Stabilizing Route Selection in BGP

P. Brighten Godfrey, Matthew Caesar, Ian Haken, Yaron Singer, Scott Shenker, Ion Stoica
pbg@illinois.edu, {yaron,shenker,istoica}@cs.berkeley.edu, caesar@cs.illinois.edu, haken@berkeley.edu

***Abstract*—Route instability is an important contributor to data plane unreliability on the Internet, and also incurs load on the control plane of routers. In this paper, we study how route selection schemes can avoid these changes in routes. Specifically, we characterize the tradeoffs between *interruption rate*, our measure of stability; *availability* of routes; and *deviation* from the network operator's preferred routes. We develop algorithms to lower bound the feasible points in the tradeoff spaces between these three cost metrics. We also propose a new approach, *Stable Route Selection* (SRS), which uses flexibility in route selection to improve stability without sacrificing availability, and with a controlled amount of deviation.**

Our large-scale simulation results show that SRS can significantly improve stability while deviating only a small amount from preferred routes. We implement our protocol in a software router, Quagga, and confirm in cluster deployment that SRS's gains in route stability translate to improved reliability in the data plane. Finally, we evaluate SRS under direct feeds of route update traffic from Internet routers. In this case, we observe less improvement, but SRS can still improve stability when multiple disjoint paths are available.

I. Introduction

A number of studies [10], [22], [38] point to stability as a key problem for the Border Gateway Protocol (BGP), the interdomain routing protocol which knits the fabric of today's Internet. Network failures, policy changes, and the BGP convergence process itself can generate huge numbers of routing updates, affecting both the data and control planes.

In the data plane, it is well known that end-to-end path quality is degraded by BGP route updates (see [40] and references therein). According to a recent study, the majority of packet loss bursts are caused by inter-domain route convergence problems such as transient forwarding loops, rather than by congestion [38]. These problems are increasingly important as the Internet is becoming an ubiquitous platform for voice and video applications. A measurement of VoIP calls between clients on PlanetLab showed that almost half of problems in these calls were highly correlated with BGP updates, and BGP events were estimated to cause 90% of VoIP call drops [20]. Internet games such as Counter-Strike have similar demands for interactivity, commonly sending periodic delay-sensitive bursts of packets [11].

In the control plane, route update processing incurs CPU load, a problem which has attracted a significant amount of concern [9], [23], including at the Internet Architecture Board [26]. Current demands on router CPUs appear to be feasible [19]. However, there are two ways update processing can be problematic. First, since BGP updates often arrive in bursts [19], convergence can be slowed [9], [23], which (as discussed above) worsens data plane reliability. Second, update processing could become increasingly expensive or inconvenient if future dramatic routing table growth occurs from IPv6 deployment, further IPv4 deaggregation, or unforeseen factors.

The main mechanism for improving stability in BGP is route flap damping [37], which filters routes that have a short-term update rate above some threshold. Unfortunately this seemingly simple strategy is fraught with problems. In 2002, Mao et al. [24] demonstrated that flap damping creates pathological conditions that slow convergence. Flap damping also worsens *availability*—the fraction of time that a router has a route to a destination—by occasionally shutting off *all* available routes. The operator community has become aware of these problems, with the RIPE Route Working Group advising in 2006 that "the application of flap damping in ISP networks is NOT recommended. ... With current vendor implementations, BGP flap damping is harmful to the reachability of prefixes across the Internet." [32] Other approaches to improving stability require protocol modifications [6], [24] or address narrow cases [2], [12], [16].

Thus, despite the fact that the problem was recognized more than a decade ago, there is still no compelling mechanism for stabilizing BGP routes, and key questions remain unanswered. For example, notwithstanding the fact that it can delay convergence, does flap damping provide an overall improvement in stability or not? Within the framework of the BGP route selection process, how much is it possible to improve stability, and at what cost?

This paper takes a principled approach to stabilizing Internet routing. Our high-level method is as follows:

1) We identify *general approaches* to stabilizing path-vector routing protocols such as BGP, and their *inherent tradeoffs* with other objectives.
2) We characterize how well each approach can perform by sandwiching the set of feasible points in the tradeoff spaces between upper and lower bounds.

This evaluation makes possible a more informed choice of a method to combat instability.

Characterizing the tradeoff spaces: This paper studies three general approaches to stabilizing routing: (1) reducing route convergence overhead, i.e., instability caused by BGP's distributed convergence behavior rather than by link state changes; (2) avoiding instability by preferring stable routes; and (3) avoiding instability by occasionally shutting off all routes between a particular source and destination. The latter two approaches imply tradeoffs with *availability* and *deviation* from preferred routes, respectively.

This project was supported by a Cisco Collaborative Research Initiative grant. The first author was supported in part by an NSF Graduate Research Fellowship at UC Berkeley while working on this project.

Our characterization of what can be accomplished with each of these approaches proceeds in two parts. First, we give algorithms which lower-bound the performance of *theoretically optimal strategies*, which allow us to constrain which points in the tradeoff spaces are achievable, for any given network topology and pattern of failures. To obtain numerical results, we apply these algorithms to a measured topology of around 20,000 autonomous systems (ASes) with inferred customer/provider/peer relationships and one year of inferred link failure data from Route Views [31].

Second, we evaluate the performance of *implementable strategies* including flap damping and new *Stable Route Selection* (SRS) strategies that we develop. SRS has the goal of *safely* stabilizing routing. Unlike flap damping, SRS does not reduce availability; unlike BGP's MRAI timer and Path Exploration Damping [18], SRS does not cause inconsistency by delaying updates. Instead, SRS uses flexibility in route selection to prefer more stable paths. Our evaluation of these strategies uses simulations of the BGP protocol in the same environment as our lower bounds, and experiments using a cluster-based deployment of software routers.

Results: Our lower bound techniques provide the following key impossibility results within our simulation environment:

- *Reducing convergence overhead* can only improve stability by 5-20% in our simulated environment. This conclusion is robust to the presence or absence of flap damping and the degree of heterogeneity of message propagation delay. However, bigger improvements may be possible for withdrawals by the origin AS and if AS policies do not conform to standard customer-provider-peer relationships.
- By *allowing deviation* from the operator's desired paths but preserving optimal availability, standard BGP's stability cannot be improved by more than $8.1\times$.
- By also *trading off availability*, stability might be improved by an additional $2-3\times$ to a total of $\approx 20\times$ with some downtime, but bigger improvements are not possible without substantial downtime.

In addition to these lower bounds, we evaluate the performance of implementable strategies, with the following main conclusions:

- Flap damping does improve stability, but at the significant cost of about two "nines" of lost availability with Cisco default parameters.
- Our SRS strategies preserve the high availability of BGP without flap damping, while obtaining up to $5\times$ better stability—slightly better than BGP *with* flap damping, and coming within $1.6\times$ of the theoretical optimum. Alternately, SRS can provide a $2\times$ improvement over standard BGP while deviating from preferred routes for less than 8 minutes per day, on average. Experiments with a software router deployment show that SRS's benefits translate to improved data plane reliability.

We also evaluate the version of SRS which ensures limited deviation under direct feeds of route update traffic from Internet routers recorded by Route Views [31]. In this case, we find a only a 12% reduction in interruption rate compared with BGP. One reason is that this evaluation is for deployment at a single router, rather than at all routers in the network. But we also provide evidence that the smallness of this improvement is due in part to the fact that for most prefixes, all available routes share the same final AS-to-AS link. For prefixes with multiple disjoint paths, we find an a 24% reduction in interruption rate.

The rest of this paper proceeds as follows. In Section II, we define our model of BGP, metrics, and classification of approaches to stabilizing BGP. Section III describes our algorithms for obtaining lower bounds in the tradeoff spaces, and Section IV describes the upper bounds: in particular, our proposed SRS strategies. Our simulation and experimental results appear in Sections V and VI. Our results with Route Views feeds appear in Section VII. We discuss related work in Section VIII and conclude in Section IX.

II. Preliminaries

This section introduces a simple model of BGP (Sec. II-A) and the main metrics that we study (Sec. II-B). We then set the stage for the rest of the paper by classifying approaches to stabilizing BGP and their inherent tradeoffs (Sec. II-C).

A. Model of BGP

In this section we describe a simplified model of BGP which forms the basis of our analytical results.

At a high level, the operation of a BGP router is simple: for each destination (an IP prefix), it learns advertised routes from its neighbors; it selects one neighbor's route to use, according to some route selection policy; and according to some export policy, it may subsequently advertise this selected route, as well as its own local destinations, to other neighbors.

Our model adopts those general rules: We are given some fixed destination D. A route is specified as a sequence of nodes along a path to D. Each router R at any given time has *selected* either one route to D, or no route; and may be *advertising* this selected route its neighbors.

Additionally, our model includes two important constraints. First, we assume that message propagation and routing decisions take a negligible amount of time, relative to the time between link state changes. This condition, which we refer to as **batching**, effectively partitions time into epochs during which all routes are fixed, punctuated by instantaneous "batches" of convergence events. Typically, these events are triggered by link state changes in the underlying topology, but they may also occur if routers decide to change their selected paths after some non-negligible period of time, as in flap damping and some versions of our SRS strategy. Batching enables us to obtain bounds on the optimal policies. Although batching does affect the system, we will observe similar performance with and without batching in our simulations of Section V.

The second condition we impose is **path consistency**. We say paths are consistent at a given moment if for each router v_1 that has currently selected some path $v_1, \ldots, v_k$, the following are true: (1) all links (v_i, v_{i+1}) are up, (2) v_2 selected the path $v_2, \ldots, v_k$ and is advertising this path to v_1, and (3) the ultimate node v_k is the destination D. We require that path consistency holds at any time, except during convergence batches. Modulo timing differences, any classic path vector routing protocol, BGP included, attempts to satisfy path consistency. However, in Section II-C we discuss several strategies which result in inconsistency.

Subject to the conditions of path consistency and batching, routers may follow any route selection and export policies. Our numerical results will use a range of policies based on inferred real-world business relationships.

B. Metrics

In this section we define our three main metrics: interruption rate, availability, and deviation.

An **interruption** is the event that the path selected by some AS changes or is withdrawn entirely (i.e., a transition to the disconnected state). We do not count recovery events as an interruption. We use interruption rate, which measures instability, as a proxy for data plane performance and control plane CPU utilization due to its computational and analytical tractability. Our experimental results in Section VI will correlate interruption rate with packet loss.

