# Machiavellian Routing: Improving Internet Availability with BGP Poisoning

Ethan Katz-Bassett David R. Choffnes Ítalo Cunha[†] Colin Scott[‡]

Thomas Anderson Arvind Krishnamurthy

University of Washington
Computer Science & Engineering
Seattle, WA, USA
{ethan, choffnes, tom, arvind}
@cs.washington.edu

[†]Technicolor
Paris, France
italo.cunha@technicolor.com

[‡]University of California
Electrical Engineering & Computer Science
Berkeley, CA, USA
cs@cs.berkeley.edu

## ABSTRACT

We propose a new approach to mitigate disruptions of Internet connectivity. The Internet was designed to always find a route if there is a policy-compliant path; however, in many cases, connectivity is disrupted despite the existence of an underlying valid path. The research community has done considerable work on this problem, much of it focused on short-term outages that occur during route convergence. There has been less progress on addressing avoidable long-lasting outages. Our measurements show that long-lasting events contribute significantly to overall unavailability.

To address these long-term problems, we develop a system, Machiavellian routing, for automatic failure remediation, centered around the use of BGP poisoning. With poisoning, an edge network can cause other networks to send traffic to it via paths that avoid a problem in a particular transit ISP. We describe the key challenges to using poisoning to improve Internet connectivity, and we develop a set of techniques to use it predictably, accurately, and effectively.

**Categories and Subject Descriptors:** C.2.2 [Computer communication networks]: Network protocols

**General Terms:** Measurement, Reliability

## 1. INTRODUCTION

With the proliferation of mobile apps and thin clients, more people connect to the Internet more often. Much of our data lives in the cloud, and we expect it to be available anytime, from anywhere. In the future, we would like to use the Internet to run critical services, such as traffic control and outpatient medical monitoring. Because of these current, emerging, and future uses, the Internet needs to provide high availability.

We focus on disruptions to connectivity due to routing problems in which a working policy-compliant path exists, but networks instead route along a different path that fails to deliver packets. In theory this should never happen – if working paths exist, the Internet protocols are designed to find them, even in the face of failures. In practice, such outages are common. Specifically, we show that even well-provisioned cloud data centers experience frequent routing problems. Existing research provides promising approaches to dealing with the transient unavailability that occurs during routing protocol convergence [11, 14, 15, 16, 22], so we focus on events that persist over longer timescales that are less likely to be convergence-related. Our data shows that these long-term outages contribute significantly to end-to-end unavailability.

We believe that Internet availability would improve if data centers (and other well-provisioned edge networks) were given the ability to repair persistent routing problems, regardless of which AS along the path is responsible for the outage. An edge network is in a position to observe routing disruptions, when it can no longer reach parts of the Internet. We assume that an alternate path exists; some working policy-compliant path can deliver traffic, and our task is to cause the Internet to use it.

We could accomplish our goal if three properties hold: (i) the edge network can accurately determine which AS is causing the problem; (ii) the Internet provides a mechanism for an edge network to disable routes that traverse the failing AS, triggering route exploration to find new paths; and (iii) the edge network can tell when the failure is resolved so that it can revert to normal routing.

While this might seem to require a rearchitecture of the Internet, we design our approach to work with existing protocols. To achieve the first subgoal, we propose active measurement techniques that build on Hubble [13] and reverse traceroute [12] to accurately locate problems. For the second, we need a means to trigger a re-route without requir-

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 and/or a fee.
*Hotnets '11,* November 14–15, 2011, Cambridge, MA, USA.
Copyright 2011 ACM 978-1-4503-1059-8/11/11 ...$10.00.

ing modifications to BGP, the Internet's interdomain routing protocol. We propose doing so using BGP poisoning [5, 7]. In this technique, to cause incoming paths to avoid traversing an AS $A$, an origin AS $O$ inserts $A$ into its announcements, so that it appears that $A$ has already been visited. When the announcements reach $A$, BGP's loop-prevention mechanisms will drop the announcement. ASes that otherwise would have routed through $A$ will instead learn only paths that avoid $A$. For the third subgoal, to detect when to stop poisoning, we propose using active probes to a "sentinel" prefix that does not carry production traffic. We refer to this set of techniques as Machiavellian routing, as they allow the effective and strategic poisoning of problem ASes, thereby providing a practical means for end networks to improve Internet availability.

