Machiavellian Routing: Poisoning BGP for Route Control

Ethan Katz-Bassett* Thomas Anderson*

1 Introduction

Content and cloud providers place a priority on high performance, reliable paths for their networks, and they want to react quickly and effectively when problems arise. Yet, in conversations with us, major providers express frustration that operators spend much of their time in triage of wide-area problems and are often stuck with the same tools they have been using for a decade or more. For example, it is common for operators to post to mailing lists when confronted with outages, loss, or degraded performance, asking others to issue traceroutes and otherwise weigh in with their view of the problem. Operators often have little visibility outside their local network, so may have little idea of the extent of a problem, its location, or its root cause. Further, in many cases, even with a precise and accurate idea of where a problem is, if it is outside the local network, the operator lacks mechanisms to do much about it. With the mechanisms that do exist, it can be difficult to predict the cascading effects of any changes, leading to a reluctance to make changes.

In our work, we propose to conduct GENI-based Internet measurements investigating new techniques operators could use to locate and avoid problems. This work will build upon earlier systems we developed in this vein, including Hubble [5], WhyHigh [6], and reverse traceroute [4].

2 Approach

With WhyHigh, we found that Google, with its stature, often has the contacts and clout to convince another network to modify configurations in response to problems, and so a central goal was to identify which network to contact for a particular problem. However, this remediation approach of using human-human contact does not always work. Smaller providers may not have the clout or contacts necessary. Some networks, even contacted by Google, might not respond or might lack the know-how or resources to respond properly or in a timely fashion. In these cases, we believe that the provider must exercise control of how traffic routes to and from it, to avoid performance and connectivity problems. Our previous work with Hubble suggests that it should often be possible to route around failures. First, we found that, during most cases of long-term reachability problems, the destination remained reachable from some vantage points, establishing that working paths often exist around the failures. Second, we found that most failures were unidirectional, with the path in one direction between the two endpoints working even while the other was down, and that we could often isolate the direction of failure. Since the path in one direction should often be a policy-compliant option in the other direction, this result suggests that working paths often exist even between endpoint pairs with a currently failed path. Because paths often exist around failures and methods based on our earlier work [5, 4] should allow us to locate these failures, we intend to investigate techniques to allow ASes to route around them.

Consider an operator at a content network with multiple data centers (DCs) who suddenly finds that a particular client prefix cannot talk to a particular service at one of the DCs. Using techniques based on our reverse traceroute system, the operator isolates the direction of failure, the location of the failure, and the scope of the failure (which DCs cannot reach the clients). If the problem is on the path from the DC to the clients, the operator has several options to try to avoid the failure. First, she can choose a path to the clients through a different provider. Second, a different DC can serve the clients, either via detouring through it on paths from the original DC or via DNS redirection of the client to the working DC. However, these options do not apply for problems on the path from the client to the DC, as the operator cannot choose which path the client uses, and failures on paths to one DC may affect the other DCs as well (since they might receive addresses from a common super-prefix). Instead, the operator can exert influence over the paths used to reach the DC by changing the prefix advertisement. We plan to explore three ways of modifying the announcement. First, by selectively announcing the prefix only to certain providers, the operator guarantees that (non-default) paths will not reach through an unannounced provider, limiting the damage done by problems specific to that provider. Second, the DC can prepend on announcements to a subset of its providers, making those paths less

*Dept. of Computer Science, Univ. of Washington, Seattle.

preferred. Third, the DC can "poison" specific ASes known to be causing problems. Poisoning involves inserting the ASN of a particular network into the path announcement [3]. When that network receives the announcement, it will reject it to avoid causing a loop, resulting in no other ASes actually selecting (non-default) paths through that AS.

3 Testbed

In this section, we discuss the properties required of a testbed if we are to study techniques for reacting to wide-area routing problems, the difficulties involved in obtaining such a testbed, and the suitability of GENI. In order to evaluate such systems on the actual Internet, we need a testbed with the following properties:

Joint view into data and control planes: The testbed must provide visibility into network routing at both the data and control planes, from the same location. Although it is possible to monitor the data plane from PlanetLab and the control plane from RouteViews, such a setup would include different vantage points for data plane versus control plane, and this decoupling makes it difficult to accurately map changes on one plane onto changes on the other plane.