We define **availability** for a particular source-destination pair as the fraction of time that the source has a route to the destination. We will typically study the mean availability over all source-destination pairs.

Finally, **deviation** compares a sequence S of selected paths against a *benchmark sequence* of paths S^*, and is defined as the fraction of time that the route in S "matches" the route in S^*. Deviation is intended to capture how closely a particular route selection strategy (S) follows the network operators' preferred paths (S^*). Since measurements show that ASes' routing preferences are based on next-hops for 98% of IP prefixes [39], we say that two routes from a particular source "match" when their next-hops are equal. Our simulations will take S^* as the paths selected by the standard BGP decision process; thus, our numerical results measure how much various strategies differ from the status quo. As with availability, we study the mean deviation over all source-destination pairs.

C. Approaches to Stabilizing BGP

Within our model, it is possible to give a complete classification of approaches to reducing interruption rate relative to standard BGP.

At a high level, we have a simple choice: pick the same sequence of paths as standard BGP—except during the instantaneous convergence events—or pick paths which differ. These two cases respectively imply that we must either (1) *mitigate the impact* of topology or policy changes by improving the reconvergence process, or (2) *avoid* reconvergence events altogether. Within the avoidance approach, there are two pure approaches: (2a) select paths that fail less often, and (2b) select *no path*, disconnecting the source from the destination. These approaches are summarized in Fig. 1.

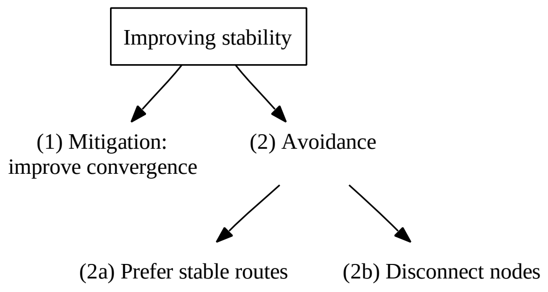

Improving stability
(1) Mitigation: improve convergence
(2) Avoidance
(2a) Prefer stable routes
(2b) Disconnect nodes

Fig. 1: Approaches to stabilizing BGP.

These three approaches require qualitatively different sacrifices ranging from free to severe. Approach (1) is the most attractive because it improves stability without compromising other objectives. The two remaining approaches directly imply tradeoffs: (2a) results in nonzero deviation, and (2b) is an extreme approach which sacrifices availability. In the limit, a network where all nodes are disconnected has no interruptions, but also has zero availability!

Note that (2b) is not equivalent to RFD's strategy of shutting off (damping) unstable routes. Damping a route causes BGP to select another route, as long as an alternate un-damped route is available. Thus, RFD mixes approaches (2a) and (2b).

Our characterization of the tradeoff spaces places limits on what can and can't be accomplished with these three approaches. We begin the characterization in Sec. III with our lower bounds algorithms. Our upper bounds, i.e., implementable strategies, include our new SRS strategies described in Sec. IV in addition to Standard BGP and RFD. The numerical results of these lower and upper bounds appear in Sec. V.

Strategies outside this classification. Note that policies which allow path inconsistency (defined in Sec. II-A) are implementable in today's BGP but fall outside our model. Such strategies are beyond the scope of this work, and may be useful; however, we note that as a result of their path inconsistency, they must be handled with care, since there is the possibility that routing loops or disconnectivity can persist for non-negligible periods of time.

To the extent that a 30-second time period is considered non-negligible, an example of a path-inconsistent strategy is the commonly-used Minimum Route Advertisement Interval (MRAI) timer. The MRAI rate-limits update messages to each neighbor of a router to one per 30 sec. Recently, Huston [18] proposed delaying updates for just longer than an MRAI interval, 35 sec, when they match a pattern likely to indicate BGP path exploration. Consecutive matches could delay updates for minutes or more.

III. Lower Bounds

How much can the above approaches reduce the rate of interruptions, and what are the associated tradeoffs with deviation and availability? To answer that question, we numerically *lower bound* the best possible interruption rate for given amounts of deviation and unavailability.

There will be a gap between the interruption rate of implementable strategies and that of our lower bounds for two reasons. First, the theoretical optimal strategy has knowledge of the future pattern of route failures. Second, we show that computing the optimal interruption rate is NP-hard, so we must resort to lower-bounding the optimal.

In Section V, we will apply these lower-bounds procedures to the measured topologies and traces used in our simulator, allowing us to compare how close our proposed strategies are to the best possible policies.

A. Inputs and outputs

The procedures take as input an AS-level topology annotated with customer-provider-peer business relationships between ISPs, which they honor; a trace of AS adjacency ("link") failures; and a sequence of preferred route selections over time for each source/destination pair.

Given this input, we will find the following:

- **Approach (1): Convergence.** We find the minimum number of interruptions required to adhere to the given preferred routes *almost always*, i.e., at all times except for negligible periods of time during convergence events (formally, a set of times of zero measure).
- **Approaches (1)+(2a): Stability-Deviation tradeoff.** We compute a set of *undominated points* (x_i, y_i) such that for each i, in the given topology and traces, it is impossible to select routes that achieve both a mean interruption rate of $< x_i$ and a mean deviation of $< y_i$, while preserving the highest possible availability. Means are over all sources for a given set of destinations (which will be a random set when we generate results in Section V).
- **Approaches (1)+(2a)+(2b): Stability-Availability tradeoff.** We compute a set of undominated points (x_i, y_i) such that for each i, it is impossible to select routes that achieve *both* a mean interruption rate of $< x_i$ *and* a mean unavailability of $< y_i$, with no constraint on deviation.

The first item, bounding convergence overhead, appears in Section III-B and is straightforward. In contrast, it is nontrivial to obtain good lower bounds in the tradeoff spaces. We explain why (Section III-C) before describing the procedure (Section III-D) which is similar for the two tradeoff spaces.

B. Convergence

For Approach (1), we must find the minimum number of interruptions achievable solely by improving the convergence process—in other words, with zero deviation and zero unavailability. Batching, described in Section II-A, allows us to easily compute this value. Given a fixed setting of each router's converged state before and after each batch, there must be at least 1 interruption for each batch in which the route changed or was withdrawn, and at least 0 if the path stayed the same. We simply sum these over all batches to produce the desired lower bound for each source-destination pair.

C. Hardness of Minimizing Interruptions

Section III-B showed that it is easy to find the optimal interruption rate for one particular point in the tradeoff spaces: zero unavailability and zero deviation. But in general, computing the optimal is difficult because of *dependencies between nodes* when routing to some particular destination.

We illustrate this with an example for another particular point in the tradeoff spaces: that of minimizing interruption rate under the constraint of zero unavailability (but any amount of deviation from preferred routes). Consider the topology of Fig. 2. Groups 1 and 2 consist of n and m ASes, respectively, which depend on routing through $AS1$ to the destination. Suppose the routes $R1$ and $R3$ are available during the time interval $[0, 10]$, while $R2$ and $R4$ are available during $[5, 15]$. Sometime during $[5, 10]$, $AS1$ must switch from $R1$ to $R2$. But this affects Groups 1 and 2: if $AS1$ switches at time 5, it triggers an interruption at each of the n nodes in Group 1; if it switches at time 10, it interrupts the m nodes in Group 2. The optimal routes for $AS1$, in terms of the cost to the network as a whole, therefore hinge upon whether $n < m$. We thus have nonlocal dependencies between nodes' route selections.

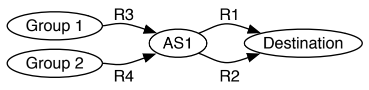


Fig. 2: Example AS topology.

In fact, it is possible to show that the problem of minimizing the number of interruptions is NP-complete, using a gadget similar to the above as one step of a reduction from SAT. The proof appears in the Appendix.

D. Tradeoff Procedure

Since it is difficult to find the exact minimal number of interruptions, we resort to lower-bounding the optimal. Specifically, we allow each node to independently select its own route, thus possibly picking a route which was not chosen by the downstream ASes along the path. This relaxation of path consistency will allow us to compute optimal tradeoff values for each (source, destination) pair separately; we then assemble these pieces together to produce an undominated point in the tradeoff spaces.

We first describe two important subroutines which we then use in our final procedure.

Optimal Path Sequence (OPS) subroutine: This core "inner loop" used by our tradeoff calculations computes the optimal sequence of path selections for a single node over time. Specifically, OPS is given a set of route options. Each option is of the form (r, t_1, t_2, c), meaning route r is available during $[t_1, t_2]$ and incurs cost c per unit time that it is selected. The cost of causing an interruption (see Sec. II-B) is fixed at 1. OPS computes the minimum cost sequence of route selections over time for the given options.

This computation is equivalent to a shortest paths problem on an appropriately constructed abstract graph whose nodes represent route choices and whose edges represent legal transitions between them, respectively. Our implementation uses a somewhat more efficient dynamic programming algorithm which iterates through time, storing the least-cost way to reach each currently available path choice.

Most Stable Paths subroutine: This subroutine computes a set of potential routes that will later be fed into the OPS subroutine. Specifically, it calculates, for each source s, destination d, and time t, *the available $s \rightarrow d$ path which will be available starting at t and continuing farthest into the future.* Given knowledge of the future availability of any given route, this can be computed *en masse* for all sources and a particular destination using a BGP-like algorithm whose path selection prefers routes that will be available longer.

Computing the future availability of any route involves examining future link failure times, which are easily obtained from the trace provided to the lower bounds procedure. However, it also involves a complication: the future failure time of a route is dependent on future changes in the business class of the route selected by each AS on the path. For example, if a downstream AS switches from a customer route to a peer route, it will no longer export the route to other peers or providers. We calculate business class switch times by running the route selection simulation twice, recording the business class switch times on the first trial, and using them to compute paths' future failure times in the second. It can be shown, along the lines of the proof of convergence in [12], that the business class switch times will be identical in both trials.

Putting it all together: We now describe how we compute a set of undominated points (i.e, a Pareto set) in the tradeoff spaces, utilizing the above subroutines. We begin with the stability-availability tradeoff.

Let $R = \{r_{sd}\}$ be a sequence of route selections for each source-destination pair (s, d), and let $intr(\cdot)$ and $down(\cdot)$ represent the number of interruptions and the amount of downtime, respectively, in R. Our goal is to produce R's for which the point

$$(intr(R), \quad down(R))$$

is undominated. To do this, we we begin with the well-known weighted sum method of multiobjective optimization, as follows. We introduce a parameter λ which intuitively sets the cost of being disconnected per unit time. We next describe how to produce a single undominated point given λ; we vary λ to produce multiple points.