## 2. QUANTIFYING UNREACHABILITY

To quantify how much of a problem Internet outages are, we conducted a measurement study using Amazon EC2, a major cloud provider [8]. EC2 presumably has the resources, business incentives, and best practices available for combating Internet outages. We show that even EC2 data centers experience many short- and long-term network connectivity problems. That network outages affect even well-connected services argues for a new solution; we believe that Machiavellian routing can fill that need.

We rented EC2 instances in each of the four AWS regions from July 20, 2010 until August 29, 2010. Our vantage points issued pings every 30 seconds to 250 targets, consisting of 5 routers each from the 50 highest-degree ASes [21]. We focus on paths to core routers, as these should be even more reliable than paths to end hosts. We define an outage as four or more consecutive dropped pings from a single vantage point to a destination. This methodology means that the minimum outage duration we consider is 90 seconds; our study does not track shorter periods of loss such as those occurring during routing protocol convergence [15].

**Much of the unavailability comes from long-lasting problems.** We define a partial outage as one in which at least one vantage point could reach the destination (in this case, a core router), establishing that the destination is up. Figure 1 shows the duration of the 10,308 partial outages in our study. Most outages are relatively short; more than 90% lasted less than 10 minutes (solid line). However, these short outages account for only 16% of the total unavailability (dotted line). The relatively small number of long-lasting problems account for much of the overall unavailability. Delayed protocol convergence does not explain long outages [15]. These types of problems are currently resolved only over slow, human timescales, opening up the possibility for significantly improving availability via automated response.

**Most outages are partial.** In 79% of the outages in our EC2 study, some vantage points had connectivity with the destination while others did not. All EC2 instances maintained connectivity with a controller at our university throughout

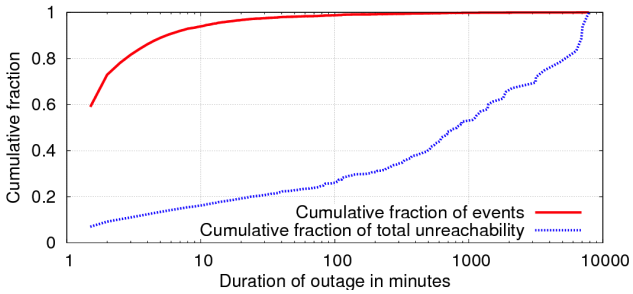


**Figure 1: For partial outages observed from EC2, the fraction of outages of at most a given duration (solid) and their corresponding fraction of total unreachability (dotted). Note that the x-axis is on a log-scale. We found that more than 90% of outages lasted 10 minutes or less, but that those short outages contributed only 16% of the total unavailability.**

the study, so we know the network was not physically partitioned between the vantage points that could reach the destination and those that could not reach. Either a misconfiguration or routing policy must be keeping some vantage points from finding the working routes, suggesting that it may be possible to route around the problem.

**Problems are not just in the endpoint ASes.** In a measurement study we describe briefly in §4.2, we found that traceroutes along failing paths often did not reach even the target AS, suggesting problems in transit ASes. The endpoints of a path are likely in the best position to detect a problem and have the most incentive to fix it, but they currently have no direct control over transit networks on which they depend.

## 3. POISONING BASICS AND CHALLENGES

We now consider how an automated response to long-lasting, partial outages might work. Suppose two ASes want to communicate, but cannot, because of some problem in an AS on the path between them. The endpoints need a way to notify networks on the path that the problem AS is not successfully forwarding traffic, thereby encouraging them to choose alternate routes that restore connectivity. Ideally, we would like to redesign BGP to let the origin AS specify such requirements explicitly with a signed announcement. For now, we use mechanisms already available in BGP to perform the notifications, to arrive at a solution that is deployable today.

Under BGP, a router matches an incoming packet against the IP prefixes in its table and routes via the path for the most-specific (i.e., longest matching) prefix. To establish the paths, an origin AS (a prefix's owner) announces to neighboring ASes that it is originating the prefix. Each AS adds itself to the path and announces the path to its neighbors. An AS learning multiple paths to a prefix is free to choose whichever it prefers, but will announce only the preferred one. An AS should never announce a route unless it is willing to deliver traffic along it; hence, partial connectivity for

a prefix often indicates that some AS along a path is not behaving correctly. Since routes are per-prefix, an update to one prefix does not change routes to others, except overlapping portions of less-specific ones.

