applications of the rejoin operation, on expectation, until p's k-region will be affected by it. Hence, after n many rejoin operations there will be a constant fraction of k-regions that just have a single peer, which is adversarial.

Now, the adversary focuses on a particular region $\hat{R}$ of size $(c \log n)/n$ and asks any honest peer to leave in it until there are only adversarial peers left in $\hat{R}$. It takes $O(1)$ departures of a peer in some k-region of $\hat{R}$, on expectation, until this is flipped with a k-region just containing a single, adversarial peer. Hence, it takes at most $O(\log n)$ departures of peers in $\hat{R}$, on expectation, until every k-region in $\hat{R}$ has been flipped with a k-region just containing a single, adversarial peer. Since it is unlikely that any of these k-regions will be affected by the $O(\log n)$ applications of the cuckoo rule when the departed peers rejoin, $\hat{R}$ will only have adversarial peers within $O(\log n)$ rejoin operations, on expectation, violating the majority rule. Though we only focused on a constant k, the attack also works for any non-constant $k = O(\log n)$.

1.4 Join and leave operations that might or might not work

Next we present two strategies for which we do not know yet whether they would work or not.

Cuckoo rule with random transposition of neighboring k-region

Suppose that whenever a peer p leaves the system, a k-region R will be picked uniformly at random in the $(c \log n)$-region $\hat{R}$ containing p and R will be flipped with another k-region chosen uniformly at random in $[0, 1)$. Afterwards, p rejoins the system using the cuckoo rule.

The intuition behind choosing a random neighboring region is that this makes it hard for an attacker to focus on a particular k-region without affecting other k-regions so that none of the attacks above works any more. Interestingly, when deviating from the rule above of choosing a neighboring k-region *uniformly* at random, attacks are again possible. For example, if in a $(c \log n)$-region R a larger probability is given to flipping an underloaded or overloaded k-region in it, then the adversary can start a subtle attack that can cause the violation of the balancing rule, even though the rule looks advantageous for R.

Let us illustrate the attack for the extreme case that always the most overloaded k-region is picked in a $(c \log n)$-region, if there are (significantly) overloaded k-regions in this region. Suppose first that we pick some peer p and perform $\Theta(n \log n)$ many rejoin operations with it. In this way, we create at least $\sqrt{n}$ overloaded k-regions with $\Theta(\log n)$ many peers each, w.h.p., which is not difficult to show. Next, we focus on a fixed $(c \log n)$-region R. We always pick a $(c \log n)$-region R' different from R that contains at least one k-region R'' with $\Theta(\log n)$ peers. It is not too difficult to show that such a k-region exists for the number of operations we need the adversarial strategy to run, w.h.p. Remove a peer in R'. The probability that R'' will be moved into R in this case is $(c \log n)/n$. Hence, within $\Theta(n)$ many rejoin requests it is quite likely that $\Theta(\log n)$ many k-regions with $\Theta(\log n)$ peers each will be moved into R. With constant probability, only a constant fraction of these k-regions will be moved out of R within $\Theta(n)$ many rejoin operations resulting in $\Theta(\log^2 n)$ peers in R at the end. This, however, violates the balancing condition.

When choosing a k-region uniformly at random in a $(c \log n)$-region, this attack can be prevented. Nevertheless, it seems to be quite challenging to formally prove that every other attack can also be handled. The basic technical problem seems to be that k-regions moved into a $(c \log n)$-region R due to a flip operation carry history with them making it hard to study the evolution of R. A strategy avoiding this problem is the following one.

Cuckoo rule with eviction and random transposition of neighboring k-region

Suppose that whenever a peer p leaves the system, a k-region R will be picked uniformly at random in the $(c \log n)$-region $\hat{R}$ containing p. All peers in R will be moved to new points chosen uniformly at random in $[0, 1)$, and then R will be flipped with another k-region chosen uniformly at random in $[0, 1)$. Afterwards, p rejoins the system using the cuckoo rule.

When using this strategy, R will not carry any history any more when moved into another $(c \log n)$-region, which greatly helps the analysis. Still, there are some technical difficulties because adversary can create some bias on the number and type of nodes that are replaced in that way. By choosing a heavily loaded $(c \log n)$-region, more nodes will

be moved, on expectation, than when using the cuckoo rule. Also, the adversary can bias the ratio between honest and adversarial nodes that are replaced. To avoid these problems, we came up with the following strategy, which can be analyzed with a reasonable amount of effort.