A straightforward application of the weighted sum method would then find the optimal feasible value R^* of R which minimized $intr(R) + \lambda \cdot down(R)$. However, as we showed in Sec. III-C, computing R^* is hard. We instead optimize each source-destination pair separately. Specifically, for each source s and destination d we find (using a procedure to be described below) the valid route assignment ℓ^*_{sd} which minimizes

$$intr(\ell^*_{sd}) + \lambda \cdot down(\ell^*_{sd}). \tag{1}$$

We then construct the network-wide route assignment $L^* = \{\ell^*_{sd}\}$ and finally return the undominated (though potentially infeasible) point $p = (intr(L^*), down(L^*))$. We must show that p is in fact not dominated by any feasible point. To see why this is true, assume for the sake of contradiction that there existed some feasible route assignment $X = \{x_{sd}\}$ for which both $intr(X) < intr(L^*)$ and $down(X) < down(L^*)$. This implies

$$intr(X) + \lambda \cdot down(X) < intr(L^*) + \lambda \cdot down(L^*)$$

and hence that for some source-destination pair (s, d),

$$intr(x_{sd}) + \lambda \cdot down(x_{sd}) < intr(\ell^*_{sd}) + \lambda \cdot down(\ell^*_{sd}).$$

But then ℓ^*_{sd} must not have minimized (1)—a contradiction. Hence, the point p returned by the procedure is undominated.

We have thus reduced the problem to that of minimizing (1) for an individual source-destination pair (s, d). We note[1] that whenever a path is selected at time t, an optimal choice is the path which will be available for longest time into the future beginning at time t. Thus, the Most Stable Paths subroutine provides (a superset of) the set of routes which might be involved in the optimal sequence, ℓ^*_{sd}. We add to this set a persistently available "null route" with cost λ per unit time, and feed all these choices to the OPS subroutine, whose output must be an optimal set of route selections, ℓ^*_{sd}.

That concludes our procedure for lower-bounding the stability-availability tradeoff. Our bound in the stability-deviation space is quite similar. The main difference is that since route preferences are based on next-hops (see Sec. II-B), the set of routes in ℓ^*_{sd} might include the route *through the most preferred neighbor* which will be available longest into the future, rather than just the overall longest-lived route. We modified the Most Stable Paths subroutine to obtain these routes. We label them with costs per unit time—zero while they are preferred, or λ otherwise—and then feed them into OPS to produce ℓ^*_{sd}.

IV. Stable Route Selection

We next describe our proposed class of Stable Route Selection strategies. SRS avoids instability by using flexibility in route selection to select more stable paths (approach (2a) in the classification of Section II-C). SRS offers a tunable tradeoff between stability and amount of deviation from preferred paths. In this section, we describe where SRS fits in the context of the BGP decision process (Section IV-A), and then how our SRS strategy selects paths (Section IV-B).

A. *Fitting SRS into BGP*

BGP's *decision process* [34] allows operators to customize route selection to conform to goals such as traffic engineering or economic relationships. The BGP decision process consists of the sequence of steps shown in Table I, which select a route based on *attributes* contained in the BGP route announcements. The output of each step is a *set* of routes that are *equally good* according to that and every previous step. By adding, modifying, or filtering attributes in update messages, operators can control the specific route selected to reach a particular destination.

We insert the SRS heuristic as an additional step between Steps 1 and 2 of the BGP decision process. (Alternately, SRS could be implemented by appropriately modifying route attributes using an import filter before routes reach the decision process.) SRS selects the best route based on a combination of Steps 2-7 and a heuristic for predicting route stability. We present the details of SRS in the next section.

[1]For a proof, see Fact 5 in Appendix C.3 of [13].

Step	Action
1.	Highest local preference
2.	Lowest AS path length
3.	Lowest origin type
4.	Lowest MED
5.	eBGP over iBGP-learned
6.	Lowest IGP cost
7.	Lowest router ID

TABLE I: BGP Decision Process

An alternate design would have placed SRS before the first step, like flap damping. We chose to place SRS after the Local Preference step to ensure that the highest-level routing preferences, such as preferring customer routes over provider routes, are always maintained, even during SRS's delay period (see below). This has at least two benefits. First, it provides a useful guarantee to operators. Second, we note that it is possible for a violation of the Local Preference step to reduce availability for other ASes. In particular, ASes typically have neighbors who are *providers*, *customers*, or *peers*; a route whose next-hop is a provider or peer is exported only to customers [12]. If an AS were to select a provider route over a customer route, it would block the export of the route to other providers and peers, potentially disconnecting those neighbors from the destination. Although this case may be uncommon, it is our primary concern to ensure high availability.

Despite the restrictions imposed on SRS by being subordinate to Local Preferences, in our simulations, sufficient flexibility remains to significantly improve stability.

B. The SRS Heuristic

The SRS heuristic is inserted after Step 2 of the BGP decision process (Table I). The heuristic has one main parameter, a *delay* δ. We will write $SRS(\delta)$ to indicate the value; SRS with the parameter omitted refers to $SRS(\infty)$. SRS uses a procedure, $pref(r_1, r_2)$, that implements Steps 2-7 in Table I. $pref(r_1, r_2)$ returns "first" if r_1 is more preferred according to Steps 2-7, "second" if r_2 is more preferred, or "equal" if they are equally preferred. SRS then decides which of two routes r_1, r_2 should be selected as follows:

1) If r_1 has been up for time $\geq \delta$ and $pref(r_1, r_2) =$ "first", then select r_1.
2) Otherwise, if r_2 has been up for time $\geq \delta$ and $pref(r_1, r_2) =$ "second", then select r_2.
3) Otherwise, if one of r_1 and r_2 is currently selected, keep that route.
4) Otherwise, if one of r_1 and r_2 has lower AS path length, select that route.
5) Otherwise, select the route that has been up for the longest time.

The single winning route is selected by iterating this pairwise comparison over all available routes.

The intuition behind this choice of steps is as follows. Steps 1 and 2 select preferred routes, as long as they are not recent advertisements. This step assumes that recently advertised routes are more likely to be withdrawn soon, and provides a tradeoff in the parameter δ. $SRS(0)$ is equivalent to the decision procedure $pref(\cdot, \cdot)$, while $SRS(\infty)$ gives no consideration to preferred routes, reserving maximum flexibility for stability.

The strategy of sticking with the current choice (Step 3) and then using a "longest uptime" strategy if that choice fails (Step 5) has been used in many contexts from page replacement to peer-to-peer systems, and is a good heuristic for stability since past behavior is frequently correlated with future behavior [14]. In simulations, we found that inserting the shortest-path step (Step 4) often somewhat improved both stability and mean path length.

V. Analytical and Simulation Evaluation

In this section we describe results from a simulation-based evaluation, as well as the results of our lower bounds algorithms applied to the same environment as the simulator. We describe our simulation methodology in Section V-A and present results in Section V-B.

A. Methodology

Data sets. We infer the inter-domain AS-level topology by culling AS adjacencies from Route Views [31] feeds. Since policies on the Internet are not widely disclosed, we use [35] to infer and assign local preferences associated with business relationships as done in [22], [24], [36], characterizing links as either provider-customer or peer-peer. We assume ASes distribute routes according to the common-case import and export policies as discussed in [35]: ASes prefer customer over peer and peer over provider routes, and don't advertise routes from peers/providers to other peers/providers.

To infer the pattern of failures, we record the appearances and departures of links from the Route Views feeds. Specifically, we consider a link to be available at time t if some route which uses the link is currently advertised to a Route Views peer at time t. In this manner, we infer a trace of link state changes from Route Views from January 1, 2006, to December 30, 2006, which we replay against our simulator.

Simulator. To evaluate the performance of various route selection strategies, we use a discrete-event BGP simulator extended from that of [8]. The simulator's events are at the level of link state changes and BGP update messages. As in some past studies [5], [36], we represent each AS as a single node running a BGP instance, for scalability and since internal AS topology are not available. Inter-AS packet propagation delay is selected randomly for each packet, uniformly distributed between 5 and 15 ms.

The simulator runs a simplified version of the BGP protocol described in RFC 1771 [30]. Since our simulator models each AS as a single router, Steps 3-6 of the decision process (see Table I) are not executed. For Step 7, we assign each AS a uniform random router ID.

We implement batching (see Sec. II-A) in the simulator to compare with our lower bounds. We do this in such a way that the BGP update messages are processed *in the same order* that they would have been with link delays and MRAI timers turned on and with subsequent topology changes delayed until after the convergence process completed.

Each plot incorporates measurements of at least 100 trials. In each trial we select a single random destination to which all nodes route over a random month of our year-long data. We gather measurements only after the first 24 hours of simulated time, to eliminate initial convergence effects. Since some data is missing from our topology causing a minority of nodes to be always disconnected, when collecting measurements we ignore source-destination pairs whose availability in the Standard BGP strategy (without flap damping) is < 0.99. Other than excluding ASes which were usually disconnected, this did not substantially affect our results.

Route selection strategies. Our simulations will compare the basic BGP decision process, which we call *Standard BGP*, with SRS and with Route Flap Damping (RFD) as in RFC 2439 [37].

RFD maintains a numeric penalty value $p_{P,N}$ associated with every (prefix P, neighbor N) pair. Upon receipt of an advertisement or withdrawal, the router increases $p_{P,N}$. When $p_{P,N}$ increases beyond a *cut-off threshold*, the route is excluded from consideration when selecting routes. The penalty decays exponentially, and the route is reconsidered for use when its value falls below a *reuse* threshold. In our tests, unless otherwise stated, the strategy “RFD” refers to flap damping with Cisco’s default parameters, which increase $p_{P,N}$ by 500 after attribute changes and by 1000 for withdrawals, and specify a reuse threshold of 750, a cut-off threshold of 2000, and a decay half-life of 15 minutes [24]. We also test with flap damping parameters used by Juniper [24], SprintLink [33], and three sets of parameters recommended by RIPE [28].