BGP provides limited means for conveying information between ASes. BGP communities, a common avenue for such communication, are not currently standardized enough and do not propagate far enough to provide the Internet-wide control we seek. However, the AS path is always propagated with the announcement, and so it is a natural candidate to carry outage information. We propose using BGP's built-in loop prevention to "poison" a problem AS that is announcing routes but not forwarding packets. To poison an AS *A*, the origin can announce the prefix with *A* as part of the path, causing *A* to reject the path (to avoid a loop) and withdraw its previous path from its neighbors [5, 7]. This withdrawal causes ASes that previously routed via *A* to explore alternatives. Although BGP loop prevention was not intended to give *O* control over routes in other ASes, it lets us experiment with failure avoidance. Importantly, the poison affects only traffic to *O*'s prefix experiencing the problem.

However, poisoning on its own is not a solution. We still have to address the following challenges:

**Predictability**: As seen in §2, most outages resolve quickly. In §4.1, we investigate whether we can differentiate transient problems from longer-term outages that need poisoning for remediation.

**Accuracy**: To make poisoning useful, we must pinpoint which AS to poison. With standard tools, it is extremely hard to know which AS is causing a problem. Frequently, an operator turns to traceroute when unable to reach a destination *D*, to learn where the failure might be. In these cases, the traceroute "trails off" after some final responsive hop. While it is tempting to assume that the failure is located just beyond this last hop, the true failure may be on an asymmetric reverse path. Traceroute cannot account for these issues, leaving operators with a false positive; poisoning the AS of the traceroute's last hop would not fix the problem.

Recent work addressed some of these limitations. Hubble isolates the direction of failure [13]. Reverse traceroute measures reverse paths from arbitrary destinations [12]. However, it requires both directions of the path to be working. Neither system can measure the working path in unidirectional failures or locate reverse failures. In §4.2, we present an approach that builds on these systems to isolate failures.

**Disruptiveness**: Poisoning an announcement by inserting an AS increases AS path length, and experiments found that BGP normally takes multiple minutes to converge when switching to longer paths, with accompanying packed loss to the prefix during this period [15]. Given that such loss would affect those networks with working paths to the prefix, we would like to poison in a way that shortens and smooths this convergence period.

Further, in some cases, an AS will not fail completely. Rather, some paths through an AS to a particular prefix may work while others have failed, but poisoning disables the entire AS. Ideally, we would like to make use of the working paths while routing around the broken ones. In §4.3, we propose how to make our BGP announcements in a way that should make them less disruptive.

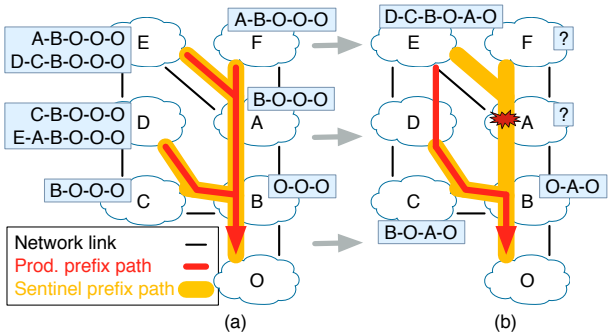


**Figure 2: Routes and routing tables (a) before and (b) after *O* poisons *A* to avoid a problem. Each table shows *only paths to the production prefix*, with the in-use, most-preferred route at the top. Poisoning *A* for the production prefix causes it to withdraw its route from *E* and *F*, forcing *E* to use its less-preferred route through *D* and leaving *F* with only the sentinel. Routes to the sentinel prefix do not change, allowing *O* to check when the problem has resolved.**

**Revertibility**: Assuming we can accurately identify the AS *A* responsible for a problem, we can use BGP poisoning to target it and cause other ASes to route around *A*. However, *A* will eventually resolve the underlying issue, at which point we would like to be able to revert to the unpoisoned path, allowing ASes to use paths through *A*, if preferred. When the poisoned announcement is in place, however, *A* will not have a path to the prefix in question, so we will not have a way to test the prefix's reachability via *A*. §4.3 describes how we propose to address this.