1.5 Join and leave operations that do work

The join operation works in the same way as the cuckoo rule. But whenever a peer wants to leave the network, we use the following leave operation:

Leave(v): If a peer v leaves the system, then a k-region R is chosen uniformly at random among the k-regions of $R_{kc\log n}(x)$ for some (sufficiently large) constant c, where x is the position of v. R is flipped with a k-region R' chosen uniformly at random in $[0,1)$, and then all peers in R (as well as v) have to rejoin the system from scratch using the cuckoo rule.

Hence, the departure of a peer may spawn several join operations. We call this algorithm the *cuckoo&flip strategy*. With this strategy the balancing and majority conditions can be kept, with high probability. More precisely, we will show the following result.

THEOREM 1.1. *For any constants ϵ and k with $\epsilon < 1/4 - (2\log k + 1)/k$, the cuckoo&flip strategy satisfies the balancing and majority conditions for any polynomial number of rejoin requests, with high probability, for any adversarial strategy within our model.*

Hence, as long as $\epsilon < 1/4$, only a constant factor overhead has to be paid (on average) compared to standard join and leave operations without any additional replacements of peers. The rest of the paper is devoted to the analysis of this theorem.

2. ANALYSIS OF THE CUCKOO&FLIP STRATEGY

Our analysis uses the following Chernoff bound for weighted random variables (e.g., [10]):

LEMMA 2.1. *Let $X_1,\ldots,X_n$ be independent random variables with $0 \le X_i \le b$ for all i, let $X = \sum_{i=1}^{n} X_i$ and $\mu = \mathrm{E}[X]$. Then it holds for any $\delta > 0$ that*

$$\Pr[X \ge (1+\delta)\mu] \le e^{-\delta^2\mu/(2b(1+\delta/3))}$$

and for any $0 < \delta < 1$ that

$$\Pr[X \le (1-\delta)\mu] \le e^{-\delta^2\mu/(2b)}$$

In the following, we simply call the peers *nodes*. Recall that a region is an interval of size $1/2^r$ in $[0,1)$ for some positive integer r that starts at an integer multiple of $1/2^r$. Let $\hat{R}$ be any fixed region of size $(c\log n)\cdot k/n$, for some constant c, for which we want to check the balancing and majority conditions over polynomial in n many rejoin requests. Thus, $\hat{R}$ contains exactly $c\log n$ many k-regions. We assume that the joining of the nodes proceeds in rounds, one round per join operation (note that the departure of a node may spawn several join operations, including its own). The *age* of a k-region is the difference between the current round and the last round when it was emptied due to a departure or a join operation, and the age of $\hat{R}$ is defined as the sum of the ages of its k-regions. A node in a k-region R is called *new* if it was placed into R when it joined the system, and otherwise (i.e., it was moved into R due to a join operation of another node) it is called *old*.

We assume that before the adversary starts with its rejoin requests, only the n honest nodes were in the system, and sufficiently many rejoin requests have been executed on the honest nodes so that every k-region has been entered by a new node at least once. Afterwards, the adversary enters with its ϵn adversarial nodes one by one, using the cuckoo rule in each round, and then it starts executing rejoin requests on the honest and adversarial nodes as it likes. The assumption of acting on a sufficiently old system significantly simplifies the proofs since otherwise extra proofs would be necessary for the start-up behavior of the system.

The next lemma follows directly from the cuckoo&flip rule because every k-region can have at most one new node at any time.

LEMMA 2.2. *At any time, $\hat{R}$ contains at most $c\log n$ new nodes.*

In order to bound the number of old nodes in $\hat{R}$, we first have to bound the age of $\hat{R}$ (Lemma 2.3). Then we bound the maximum number of nodes in a k-region (Lemma 2.4) and use this to bound the number of evicted honest and adversarial nodes in a certain time interval (Lemma 2.5). After that, we bound the number of old honest and adversarial nodes in $\hat{R}$ (Lemma 2.6) and use this to prove an upper bound on the number of honest nodes that the adversary has asked to leave $\hat{R}$ (Lemma 2.7). Finally, all insights are collected to prove Theorem 1.1.

There are two kinds of node departures (i.e., departures of nodes that are ordered by the adversary to rejoin). *External* departures are events in which a node outside of $\hat{R}$ leaves, and *internal* departures are events in which a node inside of $\hat{R}$ leaves. Whenever there is an external departure, the probability that $\hat{R}$ will get the emptied k-region is equal to $(c\log n)k/n$, which is equal to the probability of a k-region in $\hat{R}$ being hit by a joining node. In both cases, the age of that k-region in $\hat{R}$ is 0. Hence, with respect to the age, we can reduce the effect of external departures in our analysis to join operations. For internal departures, notice that some k-region in $\hat{R}$ switches its position with a k-region chosen uniformly at random from the entire system. So internal departures somehow help $\hat{R}$ in having k-regions of average age in it. These insights can be used to show the following lemma.