B. *Results*

This section presents the results of our simulation and our lower-bounds analysis, both applied to the environment described above. Our main conclusions are as follows:

- Batching, which allows us to obtain bounds on the optimal policies, preserves the qualitative performance of various strategies (Section V-B1).
- Improvements to convergence cannot obtain a large improvement in our environment. This conclusion is surprisingly robust under various message propagation delay distributions, but convergence overhead can be larger due to policy misconfiguration and withdrawals by origin ASes (Section V-B2).
- $SRS(\infty)$ can obtain a dramatic improvement in stability compared with Standard BGP and greater than that of RFD, without sacrificing availability and closely approaching our lower bound (Section V-B3).
- By adjusting the delay parameter appropriately, SRS can reduce interruption rate by 68% while deviating from preferred paths less than 0.6% of the time (Section V-B4).
- SRS only slightly increases mean path length (Section V-B5), and stability-aware routing can obtain significant improvements in stability even under limited deployment scenarios (Section V-B6).

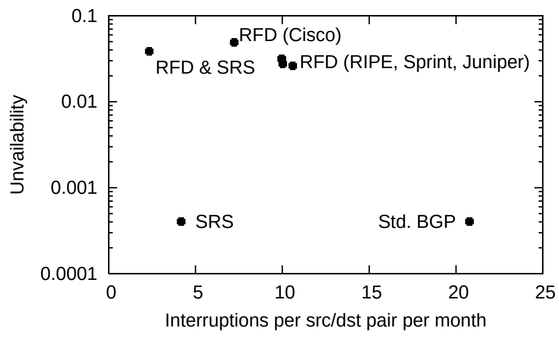
1) Effect of batching: Figure 3 shows performance of various strategies both without and with batching. Recall from Section II-A that batching makes link delays and MRAI timers negligible, allowing us to lower-bound the interruption rate of optimal policies. We see a substantially similar relationship between the strategies with batching on and off, which suggests that batching is a reasonable approximation under which to compare strategies. The main difference is inflated interruption rates, which can be explained by the fact that in Fig. 3(a) some link state changes may be effectively skipped as a result of link delays and MRAI timers, while in the batched environment the system finishes reconverging after every topological change.

2) Convergence overhead: Figure 3(b) shows the interruption rate of Standard BGP and SRS along with their hypothetical counterparts with optimal convergence—which transition from the initial to the final path in each batch without any path exploration process. This shows that in our environment, convergence has only a minor contribution to interruption rate. But on what aspects of the environment does this conclusion depend? We investigated how origin events, policy misbehavior, and heterogeneity of link delays affect convergence overhead. In what follows we define the **overhead ratio** of a route selection scheme as the number of interruptions experienced by the scheme divided by the minimum number of interruptions needed to produce the same route selections at all times except during convergence events.

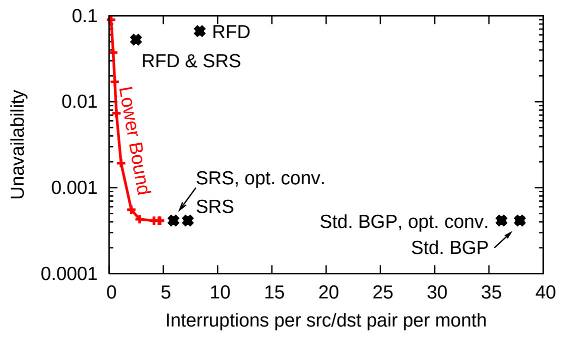
Origin events. ISPs may withdraw and announce prefixes either intentionally or due to configuration error, triggering long sequences of path hunting which are triggered relatively rarely by our Route Views traces of link state changes. Here, we consider the effect of an origin AS announcing or withdrawing a single prefix. To do this, we constructed a trace consisting of announcements/withdrawals separated by long periods of time. This allowed us to measure the convergence overhead for each change in isolation, and allowed us to compare against the BGP Beacons measurement study by Mao et al. [25]. Like [25], we found that prefix advertisements generate roughly 3-5$\times$ fewer updates than withdrawals. For example, with our default configuration, we found routing advertisements incurred an overhead ratio of 1.14 while withdrawals incurred an overhead ratio of 6.00 on average. In addition, we found that failing a single link adjacent to the destination incurred an average overhead ratio of 3.90, while repairing the single link incurred an average overhead ratio of 1.03.

Policy misconfiguration. Misconfigured ISPs may introduce oscillations and instability which can lengthen convergence periods. We randomly selected a small fraction of ASes to “misbehave” by selecting routes based on a uniform-random preference ordering among next-hops, rather than our default configuration based on the Gao-Rexford policies [12]. We found that having a fraction of ASes misbehave in this manner can increase convergence overhead; for example, the convergence overhead ratio is 2.68 with 2% of ASes misbehaving, or 2.70 with 5% misbehaving.

Link delay. Increasing the delay of links, or the heterogeneity of delays across links, has been associated with worsening of



(a) Batching off



(b) Batching on

Fig. 3: Performance of various strategies in the stability-availability space, with batching off and on. SRS's delay parameter is fixed at ∞. Unless otherwise specified RFD refers to flap damping with Cisco standard parameters.

routing convergence times [21]. To measure this, we varied the distribution of delays of inter-AS links in our simulator. We assigned each link a delay sampled from a Pareto distribution with shape parameter α, which controls the variance of the distribution, and varying mean μ. Results are shown in Table II. Like previous work, we found that increasing link delays increases convergence *time*. However, varying the variance of links, and varying the mean delay of links, only changed convergence *overhead* slightly. Like previous work [15], we also found that enabling MRAI increases convergence time, but reduces control overhead.

α	**Convergence overhead ratio**	
	$\mu = 4$ ms	$\mu = 100$ ms
2.000001	1.154	1.159
2.0001	1.166	1.151
2.01	1.168	1.167
2.1	1.167	1.165
2.4	1.163	1.165
4	1.165	1.166
15	1.169	1.170
2000	1.172	1.172

TABLE II: Effect of varying link delay.

3) Stability-availability tradeoff: Figure 3 shows performance of various strategies in the stability-availability space. Comparing the points in Fig. 3(a), where batching is disabled, Standard BGP maintains a high availability of 99.98%, but suffers from a high rate of 20.8 interruptions per month. Route Flap Damping (RFD) with Cisco's default parameters reduces the mean rate of interruptions by $2.9\times$, but sacrifices two "nines" of availability, and the other parameter values used by Sprint and Juniper and recommended by RIPE have substantially similar performance. In contrast, by preferring more stable paths, SRS is able to achieve the high availability of standard BGP with even fewer interruptions than RFD. Although we do not advocate the use of RFD, we note that using RFD and SRS in conjunction results in an additional 3.1-fold decrease in interruption rate over RFD and slightly improves availability over RFD alone. This is to be expected, since by picking more stable paths, RFD is triggered less often.

Figure 4 explores the pattern of interruptions in more detail

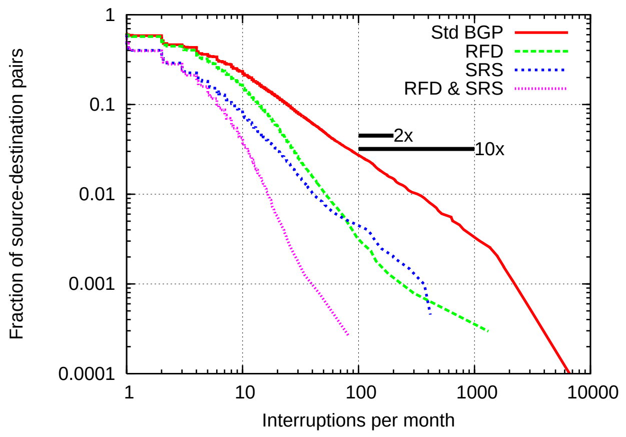
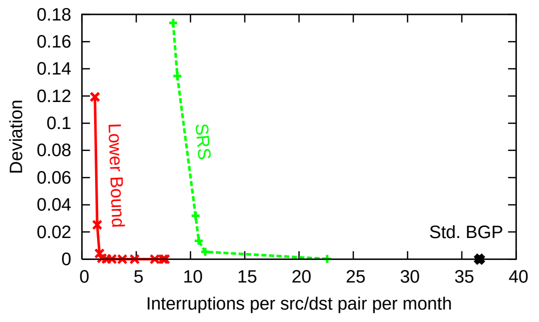


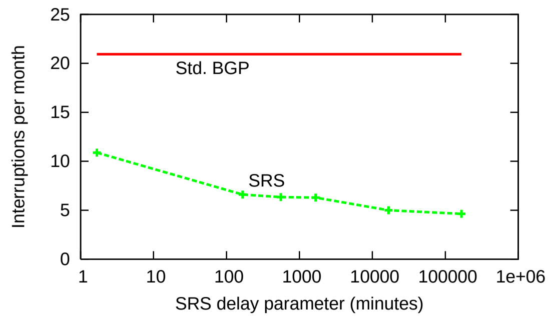
Fig. 4: Complementary CDF of interruptions per month over all measured source-destination pairs. SRS's delay parameter is fixed at ∞.

with a complementary CDF over all measured end-to-end paths. That is, the y axis shows the fraction of (source, destination) pairs that have *at least* the interruption rate on the x axis. Standard BGP's long tail shows what other studies [10] have observed: a small number of Internet routes suffer from high instability. Both SRS and RFD drastically reduce the size of this tail. SRS is able to achieve roughly the same benefit in the tail as RFD without incurring RFD's reduction in availability. Interestingly, and unlike RFD, SRS improves the stability for the upper part of the curve, i.e., for routes that have only moderate instability. Finally, when we combine SRS with RFD, we note that the instability of the most unstable routes is reduced by about an order of magnitude compared with RFD in isolation.

Figure 3(b) also compares performance with lower bounds on the optimal policies. SRS performs surprisingly close to optimal among strategies which achieve the highest availability, with an interruption rate just 55% higher than our lower bound, which uses knowledge of the future. Factoring out SRS's convergence overhead, it would be just 27% worse than optimal. These results indicate that the SRS heuristic does a



(a) The stability-deviation tradeoff space with *batching on*.



(b) SRS's interruption rate as a function of its delay parameter, with *batching off*.

Fig. 5: Deviation and interruption rate of SRS(δ) for various δ. SRS's delay parameter δ varies from 100 sec to effectively ∞ (a value longer than the one-month trace).

very effective job of predicting the relative stability of paths in this environment. It also shows that BGP's stability cannot be improved by more than about $8.1\times$ without sacrificing availability, given our assumptions such as the preservation of common business relationship-based routing policies and path consistency (see Sec. II-A).