**Efficacy**: Finally, for poisoning to be effective at resolving problems, the Internet must have enough topological resiliency for ASes to find alternate routes once paths through the poisoned AS become unavailable. In §5, we investigate how often alternate paths exist.

## 4. MACHIAVELLIAN ROUTING

Here, we describe Machiavellian routing, a set of techniques that allow the effective and strategic poisoning of ASes that are causing problems. We first sketch the four stages of Machiavellian routing: what happens in the steady state, before any problems; what happens when a problem is detected, to determine whether and whom to poison; what announcements are made during poisoning; and, finally, how the system detects that the problem is resolved and then reverts to the steady state.

First, under normal conditions, the origin AS *O* prepends to the *production prefix P*'s announcement, announcing *O-O-O* as the baseline announcement (Fig. 2(a)). *O* also an-

nounces a *sentinel prefix S*, with the same path. *S* is a less-specific prefix that contains *P*, but also contains a sub-prefix that does not host any production services.

Second, when *O* detects that some networks (*E* and *F* in the figure) cannot reach *P*, our system uses active measurements to locate the AS *A* causing the problem. Then, our system predicts whether the problem is likely to persist long enough to be a candidate for poisoning.

Third, if the system decides to poison *A*, *O* announces *O-A-O* for *P* (Fig. 2(b)). This path has the same length and next hop as the baseline, minimizing any disruption of working paths. Meanwhile, *O* continues to announce the sentinel prefix with the baseline, ensuring that ASes like *F* that are "captive" behind *A* have a BGP path [4].

Fourth, from an address that is in *S* and not in *P*, the system pings *E* and *F*. Responses are routed along the unpoisoned baseline paths, so a successful ping indicates that the paths work again. When they do, *O* reverts *P* to the baseline announcement, returning to the initial state.

In the following sections, we describe these mechanisms for Machiavellian routing in more detail, as well as how they address the challenges raised in §3.

## 4.1 Deciding when to poison

We hope to demonstrate that it is possible to learn properties that, with high likelihood, allow us to differentiate between outages that will clear up quickly and those that will persist.

As initial evidence of this predictability, we analyzed outage durations from our EC2 study. We found that, of the 12% of problems that persisted for as least five minutes, 51% lasted at least another five minutes. Further, of the problems that lasted ten minutes, 68% persisted for at least five minutes past that. These results indicate that if an outage affecting EC2 is not transient, the problem is likely to persist long enough to justify using poisoning to fix it.

Our approach allows an edge network to use poisoning to attempt to resolve problems on the reverse paths from clients back to it. We focus on this direction of failures because it is generally the harder one to resolve, as the edge network has less visibility into problems and less control over routes. If, instead, the problem is on the forward path, standard traceroute and traditional rerouting approaches often allow an edge network to resolve such issues, without needing poisoning. For example, a well-connected data center can choose to route through a different provider that avoids the problem.

## 4.2 Accurately locating failures

We propose building upon reverse traceroute and Hubble to provide much better failure isolation than previously possible. Because of space limitations, we sketch only the basics of the approach here; we wish simply to convince the reader that we could isolate failures along either the forward or reverse path, sufficient to enable accurate BGP poisoning.

To isolate a failure, our system must first determine whether the outage is on the forward and/or the reverse path. We use Hubble's approach for determining the working direction of a path (if any), relying on spoofed probes to traverse only one direction at a time.

Next, we use a combination of historical and on-demand active measurements to isolate the AS causing the failure. If the problem is on the reverse path, the origin AS *O* could avoid the problem via poisoning if the problem AS were known. In this case, a source *S* in AS *O* cannot directly measure a reverse traceroute from destination *D*, as such a measurement would require a response from *D* to determine the initial hops, even with the help of a spoofing node *V*. However, if *S* has a pre-measured reverse path from *D* that predates the failure – say, by periodically issuing reverse traceroutes from a set of important destinations – then it can determine the hop *h* along that path that is farthest from *S* and can still reach it, as well as the first hop *h'* past *h* that cannot. This means that *h'* no longer has a working path to *S*, and so poisoning the AS containing *h'* may resolve the problem. If it does not, one explanation is that, because *h'* did not have a route, *D* switched to another path, which also did not work. We are evaluating techniques that use historical path information, multiple vantage points, and path prediction [17] to identify likely alternate paths, which would enable our system to test them and find candidates for poisoning. For forward failures, we have similar techniques to provide rich information about the location of the problem and about alternate working paths.