LEMMA 2.3. *At any time, $\hat{R}$ has an age within $[(1-\delta)(c\log n)n/(4k), (1+\delta)(c\log n)n/k]$, with high probability, where $\delta > 0$ is a constant that can be made arbitrarily small depending on the constant c.*

PROOF. Let the random variable A denote the average age of a k-region in the system.

First, we prove an upper bound on $\mathrm{E}[A]$. Consider any k-region R in the system and let us follow it as it gets flipped with other k-regions. Let the random variable X be the age of R. If we ignore the occasions in which R is emptied due to a departure in its current $(c\log n)k$-region, it holds that $\Pr[X = t] = (k/n)(1-(k/n))^{t-1}$, i.e., X is geometrically distributed with probability $p = k/n$. Hence, $\mathrm{E}[X] \le n/k$, which implies that $\mathrm{E}[A] \le n/k$.

Next, we prove a lower bound on $\mathrm{E}[A]$. Recall that in each leave operation at least two k-regions get emptied: one k-region due to the departure of a node in its $(k \cdot c \log n)$-region and one k-region when the departed node joins again. We do not consider it an extra round when a k-region is emptied due to a departure of a node, but when associating it with the round of the first join event caused by this, there can be at most two events of k-regions getting emptied in a round. Under this restriction, the lowest possible value A can get is if there are exactly two k-regions with the same age for all ages from 1 to $(n/k)/2$, which results in $A \geq (n/k)/4$. Thus, also $\mathrm{E}[A] \geq (n/k)/4$.

Now, focus on the $(k \cdot c \log n)$-region $\hat{R}$ and let $R_1, \ldots, R_C$ be its k-regions, $C = c \log n$. Recall the definition of internal and external departures. For each external departure, the probability that the emptied k-region is flipped with a k-region in $\hat{R}$ is equal to $C \cdot k/n$, which is equal to the probability that a k-region in $\hat{R}$ is emptied due to a join operation. Hence, the effect on the age of $\hat{R}$ for an external departure is equivalent to a join operation. However, internal departures are different, so they need a special treatment.

Suppose that all k-regions in the system are numbered from R'_1 to $R'_{n/k}$. Whenever there is an external departure that causes a k-region in $\hat{R}$ to be flipped, we assume that the two involved k-regions stay where they are and only their peers are flipped (i.e., the label distribution in $[0, 1)$ stays as it is). If there is an internal departure, then the k-regions affected by it flip their labels (in addition to flipping their peers). Hence, the labels of the k-regions in $\hat{R}$ only change when there is an internal departure.

Let us consider now any adversarial strategy that runs for T many rejoin operations, where T is polynomial in n. In this case, the adversary can cause at most T internal departures, which means that it can create at most T different subsets of k-regions out of $\{R'_1, \ldots, R'_{n/k}\}$ in $\hat{R}$. For an upper bound on the age of $\hat{R}$ we need to show that it is very unlikely that *any* of these T subsets has an age that is too large at *any* time during the adversarial strategy.

Consider some fixed subset S of k-regions. We know from above that for each of these k-regions it holds for its age X that $\Pr[X \geq t] \leq (1 - (k/n))^{t-1}$ for every t. Even though the ages of the k-regions slightly depend on each other (since no more than two can have the same age), it follows from techniques in the proof of Lemma 2.5 in [4] that the age of $\hat{R}$ is at most $(1 + \delta)(c \log n)(n/k)$, w.h.p., where $\delta > 0$ is a constant that can be made arbitrarily small depending on the constant c. This probability can still be kept polynomially small when multiplying it with T (which is polynomial in n) to take care of all T subsets S, if c is large enough.

It remains to prove a lower bound for the age of $\hat{R}$. Consider some fixed subset S of k-regions for $\hat{R}$. All k-regions in S that have been in $\hat{R}$ from the beginning (i.e., the *old* k-regions of $\hat{R}$) have an age X that satisfies $\Pr[X \geq t] \geq (1 - (k/n))^{2(t-1)}$ for every t since every round can have at most two events in which a k-region is emptied. For the other k-regions in S we note that they have been selected uniformly and independently at random from all k-regions in the system. For a lower bound on the age of these regions, we can use the age distribution for the lower bound on $\mathrm{E}[A]$. Let d_1 be the number of old k-regions in $\hat{R}$ and $d_2 = c \log n - d_1$ be the number of young k-regions in $\hat{R}$. From the proof of Lemma 2.5 in [4] it follows that the total age of the old regions is at least $d_1(n/2k) - \delta(c \log n)(n/k)$, w.h.p., and from Lemma 2.1 with $b = n/(2k)$ it follows that the age of the young k-regions in $\hat{R}$ is at least $d_2(n/4k) - \delta(c \log n)(n/k)$, w.h.p. Hence, the total age of $\hat{R}$ is at least $(1 - 2\delta)(c \log n)(n/k)/4$, w.h.p., where $\delta > 0$ is a constant that can be made arbitrarily small depending on the constant c. Again, this probability can still be kept polynomially small when multiplying it with T to take care of all T subsets S, if c is large enough. □

Since old nodes are only created in join operations, the following lemma holds.