Finally, the "Lower Bound" curve in Figure 3(b) demonstrates limits on how much improvement can be gained by occasionally disconnecting nodes. This lower bound admits the possibility that stability can be improved to about 2 interruptions per month with small availability loss, but any further improvements come at the cost of significantly more downtime. For example, reaching 1 interruption per month requires reducing availability from 99.96% to below 99.8%, *i.e.*, over $4\times$ as much downtime.

4) Stability-deviation tradeoff: The above results, which set SRS's delay parameter to ∞, assume SRS is permitted to deviate greatly from the operator's preferred routes. In this section, we explore the tradeoff between stability and the fraction of time that routes deviate from the next-hops chosen by Standard BGP.

Figure 5(a) shows the stability-deviation tradeoff that results from varying SRS's delay parameter, in the batched environment. With a delay of $\delta = 100$ sec, SRS cuts interruption rate by about 38% compared with Standard BGP and has a mean deviation of 0.021%, with 99.5% of ASes having deviation $< 1\%$. At the knee of the tradeoff curve, with $\delta = 167$ minutes, interruption rate is 69% lower than Standard BGP; mean deviation is 0.54% and 86% of ASes have deviation $< 1\%$. Figure 5(b) shows that even with batching off (MRAI and link delays on), SRS with $\delta = 167$ minutes still reduces interruption rate by 68%. These results are promising: many ISPs base bandwidth utilization payments on the 95th percentile of the traffic load for each month [27], so a deviation of less than one percent may be acceptable.

We also note there may be cases where it is useful to move beyond the knee to obtain greater stability. For example, if an ISP has the ability to route multiple classes of traffic along different routes, it would be possible to send the most stability-sensitive flows (*e.g.*, real-time voice traffic) along SRS paths, and other flows along whichever paths minimize maximum link utilization. This would allow the ISP to achieve critical traffic engineering objectives while still providing greater stability where it matters.

Figure 5(a) also plots our lower bound to the optimal achievable points in the stability-deviation space (without sacrificing availability). In contrast with the stability-availability lower bound, this is quite far from the upper bound, SRS. It is likely that the true optimal is actually much closer to SRS. To see why, recall that due to computational intractability of computing the optimal, we lower-bound the optimal by computing the optimal sequence of paths for each node independently, allowing nodes to take inconsistent paths. In particular, a node can adhere to its preferred next hop, while choosing the remainder of the path to be maximally stable. But this has implications on the amount of deviation of the nodes along the path, which our lower bound algorithm does not take into account. Providing a substantially better lower bound would likely be possible for an alternate definition of deviation: requiring that the entire path matches the preferred path, rather than just the next hop.

5) Path length: The previous section dealt with adherence to general routing objectives, in the form of next-hop preferences. In this section, we study one objective which is not reducible to next-hops: path length.

Figure 6 shows the CDF of mean path length over source-destination pairs. SRS(∞) has a mean path length of 4.14 AS-level hops, or just 4.2% greater than Standard BGP's 3.97 hops. We note that BGP itself empirically incurs an AS-level path inflation of 49.8% due to policies [4]. Hence an additional inflation of 5% (resulting in a combined inflation of roughly 56.4%) may be tolerable to some ISPs, while others may tune SRS to reduce this inflation at the expense of a somewhat lower stability.

This low inflation may be expected since part of the SRS heuristic prefers shorter paths (Sec. IV). In addition, path lengths in this environment are constrained by the hierarchical nature of the AS graph when business relationships are satisfied: we found that a hypothetical strategy which always preferred *longest* paths would have mean path length just 32%

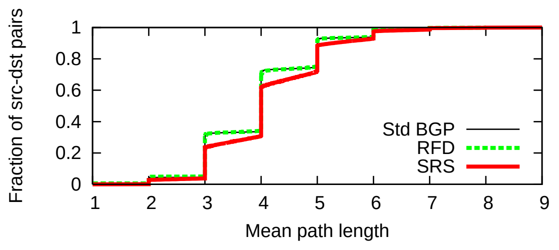


Fig. 6: CDF of mean path length over all measured source-destination pairs.

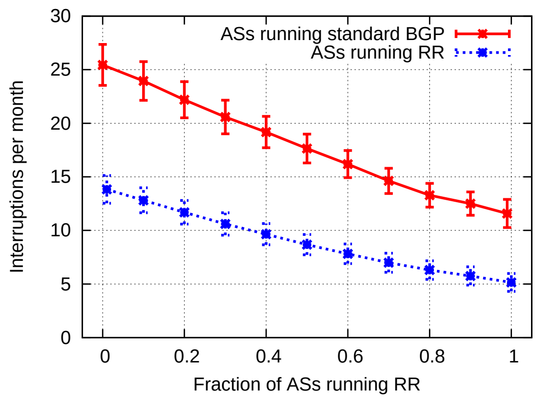


Fig. 7: Interruption rate under partial deployment of a stability-aware routing strategy, with delay parameter $\delta = \infty$ and batching off.

longer than Standard BGP.

6) Partial Deployment: Since Internet path selection is a collaborative process, SRS's benefit to one autonomous system would depend on whether other ASes have also deployed it. But for reasons of deployment incentive, it would be helpful if a single AS can unilaterally deploy SRS and achieve at least some improvement.

The following evaluation of partial deployment scenarios uses a Random Replacement strategy (following [14]), rather than SRS. Specifically, instead of Steps 4 and 5 of the SRS heuristic described in Section IV-B, we select a random available route from the set of routes most preferred by the previous steps. These results use RR only for historical reasons, and the performance is very similar to SRS with delay $\delta = \infty$.

Figure 7 shows the performance of this RR strategy with the extent of deployment ranging from 1% to 100% of ASes. The set of ASes running RR is selected randomly in each trial, and the others use standard BGP. We measure the interruption rate on those nodes running RR separately from those not running RR. The leftmost points in the plot show that the first ASes deploying RR would see a significant drop in their own interruption rate of roughly $1.8\times$. As more ASes deploy stability-optimized route selection, the interruption rates decrease further, so that when all ASes implement RR, we achieve a $5\times$ reduction in the interruption rates (similar to the improvement with a full deployment of SRS(∞) as shown in previous plots). This is due to the distributed nature of BGP path selection: each AS that prefers more stable routes effectively stabilizes routes on behalf of other ASes that route through it (including those running standard BGP!).

The factor of $1.8\times$ improvement, however, is for the average AS; certain ASes obtain a somewhat bigger improvement. In particular, we found that individual Tier-1 ASes (as classified by [35]), which have more flexibility in path choice, had a median reduction in interruption rate of $2.7\times$, with 4 of the 28 Tier-1s seeing more than a $4\times$ improvement.

C. *Deficiencies of the simulation results*

The simulation results presented in this section afforded us a great deal of flexibility—scaling to large networks, simulating partial or full deployment, and comparing with the theoretical optimal. However, the simulation is disconnected from a real deployment in two key ways. First, it is a control-plane simulation, rather than an implementation including a data plane, so we cannot directly measure data-plane effects. We address this deficiency in Section VI by testing a software router deployment.

Second, the environment (topology, routing policies, link failure patterns) may be unrealistic, even though we used data sets inferred from real-world measurements. We address this deficiency in Section VII by using direct feeds of BGP updates from actual Internet routers.

VI. Software Router Experiments

In this section we evaluate SRS in a network of software routers. Although we have scaled it only to tens of nodes, this deployment allows us to (1) measure performance in a realistic implementation, and (2) validate our control-plane simulations by correlating them with observed data-plane performance.

A. *Methodology*

Our experimental evaluation is based on the BGP implementation in the Quagga software router [29]. We modified Quagga in two main ways: we implemented new route selection policies in the decision process, and we built a custom forwarding plane to enable more flexible experimentation. We then evaluated the resulting software router by deploying it on a cluster and emulating failures. These steps are described in more detail below, and the overall arrangement of components is depicted in Figure 8.

Route selection policies. We altered Quagga's decision process to (optionally) run SRS with delay $\delta = \infty$ or $\delta = 10$ min. We also tested several strategies already implemented in Quagga: what we have referred to as Standard BGP; Quagga's default strategy, which employs a longest-uptime step directly before the final router-ID step of the decision process; and Route Flap Damping, again with Cisco-default parameters.

Forwarding plane. We built a custom data plane to enable more complete instrumentation and to conveniently run on a shared cluster. In effect, this data plane allows us to use Quagga as a router for an overlay network. As depicted in Fig. 8, we instrumented Quagga so that it sends route updates, in the form of (destination, next hop) pairs, to our forwarding plane, rather than to the kernel. The forwarding plane generates, receives, and forwards UDP packets to remote forwarding plane processes, as directed by Quagga's route updates. Data packet generation occurs every 5 seconds, with 10% random jitter, for every (source, destination) pair. That is, the time between packets for a (source, destination) pair is uniform random in $[4.5, 5.5]$.

Emulating failures. We emulate link failures by leveraging Linux's iptables facility, with which we can block or unblock UDP and TCP traffic between pairs of software routers at appointed times. This allows us to emulate a trace of link failures and recoveries over our chosen network topology.

We configured our software routers using two small-scale network topologies. First, we employ IS-IS link-state updates and topology traces from the Abilene backbone network [1]. Although our primary target is interdomain rather intradomain routing, this gives us a realistic environment of scale appropriate for our testbed. The topology contains 11 nodes and 14 edges. We use a portion of this trace from 10 August 2004 to 13 Oct 2004. We "compressed" this trace by reducing to 2 minutes every interval of greater than 2 minutes in which no events occurred. This reduced the length of the one-month trace to 7.3 hours. The compression step preserves the ordering of events, while allowing us to run tests in a shorter time period and to stress-test the route selection policies in a more challenging environment. This does change the pattern of failures and can have an effect on the performance of the strategies we test. However, the main goal of this section is not to test a realistic pattern of failures, but rather to observe the effects of moving from a simulation environment with only a control plane, to an implementation with a data plane.

The second environment we use is a synthetic Erdös-Rényi $G(n, m)$ random graph, i.e., n nodes with m edges connected uniformly at random. We used $n = 25$ and $m = 50$, so that the average node degree is 4. We generated a bimodal pattern of failures among the m edges: a random set of 40 are stable, never failing; the other 10 are unstable, with a heavy-tailed uptime distribution of mean 2 minutes, and constant 1-minute downtimes. The trace lasts 2 hours.