We implemented an initial system based on these techniques, combined with atlases that track up-to-date forward and reverse paths between a set of PlanetLab vantage points and a set of key PoPs in transit networks [17]. In the last month, the system identified 320 partial outages as candidates for avoidance using techniques such as poisoning. In 40% of cases, the system identified a different suspected failure location than what one would assume using traceroute alone. As an example, traceroutes from George Mason University to a network in Russia started failing, with the last responsive hop in TransTeleCom. However, our system located the failure as being along the reverse path, in Rostelecom, an ISP that did not appear on current or historical forward traceroutes.

Further, we noticed emails to an operators' mailing list [20] discussing some outages our system was tracking, giving anecdotal evidence that our failure localization techniques work. For example, on June 3, our system blamed XO for a number of outages towards destinations in XO and elsewhere. Later, operators posted about an XO fiber cut affecting traffic through Level3 and XO. In the future, we plan to validate our techniques via post-mortem emails to operators.

## 4.3 Machiavellian BGP announcements

**Prepend the announcement.** We believe we can speed up convergence and reduce the path exploration induced by our

poisoned announcements by using *O-O-O* as the baseline path, then announcing *O-A-O* if we later need to poison *A*. These two announcements are the same length and have the same next-hop AS, and so, under default BGP, these are equally preferred. Conventional wisdom suggests that this property will speed convergence by reducing path exploration. We plan to evaluate this approach by making poisoned announcements and measuring convergence times. Our work is orthogonal to efforts to reduce convergence effects [11], which we would benefit from.

**Advertise a less-specific sentinel prefix.** We propose using a *sentinel prefix* for testing reachability. When the production prefix experiences problems, the operator continues to advertise the sentinel with the baseline (unpoisoned) path. By sending active ping measurements from this prefix to destinations that had been unable to reach the production prefix prior to poisoning, we can detect when to revert the production prefix. This technique relies on two assumptions: first, before any poisoning, ASes will choose the same paths to reach both prefixes; and, second, a problem affecting one prefix will also affect the other prefix. We can test these experimentally.

Aggregation, address availability, and other concerns influence the choice of sentinel. We will explore these issues in the future. For now, we assume the sentinel is a less-specific prefix containing both the production prefix and a prefix that is not otherwise used. Responses to pings from the unused portion of the sentinel will route via the sentinel prefix, regardless of whether the hops also have the poisoned more-specific prefix. Further, ASes that do not learn of the poisoned path, because they are "captive" behind *A*, will receive the less-specific prefix and can continue to try routing to the production prefix on it, through *A*, instead of being cut off. This approach may also be helpful in cases where some paths through *A* fail while others work.

In such cases, *O* may also be able to use selective advertising to influence which paths within *A* are used and steer traffic onto working paths. If *O* has multiple providers that transitively connect to *A* at two different points-of-presence (PoPs), *O* can poison *A* in advertisements to one provider, but announce an unpoisoned path through the other provider. *A* will only accept the unpoisoned path, which it will only receive at one of its PoPs. So, *A* will route all traffic to *O*'s prefix to egress via that PoP, influencing which routes are used inside *A*.

## 5. PRELIMINARY EVALUATION

To improve availability, poisoning should restore connectivity to networks that lack it and not cut off networks that have working paths. We evaluated how effective poisoning might be using three initial experiments. First, we connected to a BGP-Mux at Georgia Tech [3] to announce poisoned BGP prefixes, poisoning a different transit AS in each of 62 experiments. We investigated whether RouteViews peers [18] had post-poisoning routes. In all cases but one, peers had paths, unless the peer was the poisoned AS. In that one case, after poisoning Verizon, we learned that Cogent seems to drop paths that contain Verizon, presumably due to how their peering is implemented, and some peers were unable to find a path without being able to route through Cogent or Verizon.