Multiple locations: The testbed must allow us to simulate having multiple data centers. Because we are primarily interested in problems that involve multiple geographically distributed data centers, our experiments must exercise such a setting. Therefore, even if we were able to convince our campus network operators to provide us with a testbed that met the other requirements, the experiments would be unrealistic and unlikely to suffice.

Multi-homed: The testbed must allow us to simulate multi-homing. Because any real-world data center typically has multiple providers, it is important that our testbed includes this property. Some potential approaches to detecting and mitigating the problems of interest likely involve taking advantage of the availability of multiple providers. Because a university typically connects to the Internet through a single primary provider, even experiments that achieve multiple locations via multiple universities will not satisfy our needs without a seamless way to use the providers of all universities from any one of them.

Routing control: The testbed needs to allow us to control routing, both in terms of outgoing routes selected and in terms of the routes announced for prefixes used in the experiments. This requirement likely necessitates access to precious IP address space, a direct BGP feed from multiple ASes, and the ability to announce the address space in varying ways. Therefore, even a solution that utilizes multiple universities with tunneling between them to simulate multi-homing will not suffice if it relies on the routes chosen and announced by the universities.

We believe that the difficulty in attaining a testbed that satisfies all four properties has stifled research in this area. In particular, as academic researchers, we have thus far been unable to perform the types of experiments necessary to build and evaluate the types of systems we are interested in. The main people with ready access to testbeds similar to what we require are those at companies with multiple data centers, leading to interesting research in this area from Google [6] and Microsoft Research [11].

GENI opens up the possibility for academic researchers to conduct vital work in this area. In particular, BGP-Mux [10] provides the properties we desire. By combining routing views from multiple institutions, some of which have multiple providers, and by coupling easily with Amazon EC2, it allows for the simulation of distributed data centers and multiple providers and for experiments that jointly exercise both the data and control planes. By allowing route advertisements and outgoing route selection, as well as by providing IP address space, it allows for experiments that involve changing routing.

4 Proposed Experiments and Measurements

In this section, we describe GENI-based experiments to investigate our proposals for how data center operators can troubleshoot and mitigate wide-area routing problems. Although our broader agenda includes researching techniques to anticipate and preempt developing problems, to locate problems, to identify their root causes, and to avoid problem areas using a range of techniques, we focus here on experiments with path poisoning, We do this because we plan to undertake these experiments in the near future and because GENI enables us to conduct them when we could not easily otherwise. We have spoken with two of the researchers who developed BGP-Mux, and they agree that our ideas are a good fit for their system. Our experiments use our earlier iPlane, Hubble, and reverse traceroute systems, all of which use federated PlanetLab / OneLab vantage points for their active probes.

We propose a set of experiments to evaluate how effectively we can avoid problems in the network through BGP poisoning. In this setting, a data center announces a prefix p and wants paths for p to avoid using AS a. In order for operators to consider adopting this fairly aggressive approach to route control, we need to demonstrate that the technique is: *effective*, allowing networks to avoid a without cutting them off completely; *non-disruptive*, with paths converging quickly after the poisoned announcement, without causing widespread loss during the convergence period; and *predictable*, allowing operators to understand the likely impact of poisoning a before they do so, as well as giving a reliable signal when it is safe to revert to the baseline announcement. These experiments require a testbed that gives a joint view into the control and data planes and that allows for routing control. In our experiments, we will announce a prefix p via BGP-Mux, then, using reverse traceroute, measure the paths to p from one pingable address n in each of a large set of prefixes N. We will then consider each transit AS a on those paths in turn and announce p with a prepended. We will assess the important properties of the technique as follows:

Effectiveness: For each pingable address n that previously used a to reach p, we will test whether it is still pingable from p and, if so, whether its path contains a, despite our efforts to avoid a. In general, n should have a path to p if it is not captive behind a [2], and its path should not include a as long as it is not using a default path to a [3].