We exclude results near the beginning and end of the trace to avoid measuring startup and shutdown effects. We show results from single trials, but we have found repeated trials produce very similar results.

B. Results

In this section we show that SRS can substantially improve data plane performance over other strategies; interestingly, a delay of $\delta = 10$ minutes performs better than $\delta = \infty$, potentially due to faster convergence as a result of slightly shorter paths. We then correlate our simulator's interruption rate metric with packet loss in the software router.

1) Software router performance: Table III classifies packets by their fate. The columns respectively indicate packets that were received, dropped because there was no route to the destination, sent along a virtual link which was down (i.e., dropped by iptables), dropped after exceeding the maximum hopcount, or dropped for unknown reasons (presumably, dropped by the underlying physical network).

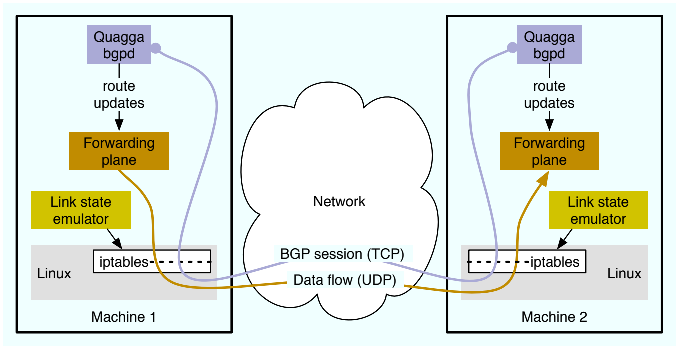


Fig. 8: A diagram of our software router, showing two nodes with a flow of data from Machine 1 to Machine 2.

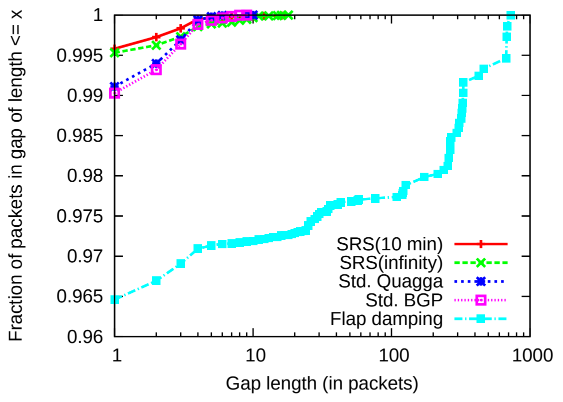
Figure 9 depicts data-plane performance in the form of the distribution of "gap lengths". We say that a generated packet lies in a gap of length 0 if it was received by the destination, and it lies in a gap of length k if it is one of a run of k dropped packets. Gap lengths are significant since brief outages can often be masked by retransmission. Note that a gap length of k corresponds to an outage of $\approx 5k$ seconds.

Strategy	**Sent**	**Recv**	**Dropped**			
			No route	Link $\downarrow$	> 30 hops	?
Abilene Environment						
SRS(10 min)	515900	513103	161	2636	0	0
$SRS(\infty)$	515900	512714	475	2696	5	10
Std Quagga	515900	509885	88	5918	1	8
Std BGP	515900	509441	123	6336	0	0
RFD	515900	496464	14277	5159	0	0
Random Graph Environment						
$SRS(\infty)$	810000	809981	0	0	0	19
Std Quagga	810000	782888	178	26532	43	359
RFD	810000	780812	14001	15186	1	0

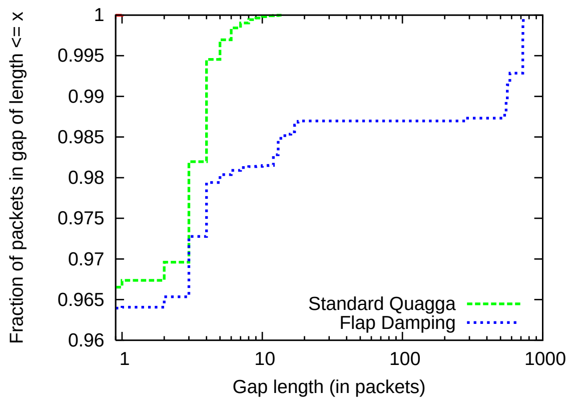
TABLE III: The fates of packets in the software router experiments.

In both environments $SRS(\infty)$ substantially decreases packet loss. In the Random Graph, it is able to avoid all unstable links within the 250-second period before measurements begin, resulting in zero measured packet loss. Although this is an artificial environment, it demonstrates that SRS successfully finds the stable paths. Flap damping, in contrast, fails to find them in a reasonable amount of time; this is somewhat surprising, given the simplicity of the bimodal failure pattern.

Despite its lower overall drop rate, $SRS(\infty)$ has a slightly longer tail in the gap length distribution (see Fig. 10), and more of its dropped packets are due to lacking a route provided by the control plane. This may represent increased convergence time as a result of longer paths. Surprisingly, even though it is given less flexibility, $SRS(10 \text{ min})$ is a win over $SRS(\infty)$ in every type of dropped packet, as well as in the tail of the gap length distribution. $SRS(10 \text{ min})$ still has a longer tail than Standard BGP, but only after the 99.98th percentile. Given its factor $2.3\times$ reduction in packet loss, $SRS(10 \text{ min})$ presents a promising alternative.



(a) Abilene environment



(b) Random graph environment

Fig. 9: Gap lengths in the software router experiments under the two environments. Packets are spaced at ≈ 5-second intervals.

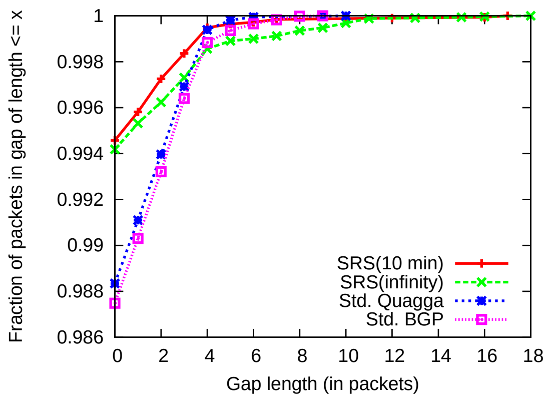


Fig. 10: Fig. 9(a), zoomed in.

Standard Quagga offers a marginal improvement over Standard BGP in the Abilene environment. RFD also reduces the fraction of packets sent along dead links, which is the largest cause of packet loss in Standard BGP and Standard Quagga. However, RFD pays for this with a large number of packets dropped due to having no route, and we can see from Figure 9 that these cause exceedingly long periods of outage in both environments.

2) Correlation with simulation: The goal of this section is to determine how well our simulator's interruption rate metric predicts data plane performance. We match the interruption rate, measured in simulation with *batching on*, with the packet loss rate of our software router experiments, both run under the same environments.

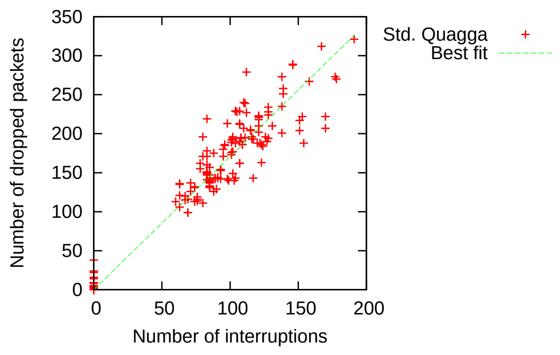
We begin with the larger of the two environments, the random graph, using the Standard Quagga strategy. Figure 11(a) plots packet loss rate vs. number of interruptions for the 600 source-destination pairs (note that many overlap at $(0, 0)$). Although some variation is to be expected due to implementation details or timing such as the random intervals between packet generation, we see a strong correlation; the data has a correlation coefficient of 0.985896. A least-squares fit of $f(x) = a \cdot x$ yields $a \approx 1.7067$, which means that the average interruption in this environment causes a loss of ≈ 1.7 packets or ≈ 8.5 sec of availability.

But this is an average which does not describe every individual interruption event. Excluding drops due to the underlying network, packets can be lost in our software router for three reasons: having no route to the destination, forwarding along a link that is down, or exceeding the maximum hopcount (i.e., encountering a forwarding loop). As long as a working path actually exists, each of these cases must result from transient conditions involving an interruption. On the other hand, an interruption need not imply packet loss: a router could simply switch between two working routes.

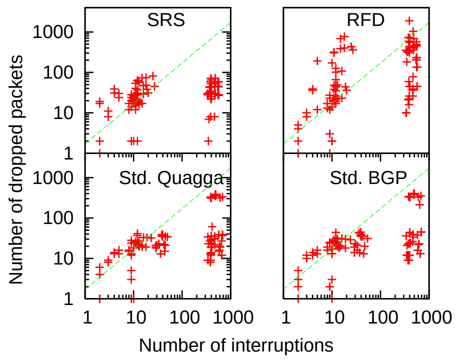
This intuition is borne out in the Abilene environment, depicted in Figure 11(b) along with the best-fit line imported from the random graph environment. Abilene has only 110 source-destination pairs, rather than 600; and the trace is more heterogeneous, with a single highly unstable link and several which occasionally fail, as opposed to the random graph environment's 40 stable and 10 unstable links. It is then not surprising that source-destination pairs' performance is more highly variable in the Abilene environment. In particular, in all four strategies, a number of points lie near the best fit line; and many points also lie below this line, corresponding to the fact that an interruption does not necessarily cause a packet loss. In the SRS, Standard Quagga, and Standard BGP strategies, fewer points lie far above the best fit line, corresponding to the fact that packet loss within the software router network does not occur without instability. Finally, in RFD, a large fraction of points have many more packet losses than the interruption rate alone would predict. This can be attributed to the fact the simulator counts instability separately from unavailability, the latter being the likely cause of these losses.

VII. Evaluation with Route Views Update Feeds

In this section we evaluate the performance of SRS using feeds of BGP updates from actual Internet routers, as collected by Route Views [31]. Note that while the simulations of Section V used traces of AS-level link state availability that were *derived* from Route Views, in this case we use unmodified streams of updates, providing a much more realistic evaluation.



(a) Random graph environment



(b) Abilene environment

Fig. 11: Correlation between number of interruptions in simulations, and packet loss in experiments, for the two environments.

