

Maelstrom: Churn as Shelter

*Tyson Condie
Varun Kacholia
Sriram Sankararaman
Petros Maniatis
Joseph M. Hellerstein*

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2005-11

<http://www.eecs.berkeley.edu/Pubs/TechRpts/2005/EECS-2005-11.html>

November 10, 2005



Copyright © 2005, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

Acknowledgement

We would like to thank Sean Rhea for his invaluable help in extending Bamboo with the Maelstrom extensions, David Wagner for his thoughtful comments on earlier versions of this paper, and the anonymous reviewers for their feedback. Tyson Condie and Joe Hellerstein are partially supported by an NSF grant (contract numbers 0209108 and 0225660), and by a gift from Microsoft Corporation.

Maelstrom: Churn as Shelter

Tyson Condie, Varun Kacholia, Sriram Sankararaman, Joseph M. Hellerstein, Petros Maniatis
UC Berkeley and Intel Research Berkeley

Abstract

Structured overlays are an important and powerful class of overlay networks that has emerged in recent years. They are typically targeted at peer-to-peer deployments involving millions of user-managed machines on the Internet. In this paper we address routing-table poisoning attacks against structured overlays, in which adversaries attempt to intercept traffic and control the system by convincing other nodes to use compromised nodes as their overlay network neighbors. In keeping with the fully-decentralized goals of structured overlay design, we propose a defense mechanism that makes minimal use of centralized infrastructure. Our approach, induced churn, utilizes periodic routing-table resets, unpredictable identifier changes, and a rate limit on routing-table updates. Induced churn leaves adversaries at the mercy of chance: they have little opportunity to strategize their positions in the overlay, and cannot entrench themselves in any position that they do acquire. We implement induced churn in Maelstrom, an extension to the broadly used Bamboo distributed hash table. Our Maelstrom experiments over a simulated network demonstrate robust routing with very modest costs in bandwidth and latency, at levels of adversarial activity where unprotected overlays are rendered almost completely useless.

1. Introduction

In recent years, the systems and networking communities have devoted significant attention to techniques for coordinating large numbers – millions – of computers in a decentralized fashion. Originally motivated by peer-to-peer filesharing applications, this research demonstrated massively distributed systems whose funding, provisioning, and management are decentralized across numerous parties with little shared trust. More recently, this design philosophy has been applied to a host of applications including content distribution networks [13], geographic location services [17], file systems [10], monitoring systems [4], document archives [30], and distributed query processing [19].

Central to any of these systems is the notion of an *overlay network*: a coordination mechanism for nodes running a distributed application to track each other, and to route

messages among themselves. Such large, open systems face constant *churn*, the arrival and departure of nodes, as some fail, hardware is replaced, connectivity changes, or software is upgraded. Much design and engineering are devoted to maintaining performance while tolerating churn.

A particular class of overlays, *structured overlays* such as Chord [29] and Pastry [25], presents a hash table abstraction on top of a population of networked computers. Each participating node in the overlay has an ID from a large identifier space, and is responsible for handling messages addressed to an extent of the identifier space around its own ID. In order to route messages in the overlay, every node maintains a routing-table of “links.” The set of nodes and links in the system forms a structured network graph, over which ID lookups can be routed to the responsible node efficiently, even as the network churns. When used to store data, structured overlays are often called *distributed hash tables* (DHTs), though many structured overlay applications do not require storage.

An adversary who can subvert overlay routing can modify the overlay’s behavior and hurt applications: for example, she can convince a correct (i.e., “good”) node to redirect an outgoing link to herself, thereby *poisoning* its routing-table. All lookups routed via that link will end up in the adversary’s control; she can forward them or respond to them as she wishes. This has been called *routing-table poisoning* or an *eclipse attack* in the literature. Once an adversary poisons a good node’s routing-table, she can *amplify* that poisoning by intercepting the good node’s maintenance traffic, and convince the node to update its routing-table to include additional compromised neighbors (Section 2.2).

Previous Proposals: Previous defenses against eclipse attacks have typically involved the use of a trusted third party that regulates indirectly how nodes partition the ID space, for example, by authoritatively assigning IDs to nodes [7]. The intuition is that if the adversary’s node IDs are chosen uniformly at random by an uncompromised authority, then even the adversary receives responsibility of an ID space share that is proportional to the number of her nodes. She can therefore affect the system only in proportion to her presence. Centralized, globally trusted certification authorities can be burdensome and difficult to adminis-

ter [11], especially when multiple, mutually distrusting administrative domains are involved. However, they can offer relief from rampant adversarial activity such as the use of forged, throwaway identities (also known as Sybil identities [12]).