In our followup experiments, we will evaluate the effectiveness of BGP poisoning at mitigating the effects of real failures. We will use Hubble to identify actual outages affecting the BGP-Mux prefix, use reverse traceroute to identify the AS containing the failure, then measure how well we can route around the problems by poisoning that AS. We will measure the effectiveness by measuring whether reachability of p improved after the poisoning.

Disruptiveness: To evaluate the disruptiveness of our poisoning scheme, we will measure, following a poisoning, convergence time on the control plane and loss on the data plane. We will determine convergence time by inspecting RouteViews feeds, and we will measure loss by pinging destinations in N from p regularly during the convergence period. We note that we are mostly concerned with loss experienced by destinations that were not using a before the poisoning; presumably, operators would only resort to poisoning in situations where paths through a were not working at all or were performing so poorly that impacting them during convergence would not be a major concern. Then, after unpoisoning a and allow paths to revert, we will assess whether we can reduce the disruption as follows. First, we will re-announce p, prepending the BGP-Mux's ASN b an additional time. Once paths have converged, we will replace the extra b with a. Our hope is that networks not using paths through a will equally prefer the prepended and the poisoned path, smoothly the convergence period.

Predictability: After evaluating the effectiveness and disruptiveness of poisoning paths to p, we will test two approaches to predicting the effects of poisoning a. First, we will evaluate using a test prefix p'. We will start by announcing p' from BGP-Mux with an announcement identical to that for p. Then, like our tests for p, we will poison each transit AS a for p' in turn, assessing which addresses in N remain reachable and do not use paths through a. We can then compare these results to those for p to make sure there are no prefix-specific anomalies. Assuming the prefixes behave the same, as we would expect, an operator announcing p in a real deployment could allocate a prefix for testing purposes. Then, he could use the second prefix p' to, when paths were working, preemptively explore the effects of poisoning different ASes, to know in advance what will happen if a particular a starts causing problems for p. Second, we will evaluate how accurately we can predict the effects of a poisoning without using a test prefix. In particular, we will use our system iPlane's path predictions [7] to predict what paths will be chosen given a topology without a, then compare these predictions to the actual paths chosen after poisoning a.

Just as we hope to use a test prefix to predict the outcome of poisoning a, we hope to use it to determine when it is safe to stop poisoning. When forced to poison a for p, the operator can revert to the non-poisoned announcement for p' and regularly assess reachability of p' to decide when it is safe to unpoison a for p. We will evaluate this approach using problems identified by Hubble, as with the second round of effectiveness experiments.

5 Old Approach

We first give an overview of our planned approaches, then discuss experiments we plan, focusing on how using GENI will allow us to develop and evaluate our techniques.

Monitoring router health: Network operators are often unaware of developing problems in their networks until they cause widespread loss or disconnectivity. We may be able to preemptively detect problems by detecting anomalies in the number of packets a router is sending, using them as distress signals. Many routers use an incrementing counter to set the IPID field on packets they source [1]; for a router that does, we can remotely monitoring the rate at which it is sourcing packets by periodically probing it and observing the rate of change of the IPID values in the responses. Unexpected changes in this rate may signal developing problems at the router. For example, in preliminary hand investigations, we observed events including BGP session resets, sudden bursts of TTL expirations, and large numbers of packets destined to unreachable destinations. When we brought the TTL expirations to the attention of an operator responsible for the router, he indicated that he would find it useful to have a live system flagging such events as they occurred, to allow for investigation. We plan to research whether we can develop predictive models of healthy packet-sourcing behavior, as well as signatures of particular anomalies, in order to deploy low cost IPID-based monitoring.