Second, to demonstrate that we can use poisoning to reroute traffic around a target AS, we used the BGP-Mux to announce a BGP prefix and receive traffic to the announced prefix via Georgia Tech. We first announced an unpoisoned /24, waited for BGP to converge, then issued traceroutes to the prefix from over 400 PlanetLab hosts. Of those, 80% reached Georgia Tech via Internet2, 8% reached through Hurricane Electric, 3% reached via Qwest, and the rest were split across a range of networks.[1] We then poisoned Internet2 and reissued the traceroutes. All but two hosts were able to reach Georgia Tech after the poisoning. Testing revealed that these two hosts have trouble reaching other sites via the commercial Internet, regardless of poisoning. Of the rest, 63% reached via Hurricane Electric, 26% used Qwest, and none used Internet2 – we were successfully able to redirect traffic around the "problem."

As this experiment suggests, if we accurately pinpoint which AS *A* to poison, other networks that had been trying to route via *A* will explore alternatives, if they have any. However, some ASes may only have paths through *A* and be "captive" behind it [4].

For our third experiment, we analyzed a large AS topology to garner some insight into how often alternate paths exist. The topology, along with AS relationships, is from a dataset combining public BGP feeds with more than 5.5 million AS paths from traceroutes between BitTorrent peers [6]. For each AS path with length greater than 2 (i.e., traversing at least one transit AS), we iterated over all the transit ASes in the path. For each such AS *A*, we simulated poisoning *A* by removing all its links from the topology. We then checked if a valley-free path [9] still existed between the source and destination; that is, were they able to restore connectivity while avoiding *A*?

We found that an alternate path existed in 76% of the 10 million simulated poisonings. For the other cases, in which no policy-compliant alternative was found, more than half involved poisoning a tier-1 network or similarly large AS (e.g., Comcast, Telefonica). At least one of these ASes appears in most paths, and they could be successfully poisoned in 87% of cases. In the remaining cases, no valley-free path existed after removing the large AS. Our results from our current failure isolation deployment (§4.2) indicate that failures in such ASes are quite rare – despite appearing in most paths, such networks accounted for only 20% of the measured outages. Excluding these ASes, we found alternate paths for nearly 90% of simulated poisonings. In practice, we can preemptively determine whether poisoning will work against a particular AS. When the production prefix is not

[1] Possibly due to anomalies in traceroute or IP-to-AS mapping.

experiencing problems, the operator can poison an AS for the sentinel prefix, then test reachability with ping.

Our simulation study has some limitations. We may miss alternate paths because (i) the valley-free assumption may be too strict,[2] and (ii) many rarely used backup paths are likely not in our topology. Similarly, we may identify valley-free paths that are not used in practice. Despite these caveats, this initial result shows that poisoning can be used safely and effectively in most cases.

## 6. RELATED WORK

Previous research used poisoning as a measurement tool to uncover hidden network topology [7] and to assess the prevalence of default routes [5]. While inspired by this work, ours differs in that we propose using poisoning operationally as a means to improve Internet availability.

Ongoing work seeks to verify the origin of BGP announcements [19]. By allowing an AS to poison only prefixes it originates, our approach is consistent with that goal. Proposals to verify the entire path [2] are also consistent with our general approach, if we consider the poison as a (validated) hint from the origin AS to the rest of the network that a particular AS is not correctly routing its traffic. By the time such proposals are deployed, it should be feasible to develop new routing primitives or standardized BGP communities to accomplish what we currently do with poisoning.

In the absence of a practical solution to long-lasting, partial routing outages, researchers have proposed and some commercial services have been developed to detour traffic around outages using overlay paths. Some solutions find detour paths by using all-pairs path monitoring [1], whereas others use random selection of alternative paths at the time of failure [10]. These approaches provide a great alternative when no other solutions exist. In contrast to detouring, our poisoning approach does not require the expense of maintaining an overlay and can carry traffic at core Internet data rates along policy-compliant BGP paths that avoid the identified problem.

## 7. CONCLUSION

The Internet was designed to provide connectivity between endpoints whenever a policy-compliant path exists. Unfortunately, availability problems are frequent, undermining the use of the Internet for cloud computing and other critical services. Using EC2-based measurements, we found that long-lasting routing problems contribute much of the unavailability, motivating a fresh approach to addressing these problems.