LEMMA 2.4. *For any k-region R it holds at any time that R has at most $O(k \log n)$ old nodes, with high probability.*

PROOF. It is easy to show that every k-region R can have an age of at most $O((n/k) \log n)$, w.h.p. Consider any k-region R of age T and let X_t be the number of nodes that get replaced t rounds after R has last been evicted, $1 \leq t \leq T$. Since in the join operation a k-region is picked uniformly at random and the total number of nodes in the system is $(1 + \epsilon)n$, $\mathrm{E}[X_t] = (1 + \epsilon)k$ no matter how the nodes are distributed. Moreover, it trivially holds that $X_t \leq (1 + \epsilon)n$. Hence, $\mathrm{E}[\sum_t X_t] = (1+\epsilon)kT$ and it follows from the Chernoff bounds for weighted random variables (Lemma 2.1) that the probability that $\sum_t X_t \geq c[(1 + \epsilon)T + (1 + \epsilon)n \log n]$ is polynomially small in n if the constant c is large enough. Since each of these nodes is assigned to a new point uniformly and independently at random, it follows from standard Chernoff bounds that the number of nodes in R is at most $O(k \log n)$ w.h.p. □

Lemma 2.4 allows us to bound the number of honest and adversarial nodes that are evicted in a certain time interval (see also [4]).

LEMMA 2.5. *For a sufficiently large constant γ it holds for any time interval I of size $T = (\gamma/\epsilon) \log^3 n$ that the number of honest nodes that are evicted in I is within $(1 \pm \delta)T \cdot k$, with high probability, and the number of adversarial nodes that are evicted in I is within $(1 \pm \delta)T \cdot \epsilon k$, with high probability, where $\delta > 0$ can be made arbitrarily small depending on γ.*

Combining Lemmas 2.3 and 2.5, we obtain the following lemma.

LEMMA 2.6. *At any time, $\hat{R}$ has within $[(1-\delta)(c \log n)k/4, (1+\delta)(c \log n)k]$ old honest nodes and within $[(1-\delta)(c \log n) \cdot \epsilon k/4, (1 + \delta)(c \log n)\epsilon k]$ old adversarial nodes when ignoring node departures in $\hat{R}$, w.h.p.*

Suppose that we mark each position in a k-region R at which a node left *after* R got last emptied. The next lemma bounds the total number of marked positions in $\hat{R}$.

LEMMA 2.7. *At any time, there are at most $(2 \log k)(c \log n)$ marked positions in $\hat{R}$ belonging to honest nodes, with high probability, where the probability can be made arbitrarily small depending on the constant c.*

PROOF. Suppose that $\hat{R}$ has m marked positions. Then there must be s many k-regions in it of age at least $m/2$ each that have at least $m/2$ marked nodes in them (namely,

the older half of the marked positions), where s needs to be determined. Suppose that $m \geq (1+\delta)\gamma(c\log n)$ for some small constant $\delta > 0$ and sufficiently large constant γ. Then $s \geq \gamma(c\log n)/(2k)$, w.h.p., because Lemma 2.6 implies that $s = \gamma(c\log n)/(2k)$ many k-regions can have at most $(1+\delta)k \cdot s$ many honest nodes, w.h.p. The probability that there are s many k-regions in $\hat{R}$ of age at least $m/2$ is at most

$$\begin{aligned}\binom{c\log n}{s}\left(1-\frac{s}{c\log n}\right)^{m/2} &\leq (2ek/\gamma)^s \cdot e^{-(s\cdot m)/(2c\log n)} \\ &= e^{s(\ln(2ek/\gamma)-(1+\delta)\gamma/2)}\end{aligned}$$

Hence, if $\gamma = 2\log k$ and $c > k$ is sufficiently large, then the probability is polynomially small that such a subset exists. □

When combining Lemmas 2.2, 2.6 and 2.7, we obtain a lower bound of $(c\log n)k/4 - (2\log k)(c\log n)$ honest nodes and an upper bound of $(c\log n)\epsilon k + c\log n$ adversarial nodes in $\hat{R}$ in the worst case (when ignoring the $(1+\delta)$ and $(1-\delta)$ factors, which can be brought arbitrarily close to 1). The majority condition holds as long as

$$(c\log n)k/4 - (2\log k)(c\log n) > (c\log n)\epsilon k + c\log n$$

which implies that ϵ must satisfy $\epsilon < 1/4 - (2\ln k + 1)/k$. Due to the upper and lower bounds in Lemmas 2.6 and 2.7, also the balancing condition holds, which implies Theorem 1.1.