However, the deployment scenarios are more limited: only a single node runs SRS, whereas Section V was able to explore effects of all or a subset of nodes running SRS.

In this evaluation, SRS provides very little benefit for the average IP prefix. We find that one reason is correlation in paths: for most prefixes, although multiple paths are available, they have a common last link. For prefixes that have path diversity, SRS provides greater improvement.

A. Methodology

Our evaluation is based on a log of update messages and periodic snapshots of the routing tables (RIBs) from about 42 Internet routers, called *views*, which peer with a Route Views data collector. This data gives us a way to determine the control-plane effect of running SRS at a single router which peers with some subset of these views.

Specifically, our emulator proceeds as follows. In each trial, we create a router which has "virtual peerings" with a random subset of 5 of the Route Views views. This router receives one month of data, beginning with a snapshot of the RIB and proceeding with the update messages that it would receive from each of its virtual neighbors. We emulate the BGP and SRS decision processes over this data, recording interruptions and other measurements on a per-prefix basis. When collecting data we ignore the first 200,000 seconds (2.3 days) to avoid startup effects.

The results we present are based on 40 trials for each of the two strategies: 10 trials from each month between November 2008 and February 2009. We show 95% confidence intervals.

B. Results

In our experiments, SRS with delay $\delta = 60$ minutes decreases mean interruption rate by 12%:

Strategy	Mean interruptions per prefix per trial
BGP	20.67
SRS(60 min)	18.28

We next examine how this result behaves as a function of the prefix length and diversity of available paths (to be defined below). These results are shown in Figure 12.

Prefix length. Figure 12(c) shows that SRS has similar performance on prefixes of length ≥ 16. For shorter prefixes, the variance in the results is quite high (Fig. 12(a)); note that there are relatively few prefixes of length ≤ 16 (Fig. 12(e)).

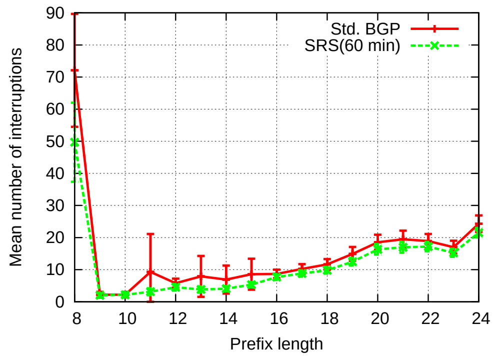
Path diversity. We define path diversity for each prefix as follows. At a given point in time, the path diversity is the number of distinct *penultimate ASes* among the available paths at that time. (This definition is essentially equivalent to the number of AS-disjoint routes.[2]) We then take the mean of this value across all times that there is at least one available route for the prefix. In plotting results, we group prefixes into bins by rounding their mean path diversity to the nearest integer i; we will refer to these prefixes as having diversity $\approx i$.

Figure 12(d) shows that for prefixes with diversity ≈ 1, SRS achieves only an 8.7% reduction in interruption rate compared with BGP. But performance improves substantially once there are multiple disjoint paths, with a 23% or 27% reduction when path diversity is ≈ 2 or ≈ 4, respectively. It is unclear why the interruption rate is *higher* for path diversity ≈ 5; however, note that the variance of the SRS measurements is much higher here (Fig. 12(b)). Considering all prefixes with diversity ≥ 2 as a group, there is a 24% reduction in interruption rate.

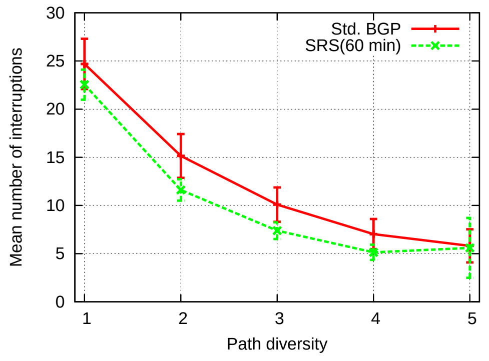
Conclusions. There are likely several reasons that SRS's improvement over BGP is more limited than our previous simulations suggested. First, as we have shown, SRS performs better with higher path diversity; but about 62% of prefixes have path diversity ≈ 1 (Fig. 12(f)). Even if the destination AS has multiple providers, this can occur if the AS is announcing the prefix to only one provider, which is a common way of performing traffic engineering. Our simulator did not consider traffic engineering.

Second, in this section our evaluation methodology allows us to emulate only a *single* router running SRS, while most of our previous results dealt with the case that *all* routers run SRS. Our results of Section V-B6 suggest that performance would improve with wider deployment.

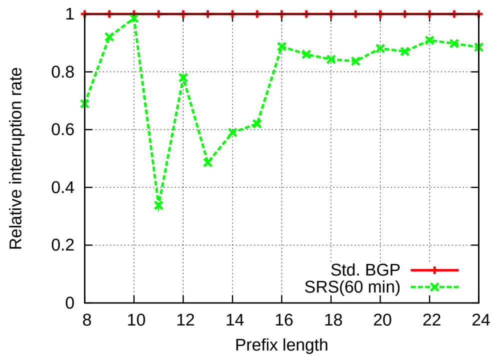
[2]This is due to the fact that converged BGP routes for any given prefix form a tree; thus, if two routes differ in the hop immediately before the destination, they must differ in *all* hops, except the destination. However, there may be slight differences due to message timing and convergence.



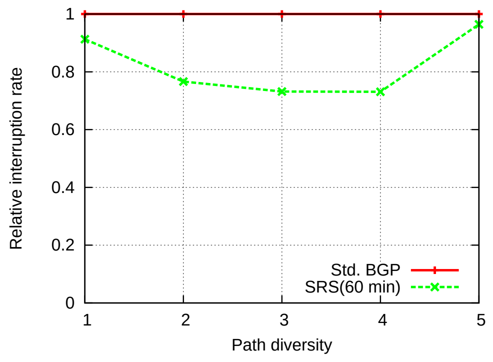
(a) Interruption rate by prefix length



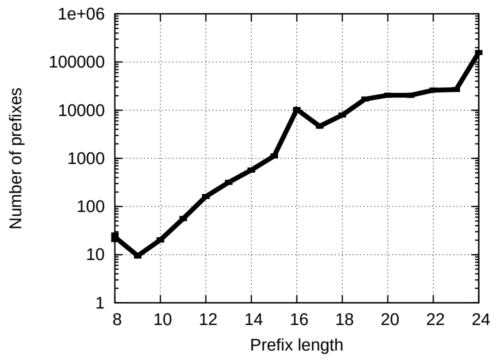
(b) Interruption rate by path diversity



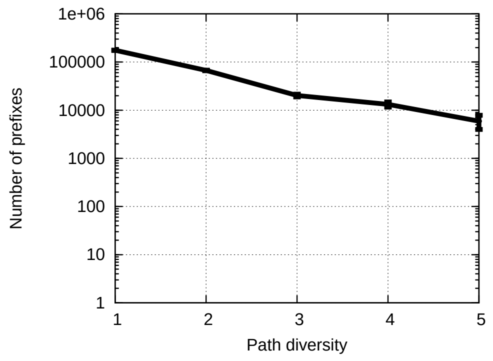
(c) Interruption rate by prefix length, relative to BGP



(d) Interruption rate by path diversity, relative to BGP



(e) Number of prefixes by prefix length



(f) Number of prefixes by path diversity

Fig. 12: Results of Route Views update feed evaluation.

VIII. Related Work

Approaches for improving stability of Internet routing typically fall into two classes: *modifying the routing protocol*, or *modifying route selection*.

Modifying the routing protocol. By appending information about the cause of a route update, the convergence process can be shortened [6], [24]. Loop-free convergence [3] aims to ensure certain correctness properties hold while routing updates are propagating. Instead of changing the network layer, data packets may be sent on overlay networks which can route around failures [17]. In contrast to these studies, we refrain from modifying BGP, and our SRS strategies focus on avoiding convergence entirely rather than reducing convergence overhead.

Modifying route selection. Some work has studied what path selection policies lead to guaranteed convergence. Griffin et al. [16] showed that a stable state exists if the ASes' policies do not contain a dispute wheel, while [12] showed that by setting policies according to certain locally-checkable guidelines (e.g., preferring customer routes), convergence to a stable state is guaranteed. These studies deal only with guaranteeing convergence due to properties of the policies, rather than link or node failures and recoveries.

By selecting among multihomed connections with low loss, performance and resilience can be improved [2]. We have overlapping goals, but address the problem in the Internet at large, rather than at multihomed edge sites only, and additionally target control plane load.

Perhaps most similar to our work, flap damping [37] suppresses the use of routes which are repeatedly and quickly advertised and withdrawn. Flap damping is known to have pathological behavior that can worsen convergence [24], and as we have seen, can severely impact availability. Duan et al. [7] improve flap damping's performance by recognizing certain sequences of updates that are not indicative of route flaps, but requires an AS to advertise information about its routing policy to neighboring ASes. Recently, Huston [18] proposed delaying updates for short intervals when they match a pattern likely to indicate BGP path exploration. This introduces inconsistent state that can result in loops and outages during the period of delay. All these proposals trade availability for stability, while SRS ensures that if a valid path to the destination exists, one will remain advertised.

IX. Conclusion

This paper characterized the space of techniques for improving stability in BGP. One of our main contributions was to develop algorithms to bound the best possible points in the stability-availability and stability-deviation tradeoff spaces. Our second main contribution was the design and evaluation of a Stable Route Selection scheme. Experimental and large-scale simulation results show that SRS achieves a significant improvement in control plane overhead and data plane reliability with only a small deviation from preferred routes. However, an evaluation of deployment at a single router using direct feeds of route update traffic from Internet routers indicates a smaller improvement.

We highlight two areas in which our evaluation could be improved with future work. First, although we have shown that SRS can improve data-plane performance, our software router evaluation would benefit from much larger-scale deployments with realistic failure patterns. Second, an important and likely difficult question is how much improvement SRS could obtain using realisting route update traffic (as in our direct update feed evaluation), but with many routers running SRS.

References

[1] Abilene observatory data collections. http://abilene.internet2.edu/observatory/.

[2] A. Akella, J. Pang, B. Maggs, S. Seshan, and A. Shaikh. A comparison of overlay routing and multihoming route control. In *ACM SIGCOMM*, 2004.