Locating performance problems and outages: Operators' mailing lists contain frequent threads in which operators query each other and issue measurements by hand to try to determine the extent and location of performance problems and outages [9, 8]. This approach is not very scalable and not very accurate. In our earlier work, we built systems that provide preliminary steps towards providing better problem localization. Our system Hubble identifies cases of partial and complete unreachability, as well as monitoring the problems over time [5]. It also provides simple classifications of where the problems seem to originate and can isolate failures to either the forward or reverse path. In collaboration with Google, our system WhyHigh provided mechanisms to identify Google client prefixes experiencing inflated latencies and, in cases of forward path problems, techniques for identifying the likely causes of inflation [6]. We believe we can use our reverse traceroute system to more accurately and precisely locate both the outages tracked by Hubble and the performance issues targeted by WhyHigh. First, we found that most disconnectivities are unidirectional [5], and we can use techniques from reverse traceroute to measure the complete path in the working direction (not possible with traceroute alone, as either the source will not be able to send a packet to the destination, or it will not be able to receive the response) and to isolate the failed link in the other direction. Second, a major limitation in the WhyHigh work was that Google lacked visibility into the paths from clients back to Google data centers, restricting much of their diagnosis to forward path problems. In the reverse traceroute paper, we provided an example of using the system to diagnose the cause of reverse path inflation. Third, we developed a technique, using reverse traceroute, to measure the one-way latency of individual backbone links, a potentially important step in identifying the causes of slow performance. We plan to evaluate the effectiveness of the technique for pinning down a failure by evaluating over Hubble data, and we are working with Google to investigate incorporating reverse traceroute into their operations.

Identifying the root cause of path changes: In conversations with us, operators at major cloud and web service providers revealed that a large portion of their time is spent on wide-area network triage and break fix – network interruptions occur outside their network, and no major new tools have emerged to help. As we learned in the WhyHigh work, it is very helpful just to point operators towards which issues to tackle (because they are hurting performance and better alternatives exist) and which ASes are responsible. While WhyHigh focused on prefixes that had chronically bad performance, we have started working to apply this lesson to sudden path changes that cause degraded performance. The problem is challenging because routing changes at a given AS can have cascading effects even on distant networks – the AS responsible for a path change between a provider and its client need not be on either the old or new path between them. However, we have proved that the responsible AS must either be on the new path of an AS that used to be on the route between provider and client, or it must be on the old path of some AS that is now on the route. This fact has two important implications for monitoring path changes. First, in order to measure the new path of ASes that used to be on the route, we must be able to measure paths from arbitrary locations, not just from the provider and client. Second, in order to measure the old path of some AS newly on the route, we must have anticipated that the client might reach the provider through that AS in the future. Two of our previous systems should give us these abilities. Reverse traceroute lets us measure paths from arbitrary locations, and iPlane predicts possible paths between arbitrary source / destination pairs [7], allowing us to predict which routes might be used in the future. We plan to design a technique, based on these earlier systems, to monitor and locate the causes of path changes.

References

[1] A. Bender, R. Sherwood, and N. Spring. Fixing Ally's growing pains with velocity modeling. In *IMC*, 2008.

[2] M. A. Brown, C. Hepner, , and A. Popescu. Internet captivity and de-peering. In *NANOG 45*, 2009.

[3] R. Bush, O. Maennel, M. Roughan, and S. Uhlig. Internet optometry: assessing the broken glasses in Internet reachability. In *IMC*, 2009.

[4] 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.

[5] 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.

[6] R. Krishnan, H. V. Madhyastha, S. Srinivasan, S. Jain, A. Krishnamurthy, T. Anderson, and J. Gao. Moving beyond end-to-end path information to optimize CDN performance. In *IMC*, 2009.

[7] 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.

[8] Outages mailing list. `http://isotf.org/mailman/listinfo/outages`.

[9] `http://www.merit.edu/mail.archives/nanog/`. North American Network Operators Group mailing list.

[10] V. Valancius, N. Feamster, J. Rexford, and A. Nakao. Wide-area route control for distributed services. In *USENIX Technical*, 2010.

[11] Z. Zhang, M. Zhang, A. Greenberg, Y. C. Hu, R. Mahajan, and B. Christian. Optimizing cost and performance in online service provider networks. In *NSDI*, 2010.