

# Stabilizing BGP, Safely

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

## ABSTRACT

Route instability incurs significant load core routers and is widely recognized as a major contributor to data plane unreliability on the Internet. Route flap damping provides some protection against instability, but introduces pathologies and reduces availability.

Concerns about the scalability of the routing system and the increasing prevalence of real-time applications have prompted a renewed interest in stability. We believe it is time for a principled approach to stabilizing Internet routing. This paper takes a step towards that goal by characterizing the tradeoff between stability and availability, and between stability and deviation from preferred routes. We propose a new approach, *Stable Route Selection* (SRS), which uses flexibility in route selection to improve stability without sacrificing availability. Our extensive simulation and experimental results show that SRS improves stability and data plane reliability while deviating only a small amount from preferred routes.

## 1. INTRODUCTION

A number of studies 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 [10, 21, 37]. Network failures, policy changes, and the BGP convergence process itself can generate huge numbers of routing updates, causing problems in 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 [39] 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 [37]. These problems are increasingly important as the Internet is becoming an ubiquitous platform for voice and video applications. Roughly 50% of problems in VoIP calls are highly correlated with BGP updates, and BGP events have been estimated to cause 90% of VoIP call drops [19]. Internet games such as Counter-Strike have similar demands for interactivity, commonly sending periodic delay-sensitive bursts of packets [11].

In the control plane, a storm of route updates can over-

load routers, leading to processing delays, slow route convergence, and packet loss. A study of the Sprint network found that BGP processes consumed the majority of utilized CPU cycles on core routers [2]. Based on measurements of a Telstra core router, Huston [17] calculated that update processing consumed an average of 30% of a 1.5GHz processor, with peak load higher. Update load has not been solved by CPU speed advances, due to continually increasing routing table size and update rate. Instability may therefore translate to more expensive routers and higher convergence times [22]. These problems led the Internet Architecture Board Workshop on Routing and Addressing to recently identify update churn as one of the challenges for future scalability of the routing system [25].

The main mechanism for improving stability in BGP is route flap damping [36], 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. [23] 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.” [31] Other approaches to improving stability require protocol modifications [7, 23] or address narrow cases [3, 12, 15].

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?

With concerns about the scalability and reliability of the routing system prompting a renewed interest in stability, we believe it is time for a more principled method 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 upper- and lower-bounding the set of feasible points in the tradeoff spaces.

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; (2) avoiding churn by preferring stable routes; and (3) avoiding churn 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.

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 with inferred customer/provider/peer relationships and one year of link failure data from Route Views [30]. Second, we evaluate the performance of *implementable strategies* including flap damping and new route selection strategies. Our evaluation uses large-scale simulations of the BGP protocol, as well as experiments using a cluster-based deployment of a software routers.

In this way, we sandwich the set of feasible points between upper and lower bounds.

### Results

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

- *Reducing convergence overhead* can only decrease instability by 5-20% in our environment of realistic patterns of link failures. This conclusion is robust to the presence or absence of flap damping and the degree of heterogeneity of message propagation delay. However, convergence overhead can be higher due to 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 a factor of  $8.1\times$ .
- 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.
- We propose new *Stable Route Selection (SRS)* strategies which 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. These results indicate that SRS is a promising approach to safely stabilizing BGP.
- Experiments with a software router deployment show that SRS’s benefits translate to improved data plane reliability.

The rest of this paper proceeds as follows. In Section 2, we define our model of BGP, metrics, and classification of approaches to stabilizing BGP. Section 3 describes our algorithms for obtaining lower bounds in the tradeoff spaces, and Section 4 describes the upper bounds: in particular, our proposed SRS strategies. Our simulation and experimental results appear in Sections 5 and 6. We discuss related work in Section 7 and conclude in Section 8.

## 2. PRELIMINARIES

This section introduces a simple model of BGP (Sec. 2.1) and the main metrics that we study (Sec. 2.2). We then set the stage for the rest of the paper by classifying approaches to stabilizing BGP and their inherent tradeoffs (Sec. 2.3).

### 2.1 Model of BGP

In this section we describe a simple 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* a route to its neighbors.

Additionally, our model includes two important constraints. First, we assume that message propagation and routing decisions take a negligible amount of time. 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 5.

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  $v_1, \dots, v_k$ , the following are true: (1) all links  $(v_i, v_{i+1})$  are up, (2)  $v_2$  selected the path  $v_2, \dots, v_k$  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 instantaneous convergence batches. Modulo timing differences, any classic path vector routing protocol, BGP included [36], attempts to satisfy path consistency. However, in Section 2.3 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 route selection strategies and export policies based on inferred real-world business relationships.

## 2.2 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 stability, as a proxy for data plane performance and control plane CPU utilization due to its computational and analytical tractability. Our experimental results in Section 6 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.

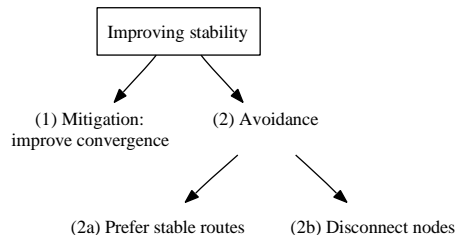
Finally, **deviation** compares a sequence  $S$  of selected paths against a *benchmark sequence* of paths  $S^*$ , and is defined as the fraction of time that the route in  $S$  “matches” the route in  $S^*$ . Deviation is intended to capture how closely a particular route selection strategy ( $S$ ) follows the network operators’ preferred paths ( $S^*$ ). Since measurements show that ASes’ routing preferences are based on next-hops for 98% of IP prefixes [38], 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.