[3] S. Bryant and M. Shand. A framework for loop-free convergence. In *IETF Internet Draft*, 2006. draft-bryant-shand-lf-conv-frmwk-03.

[4] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, I. Stoica, and S. Shenker. ROFL: routing on flat labels. In *ACM SIGCOMM*, 2006.

[5] H. Chan, D. Dash, A. Perrig, and H. Zhang. Modeling adoptability of secure BGP protocols. In *ACM SIGCOMM*, 2006.

[6] J. Chandrashekar, Z. Duan, J. Krasky, and Z. Zhang. Limiting path exploration in BGP. In *IEEE INFOCOM*, 2005.

[7] Z. Duan, J. Chandrashekar, J. Krasky, K. Xu, and Z.-L. Zhang. Damping BGP route flaps. In *IEEE International Performance Computing and Communications Conference*, 2004.

[8] C-T. Ee, V. Ramachandran, B-G. Chun, K. Lakshminarayanan, and S. Shenker. Resolving inter-domain policy disputes. In *ACM SIGCOMM*, 2007.

[9] A. Feldmann, H. Kong, O. Maennel, and A. Tudor. Measuring BGP pass-through times. In *Passive and Active Measurement Workshop*, April 2004.

[10] A. Feldmann, O. Maennel, Z. Mao, A. Berger, and B. Maggs. Locating Internet routing instabilities. In *ACM SIGCOMM*, 2004.

[11] W. Feng, F. Chang, W. Feng, and J. Walpole. Provisioning on-line games: A traffic analysis of a busy Counter-Strike server. In *Internet Measurement Workshop*, 2002.

[12] L. Gao and J. Rexford. Stable Internet routing without global coordination. *IEEE/ACM Transactions on Networking*, 9(6):681–692, December 2001.

[13] P. Brighten Godfrey. *Designing distributed systems for heterogeneity*. PhD thesis, University of California - Berkeley, 2009.

[14] P. Brighten Godfrey, Scott Shenker, and Ion Stoica. Minimizing churn in distributed systems. In *ACM SIGCOMM*, 2006.

[15] T. Griffin and B. Premore. An experimental analysis of BGP convergence time. In *ICNP*, 2001.

[16] T. Griffin, F. Shepherd, and G. Wilfong. The stable paths problem and interdomain routing. *IEEE/ACM Transactions on Networking*, 10(2), April 2002.

[17] K. Gummadi, H. Madhyastha, S. Gribble, H. Levy, and D. Wetherall. Improving the reliability of Internet paths with one-hop source routing. In *OSDI*, 2004.

[18] G. Huston. Damping BGP, 2007. http://www.potaroo.net/presentations/2007-12-02-dampbgp.pdf.

[19] Gianluca Iannaccone Kevin Fall, P. Brighten Godfrey and Sylvia Ratnasamy. Routing Tables: Is Smaller Really Much Better? In *HotNets*, October 2009.

[20] N. Kushman, S. Kandula, and D. Katabi. Can you hear me now?! it must be BGP. In *Computer Communication Review*, 2007.

[21] C. Labovitz and A. Ahuja. Experimental study of internet stability and wide-area backbone failures. In *Fault-Tolerant Computing Symposium (FTCS)*, 1999.

[22] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. Delayed Internet routing convergence. In *ACM SIGCOMM*, 2000.

[23] T. Li. Router scalability and Moore's law. In *Workshop on Routing and Addressing, Internet Architecture Board*, October 2006.

[24] Z. Mao, R. Govindan, G. Varghese, and R. Katz. Route flap damping exacerbates Internet routing convergence. In *ACM SIGCOMM*, 2002.

[25] Z. M. Mao, R. Bush, T. Griffin, and M. Roughan. BGP beacons. In *IMC*, 2003.

[26] D. Meyer, L. Zhang, and K. Fall. Report from the IAB workshop on routing and addressing. In *Internet-Draft*, April 2007.

[27] A. Odlyzko. Internet pricing and the history of communications. In *Internet Services, L. McKnight and J. Wroclawski, eds., MIT Press*, 2001.

[28] C. Panigl, J. Schmitz, P. Smith, and C. Vistoli. RIPE Routing-WG recommendations for coordinated route-flap damping parameters. In *Document ID ripe-229*, October 2001.

[29] Quagga software routing suite. http://quagga.net.

[30] Y. Rekhter and T. Li. A border gateway protocol 4 (BGP-4). In *RFC1771*, March 1995.

[31] Route Views project. http://routeviews.org.

[32] P. Smith and C. Panigl. RIPE routing working group recommendations on route-flap damping. In *Document ID ripe-378*, May 2006.

[33] SprintLink BGP dampening policy. https://www.sprint.net/index.php?p=policy_bgp_damp.

[34] J. Stewart. *BGP4: inter-domain routing in the Internet*. Addison-Wesley, New York, 1999.

[35] L. Subramanian, S. Agarwal, J. Rexford, and R. Katz. Characterizing the internet hierarchy from multiple vantage points. In *IEEE INFOCOM*, 2002.

[36] L. Subramanian, M. Caesar, C. Ee, M. Handley, M. Mao, S. Shenker, and I. Stoica. HLP: A next-generation interdomain routing protocol. In *ACM SIGCOMM*, 2005.

[37] C. Villamizar, R. Chandra, and R. Govindan. BGP route flap damping. In *RFC2439*, November 1998.

[38] F. Wang, N. Feamster, and L. Gao. Quantifying the effects of routing dynamics on end-to-end Internet path failures. Technical Report TR-05-CSE-03, University of Massachusetts, 2003.

[39] F. Wang and L. Gao. On inferring and characterizing internet routing policies. In *IMC*, 2003.

[40] F. Wang, Z. M. Mao, J. Wang, L. Gao, and R. Bush. A measurement study on the impact of routing events on end-to-end Internet path performance. In *ACM SIGCOMM*, 2006.

Appendix

Hardness of minimizing interruptions

In this appendix we show that minimizing interruption rate under the constraint of zero unavailability (but any amount of deviation from preferred routes) is NP-complete.

Definition 1: The decision problem MIN INTERRUPTIONS is as follows:

- **Instance:** A directed graph $G = (V, E)$; a destination node $d \in V$; for each edge, a set of times that it is available; and an integer k.
- **Question:** Does there exist a set of routes to d for each node v such that some $v \rightsquigarrow d$ route is in use whenever there is an available $v \rightsquigarrow d$ path, the routes are path-consistent (as defined in Sec. II-A), and the total number of interruptions is $\leq k$?

Theorem 1: MIN INTERRUPTIONS is **NP**-complete.

Proof: Clearly the problem is in **NP**. To show **NP**-hardness, we reduce from SAT. Given an instance of SAT with variables $x_1, \dots, x_n$ and clauses $C_1, \dots, C_m$, we construct an instance of MIN INTERRUPTIONS as follows. We create:

- a destination d;
- a node for each variable x_i;
- a node for each clause C_j;
- for each variable node x_i, two pairs of edges: $x_i \to a_i^0 \to d$ and $x_i \to a_i^1 \to d$, where a_i^0 and a_i^1 are intermediate nodes introduced only so that we can have two $x_i \rightsquigarrow d$ routes without using two edges between the same pair of nodes;
- for each variable x_i or $\bar{x}_i$ which appears in clause C_j, an edge $C_j \to x_i$.

We now set the availability of these edges as follows:

- Edges $x_i \to a_i^0 \to d$ are available during the time interval $[0, 10]$, and edges $x_i \to a_i^1 \to d$ are available during $[5, 15]$.
- If $\bar{x}_i \in C_j$, then the edge $C_j \to x_i$ is available during $[0, 10]$. Otherwise, if $x_i \in C_j$, then $C_j \to x_i$ is available during $[5, 15]$.

Finally, letting ℓ be the number of clauses with both negative and positive literals, we set $k = m + \ell + 4n$.

Having constructed the instance, we now show that there is a set of legal routes which results in $\leq k$ interruptions if and only if the SAT instance is satisfiable.

First, suppose $C_1, \dots, C_m$ can be satisfied. Set the x_i's equal to a satisfying assignment. We construct routes as follows. For the x_i nodes, note that they must use both $x_i \to a_i^0 \to d$ and $x_i \to a_i^0 \to d$ to cover the whole period of availability $[0, 15]$; the key is *when to switch* from the former to the latter. If x_i is false, we have the switch occur at time 10, so that x_i is uninterrupted during $[0, 10]$; and if x_i is true, we switch at time 5 so that it is uninterrupted during $[5, 15]$. Now consider a clause variable C_j. We have two cases:

- If C_j has both positive and negative literals, it will have an available route during $[0, 15]$. Since the clause is satisfied, it has some satisfied literal, x_i or $\bar{x}_i$. By our choice of routes for the x_i nodes, x_i will have no interruptions during either $[0, 10]$ (if the literal is $\bar{x}_i$) or $[5, 15]$ (if the literal is x_i). Note there is an edge $C_j \to x_i$, so we have node C_j use the route through x_i during this time period. During the remaining interval (either $[10, 15]$ or $[0, 5]$), it can use any of its other routes. Thus, the node C_j has one interruption at time 15 and one at either 5 or 10, for a total of 2 interruptions.
- If C_j has only negative or positive literals, it will have an available route during either $[0, 10]$ or $[5, 15]$, respectively. Following the above argument, it will have a continuously available route during this period, thus experiencing 1 interruption.

The the clause-nodes therefore encounter $m + \ell$ interruptions. There are 2 interruptions on each variable-node x_i, plus one on each a_i^0 and a_i^1. The grand total is therefore $m + \ell + 4n = k$, as desired.

Next, we assume the instance can be routed with $\leq m + \ell + 4n$ interruptions, and show that the SAT instance is satisfiable. Since the x_is and a_is have a total of $4n$ interruptions, the clause-nodes C_j must have $\leq m + \ell$ interruptions. By our construction, this can only occur if clauses with both positive and negative literals have 2 interruptions, and other clauses have 1 interruption. Without loss of generality, we can assume that all interruptions occur at time 5, 10, or 15 (since otherwise route changes can be delayed until the following link state change without increasing the number of interruptions). It follows that every C_j has a route through some x_i which is available continuously for either $[0, 10]$ or $[5, 15]$. From this it follows that we can construct a satisfying truth assignment by setting x_i to be false if x_i switches routes at time 10, or setting it to true if it switches at time 5. ■