Note that defenses against Sybil attacks do not mitigate the threat of amplification once a compromised node is chosen as a neighbor. This risk has become more important in recent, highly optimized structured overlays, which make aggressive use of routing-table updates not only to address churn in the network, but also for performance optimizations (e.g., latency minimization over lookup paths through the graph [14, 23]).

Our Contribution: In this paper¹ we ask two questions. First, can there be an effective defense against route poisoning attacks with a simpler, less trusted, centralized component that is easy to audit and replicate? Second, can there be a practical, implementable defense against *eclipse attacks* that addresses the performance optimizations used in recent structured overlays? We present techniques that answer both questions in the affirmative.

Specifically, we propose the use of *induced churn* as a defense against eclipse attacks. Induced churn consists of three techniques: *periodic reset of routing-tables* to less efficient but more attack-resistant ones, forced *unpredictable identifier changes*, and *rate limitation on routing-table updates*. We argue that by never allowing the overlay to quiesce, we rob the adversary of the opportunity to plan ahead on node positioning prior to an attack, and of her ability to entrench herself, amplifying her position over time. We show that for a typical, well-tuned structured overlay we reduce routing-table poisoning by an order of magnitude and increase the probability of successful lookups by as much as a factor of 5, while incurring a maintenance overhead of under 1 KBps at each node, low enough even for home users over dial-up connections. Induced churn is applicable to any overlay application that requires node *organization* without persistent storage (e.g., for query processing, multicast, or network monitoring); however for storage applications where churn imposes data migration, induced churn might be less appropriate (see Section 5).

In Section 2 we present relevant background on structured overlays, routing threats against them, and some previously proposed solutions that form the basis for our defenses. Section 3 presents the design of our induced churn defense against eclipse attacks. We evaluate our design in Section 4, with experimental results on *Maelstrom*, a prototype implementation of induced churn as an extension of the Bamboo structured overlay [23]. Our evaluation measures the improved security of the system as well as the performance hit caused by routing-table reset, unpredictable identifier changes, and limiting routing-table updates. . Further

we explore extensions, possible limitations, and big-picture implications of this work in Section 5. Finally, we conclude with related work and our future research agenda. Appendix A contains a simple analysis of our intuition and design, further supporting our experimental results. We wish to prove some of the security properties of our system as a part of our future work.

2. Background

As background, we present a brief primer on structured overlay networks. We then discuss the class of attacks that concern us, and previously-proposed defenses, before introducing induced churn in Section 3.

2.1. Structured Overlay Networks

An overlay network is a virtual network implemented on top of an established underlying network of routers; in our discussion we will focus on Internet overlays. Applications running at participant machines communicate along the edges of the overlay network using unicast transport services provided by the underlying network – in our case, by IP. Therefore, a message over an edge of the overlay may traverse many edges (IP router links) in the underlying network. The algorithm choosing overlay edges differ among overlay designs.

A structured overlay builds its topology according to a particular model graph structure such as a hypercube, a torus, a de Bruijn graph, etc. To facilitate this construction, overlay nodes take identifiers from a large ID space I , which is typically the range of a cryptographic hash function (e.g., SHA-1), and is chosen to be sufficiently large (e.g., 160 bits or more) to minimize name collisions. Overlay nodes take random IDs from I . Then the canonical model graph structure chosen for the overlay is embedded in the ID space and mapped to the existing overlay nodes.

To effect this mapping, responsibility for ID space ranges is partitioned among the nodes in the overlay at all times. In our discussion we will assume that each node is responsible for the IDs that are nearest to it in I (Figure 1); other partitioning schemes are used in the literature, but the choice has no impact on our techniques below.

API: The interface to the structured overlay consists of a single `lookup(id)` call. In response, the overlay must locate the IP address of the node currently responsible for ID `id`, typically by routing a message to that destination.

Topology and Routing: The overlay’s network topology – the mapping of the model graph structure to overlay nodes and links – is captured via the *routing-table* maintained at each participant node. For concreteness, we use Pastry [25] as an example here. Pastry is relatively easy to

¹An abbreviated version of this paper appeared in [9]