In our view, if an AS in the middle of the Internet announces a route to an edge network but fails to deliver packets along the route, then the edge network should have a way to cause the Internet to reroute around the AS causing the problem. To make this practical, we need to be able to accurately locate which AS is causing the failure, disable paths through the AS and trigger rerouting around it, and know when the problem has resolved so that we can revert to normal routing. We proposed an approach, Machiavellian routing, that accomplishes these steps on today's Internet using BGP poisoning. Our initial evaluation is promising: we were able to successfully poison an AS in the wild, and we demonstrated that Machiavellian routing would result in alternate paths in most cases, with minimal impact on working routes.

[2]It is well known that not all paths are valley-free in practice, and we observed violations in the BitTorrent traceroutes.

## Acknowledgments

We gratefully acknowledge the anonymous HotNets reviewers for feedback. We thank Nick Feamster and Vytautas Valancius for their feedback on this work and for allowing us to use their BGP-Mux system. The BGP-Mux system would not be available without the support of researchers and network administrators at the Mux sites. This work was funded partially by Google, Cisco, NSF grants (CNS-0905568 and CNS-1040663), and the NSF/CRA Computing Innovation Fellowship program. We are thankful for their support.

## 8. REFERENCES

[1] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. In *SOSP*, 2001.
[2] R. Austein, S. Bellovin, R. Bush, R. Housley, M. Lepinski, S. Kent, W. Kumari, D. Montgomery, K. Sriram, and S. Weiler. BGPSEC protocol. http://tools.ietf.org/html/draft-ietf-sidr-bgpsec-protocol.
[3] BGPMux Transit Portal. http://tp.gtnoise.net/.
[4] M. A. Brown, C. Hepner, and A. C. Popescu. Internet captivity and the de-peering menace. In *NANOG*, 2009.
[5] R. Bush, O. Maennel, M. Roughan, and S. Uhlig. Internet optometry: assessing the broken glasses in Internet reachability. In *IMC*, 2009.
[6] K. Chen, D. R. Choffnes, R. Potharaju, Y. Chen, F. E. Bustamante, D. Pei, and Y. Zhao. Where the sidewalk ends: Extending the Internet AS graph using traceroutes from P2P users. In *CoNEXT*, 2009.
[7] L. Colitti. *Internet Topology Discovery Using Active Probing*. PhD thesis, University di "Roma Tre", 2006.
[8] ec2. `http://aws.amazon.com/ec2/`.
[9] L. Gao. On inferring autonomous system relationships in the Internet. *IEEE/ACM TON*, 2001.
[10] K. P. Gummadi, H. V. Madhyastha, S. D. Gribble, H. M. Levy, and D. Wetherall. Improving the reliability of Internet paths with one-hop source routing. In *OSDI*, 2004.
[11] J. P. John, E. Katz-Bassett, A. Krishnamurthy, T. Anderson, and A. Venkataramani. Consensus routing : The Internet as a distributed system. In *NSDI*, 2008.
[12] E. Katz-Bassett, H. V. Madhyastha, V. K. Adhikari, C. Scott, J. Sherry, P. van Wesep, A. Krishnamurthy, and T. Anderson. Reverse traceroute. In *NSDI*, 2010.
[13] E. Katz-Bassett, H. V. Madhyastha, J. P. John, A. Krishnamurthy, D. Wetherall, and T. Anderson. Studying black holes in the Internet with Hubble. In *NSDI*, 2008.
[14] N. Kushman, S. Kandula, and D. Katabi. R-BGP: Staying connected in a connected world. In *NSDI*, 2007.
[15] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. Delayed Internet routing convergence. In *SIGCOMM*, 2000.
[16] K. K. Lakshminarayanan, M. C. Caesar, M. Rangan, T. Anderson, S. Shenker, and I. Stoica. Achieving convergence-free routing using failure-carrying packets. In *SIGCOMM*, 2007.
[17] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane: An information plane for distributed services. In *OSDI*, 2006.
[18] D. Meyer. RouteViews. `http://www.routeviews.org`.
[19] P. Mohapatra, J. Scudder, D. Ward, R. Bush, and R. Austein. BGP prefix origin validation. http://tools.ietf.org/html/draft-ietf-sidr-pfx-validate.
[20] Outages mailing list. `http://isotf.org/mailman/listinfo/outages`.
[21] UCLA Internet topology collection. `http://irl.cs.ucla.edu/topology/`.
[22] W. Xu and J. Rexford. MIRO: Multi-path interdomain routing. In *SIGCOMM*, 2006.