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
Institute for 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

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

its adversarial peers and knows the entire overlay network at any point in time. It can use this information to decide in an arbitrary 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 (with very high probability). That is, the join and leave operations do not have to be able to distinguish between the honest and adversarial peers and, even more, do not have to 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, but our results can also be used for other spaces. 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).

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., [3]). 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. 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), and finally we present an approach that does work (Section 1.4). 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, 8, 10, 11, 16]), 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 [8]. 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 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 [6, 7] and various solutions have been proposed that may help thwart these attacks in practice [4, 5, 14, 9, 13, 15] 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 [12]. 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 [3]. 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 [3] 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

Before we present our solution to the problem, we show that several variants of the cuckoo rule do not work.

Cuckoo rule with random peer transposition. The previous attack shows that just using the cuckoo rule alone is not sufficient. So what 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 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. Hence, on expectation, within $O(\log n)$ departures R will only have adversarial peers, which violates the majority condition.

Cuckoo rule with random transposition of k-region. Suppose that whenever a peer departs some k-region, this k-region is flipped with some other k-region chosen uniformly at random in $[0, 1)$. Also for this rule the majority condition can easily be violated with an attack similar to the one for the previous rule.

Cuckoo rule with random transposition of neighboring k-region. Suppose that whenever a peer p leaves the system, a random k-region R will be picked in the $(k \cdot c \log n)$-region $\hat{R}$ containing p and R will be flipped with another k-region chosen uniformly at random in $[0, 1)$. Interestingly, the majority condition cannot be violated any more in this case but the balancing condition can be violated. The attack for this is quite subtle. Focus on a particular $(k \cdot c \log n)$-region R. Always pick a $(k \cdot c \log n)$-region R' different from R that contains at least one k-region R'' with $\Theta(\log n)$ peers. Such a k-region is guaranteed to exist somewhere, w.h.p., for a random distribution of the peers in $[0, 1)$. Remove a peer in R'. The probability that R'' will be moved into R in this case is $(c \log n)k/n$. Hence, within $\Theta(n)$ many rejoin requests it is quite likely that $\Theta(\log n)$ k-regions with $\Theta(\log n)$ peers each will be moved into R resulting in $\Theta(\log^2 n)$ peers in R, which violates the balancing condition.

1.4 Join and leave operations that do work

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

Cuckoo&flip rule: 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. All peers in R have to rejoin the system from scratch using the cuckoo rule, and R is flipped with a k-region R' chosen uniformly at random in $[0, 1)$.

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

Theorem 1.1 *For any constants ϵ and k with $\epsilon < 1/2 - 2/k$, 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 $\epsilon < 1/2$, only a constant factor overhead has to be paid (on average) compared to standard join and leave operations without any perturbation rules. The rest of the paper is devoted to the analysis of this theorem. Due to strict page limitations, we will only sketch the proof.

2 Analysis of the cuckoo&flip strategy

In the following, we simply call 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 leave or 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 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.

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 2.1 *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 2.2). Then we bound the maximum number of nodes in a k-region (Lemma 2.3) and use this to bound the number of evicted honest and adversarial nodes in a certain time interval (Lemma 2.4). After that, we bound the number of old honest and adversarial nodes in $\hat{R}$ (Lemma 2.5) and use this to prove an upper bound on the number of nodes that left $\hat{R}$ (Lemma 2.6). Whereas the proofs of Lemmas 2.3 and 2.4 follow from previous work [3], the proofs of Lemmas 2.2 and 2.6 are new and tricky.

There are two kinds of leave events. Type-1 events involve departures of nodes outside of $\hat{R}$ and type-2 events involve departures of nodes within $\hat{R}$. Whenever there is a type-1 event, then 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 type-1 events in our analysis to join operations. For type-2 events, notice that some k-region in $\hat{R}$ switches its position with a k-region chosen uniformly at random from the entire system. These insights and some facts about the age distribution of the k-regions (for example, the worst expected age is at most $(1 + \delta)n/k$ and the average expected age is at least $(1 - \delta)n/(2k)$) can be used to show the following lemma.

Lemma 2.2 *At any time, $\hat{R}$ has an age within $[(1 - \delta)(c \log n)n/(2k), (1 + \delta)(c \log n)n/k]$, with high probability, where $\delta > 0$ is a constant that can be made arbitrarily small depending on the constant c.*

Furthermore, since old nodes are only created in join operations, the following lemma is easy to show (see also [3]).

Lemma 2.3 *For any k-region R it holds at any time that R has at most $O(k \log n)$ old nodes, with high probability.*

Lemma 2.3 allows us to bound the number of honest and adversarial nodes that are evicted in a certain time interval (see also [3]).

Lemma 2.4 *For any time interval I of size $T = (\gamma/\epsilon) \log^3 n$, 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 2.2 and 2.4, we obtain the following lemma.

Lemma 2.5 *At any time, $\hat{R}$ has within $[(1-\delta)(c \log n)k/2, (1+\delta)(c \log n)k]$ old honest nodes and within $[(1-\delta)(c \log n)\epsilon k/2, (1+\delta)(c \log n)\epsilon k]$ 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 $\hat{R}$. It needs the crucial fact that $(1+\delta)c \log n$ marked positions must spread over $\Omega(\log n)$ k-regions in $\hat{R}$, w.h.p., if c is sufficiently large, which follows from Lemma 2.5.

Lemma 2.6 *At any time, there are at most $(1+\delta)c \log n$ marked positions in $\hat{R}$, with high probability, where $\delta > 0$ is a constant that can be made arbitrarily small depending on c.*

When combining Lemmas 2.1, 2.5 and 2.6, we obtain a lower bound of $(c \log n)k/2 - c \log n$ honest nodes and an upper bound of $(c \log n)\epsilon k + c \log n$ adversarial nodes in $\hat{R}$ in the worst case. Hence, the majority condition holds if $\epsilon < 1/2 - 2/k$. Due to the upper and lower bounds in Lemmas 2.5 and 2.6, also the balancing condition holds, which implies Theorem 1.1.

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. 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.

[4] 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.

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

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

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

[8] 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.

[9] 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.

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

[11] 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.

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

[13] 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.

[14] 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.

[15] 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.

[16] 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/`.