3. CONCLUSION

In this paper we presented oblivious join and leave operations that can protect an overlay network against any sequence of rejoin requests. The operations only create a constant factor overhead, on expectation, with respect to the randomness and the number of peer movements needed in standard join and leave operations without any additional peer movements besides the joining and leaving peer. Still, our operations look involved, so it would be interesting whether one of the operations for which we do not know yet whether they work can also be shown to be robust against adversarial join and leave behavior. Also, it would be interesting to find out to which extent join and leave events can happen concurrently so that it is still possible to design protocols for scalable and robust overlay networks.

REFERENCES

[1] J. Aspnes and G. Shah. Skip graphs. In *Proc. of the 14th ACM Symp. on Discrete Algorithms (SODA)*, pages 384–393, 2003.

[2] B. Awerbuch and C. Scheideler. Group Spreading: A protocol for provably secure distributed name service. In *Proc. of the 31st International Colloquium on Automata, Languages and Programming (ICALP)*, 2004.

[3] B. Awerbuch and C. Scheideler. Robust random number generation for peer-to-peer systems. In *10th Intl. Conference On Principles of Distributed Systems (OPODIS)*, pages 275–289, 2006.

[4] B. Awerbuch and C. Scheideler. Towards a scalable and robust DHT. In *Proc. of the 18th ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, 2006. See also http://www14.in.tum.de/personen/scheideler.

[5] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. Wallach. Security for structured peer-to-peer overlay networks. In *Proc. of the 5th Usenix Symp. on Operating Systems Design and Implementation (OSDI)*, 2002.

[6] M. Castro and B. Liskov. Practical Byzantine fault tolerance. In *Proc. of the 2nd Usenix Symp. on Operating Systems Design and Implementation (OSDI)*, 1999.

[7] S. Crosby and D. Wallach. Denial of service via algorithmic complexity attacks. In *Usenix Security*, 2003.

[8] J. R. Douceur. The sybil attack. In *Proc. of the 1st International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.

[9] F. Kuhn, S. Schmid, and R. Wattenhofer. A self-repairing peer-to-peer system resilient to dynamic adversarial churn. In *Proc. of the 4th International Workshop on Peer-to-Peer Systems (IPTPS)*, 2005.

[10] McDiarmid. Concentration. In M. Habib, C. McDiarmid, J. Ramirez-Alfonsin, and B. Reed, editors, *Probabilistic Methods for Algorithmic Discrete Mathematics*, pages 195–247. Springer Verlag, Berlin, 1998.

[11] M. Naor and U. Wieder. Novel architectures for P2P applications: the continuous-discrete approach. In *Proc. of the 15th ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, 2003.

[12] S. Nielson, S. Crosby, and D. Wallach. Kill the messenger: A taxonomy of rational attacks. In *Proc. of the 4th International Workshop on Peer-to-Peer Systems (IPTPS)*, 2005.

[13] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proc. of the ACM SIGCOMM '01*, 2001.

[14] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz. Handling churn in a DHT. In *USENIX Annual Technical Conference*, 2004.

[15] J. Saia, A. Fiat, S. Gribble, A. Karlin, and S. Saroiu. Dynamically fault-tolerant content addressable networks. In *Proc. of the 1st International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.

[16] C. Scheideler. How to spread adversarial nodes? Rotate! In *Proc. of the 37th ACM Symp. on Theory of Computing (STOC)*, pages 704–713, 2005.

[17] A. Singh, M. Castro, A. Rowstron, and P. Druschel. Defending against Eclipse attacks on overlay networks. In *Proc. of the 11th ACM SIGOPS European Workshop*, 2004.

[18] E. Sit and R. Morris. Security considerations for peer-to-peer distributed hash tables. In *Proc. of 1st International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.

[19] M. Srivatsa and L. Liu. Vulnerabilities and security threats in structured overlay networks: A quantitative analysis. In *Proc. of the 20th IEEE Computer Security Applications Conference (ACSAC)*, 2004.

[20] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proc. of the ACM SIGCOMM '01*, 2001. See also `http://www.pdos.lcs.mit.edu/chord/`.