## 2.3 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 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 undamped 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. 3 with our lower bounds algorithms. Our upper bounds, i.e., implementable strategies, include our new SRS strategies described in Sec. 4 in addition to Standard BGP and RFD. The numerical results of these lower and upper bounds appear in Sec. 5.

**Outside this classification:** Note that policies which allow path inconsistency (defined in Sec. 2.1) 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 time periods on the order of 30 sec are 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.

### 3. LOWER BOUNDS

In this section we describe how we numerically compute bounds on the maximum possible improvement that can be obtained from the above approaches to improving stability in BGP. In Section 5, 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 from the best possible policies.

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 5).
- **Approaches (1)+(2a)+(2b): Stability-Availability tradeoff.** Similarly, 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.

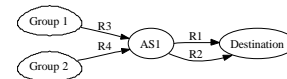
The first item, bounding convergence overhead, appears in Section 3.1 and is straightforward. In contrast, it is nontrivial to obtain good lower bounds in the tradeoff spaces. We explain why (Section 3.2) before describing the procedure (Section 3.3) which is similar for the two tradeoff spaces.

### 3.1 Convergence

Batching, described in Section 2.1, allows us to easily lower-bound the minimum number of interruptions needed to match the preferred routes almost always, i.e., with no convergence overhead. 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.

### 3.2 Hardness of minimizing interruptions

Provably providing a good bound in the tradeoff spaces is nontrivial in large part because of *dependencies between nodes* when routing to some particular destination. We illustrate this with an example for one particular point in the tradeoff spaces: that of minimizing interruption rate under the constraint of maximum availability. Consider the following topology:



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  $[5, 15]$ . Some-time 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.

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 in a reduction from 3SAT. We omit the proof.

### 3.3 Tradeoff procedure

To avoid the dependency problem we allow each node to independently select its own route, thus possibly picking a route which was not chosen by the downstream AS 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. 2.2) 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 and edges represent route choices and 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; we omit the details.

### 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 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

$$(intr(R), down(R))$$

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

given  $\lambda$ ; we vary  $\lambda$  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. 3.2, 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  $\ell_{sd}^*$  which minimizes

$$intr(\ell_{sd}^*) + \lambda \cdot down(\ell_{sd}^*). \quad (1)$$

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  $\ell_{sd}^*$  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 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  $t$  [13]. Thus, the Most Stable Paths subroutine provides (a superset of) the set of routes which might be involved in the optimal sequence,  $\ell_{sd}^*$ . We add to this set a persistently available “null route” with cost  $\lambda$  per unit time, and feed all these choices to the OPS subroutine, whose output must be an optimal set of route selections,  $\ell_{sd}^*$ .

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. 2.2), the set of routes in  $\ell_{sd}^*$  might include the route *through any neighbor* which will be available longest into the future. 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  $\lambda$  otherwise—and then feed them into OPS to produce  $\ell_{sd}^*$ .

## 4. 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 2.3). Within this approach, 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 4.1), and then how our SRS strategy selects paths (Section 4.2).

## 4.1 Fitting SRS into BGP

BGP affords a high degree of flexibility through the use of a *decision process* [33], which 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 1, which select a route based on *attributes* contained in the BGP route announcement. 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.

**Table 1: BGP Decision Process**

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

In the design evaluated in this paper, we insert the SRS heuristic as an additional step between Steps 1 and 2 of the BGP decision process.<sup>1</sup> SRS selects the best route based on a combination of Steps 2-7 and on an heuristic for predicting route stability. We present the details of SRS in the next section.

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, we will see that with Local Preferences derived from common business relationships, sufficient flexibility remains to significantly improve stability.

<sup>1</sup>In practice, SRS could be implemented by appropriately modifying route attributes, for example by applying an import filter before routes reach the decision process. This would afford the operator greater control over which steps of the decision process are affected by SRS. However, to simplify exposition, we implement SRS as a separate step in the decision process in this paper.

## 4.2 The SRS Heuristic

The SRS heuristic has one main parameter, a *delay*  $\delta$ . 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 4.1.  $pref(r_1, r_2)$  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  $\delta$ .  $SRS(0)$  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 (see [13] and references therein). In simulations, we found that inserting the shortest-path step (Step 4) often somewhat improved stability, while simultaneously providing a significant reduction in mean path length.

## 5. 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 first describe the structure and setup of our simulator in Section 5.1 and present results in Section 5.2.

### 5.1 Methodology

**Data sets:** We infer the inter-domain AS-level topology by culling AS adjacencies from Route Views [30] feeds. Since

policies on the Internet are not widely disclosed, we leverage [34] to infer and assign local preferences associated with business relationships as done in [21, 23, 35], 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 [34]: 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 an event-driven BGP simulator extended from the simulator used in [9]. The simulator’s events are at the level of link state changes and BGP update messages. Each AS is represented by a single node running a BGP instance, as in some past studies [6, 35]. 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 [29]. Since our simulator models each AS as a single router, Steps 3-6 of the decision process (Table 1) are not executed. For Step 7, we assign each AS a uniform random router ID.

We implement batching (see Sec. 2.1) 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 [36].

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 [23]. We also test with flap damping parameters used by Juniper [23], SprintLink [32], and three sets of parameters recommended by RIPE [27].

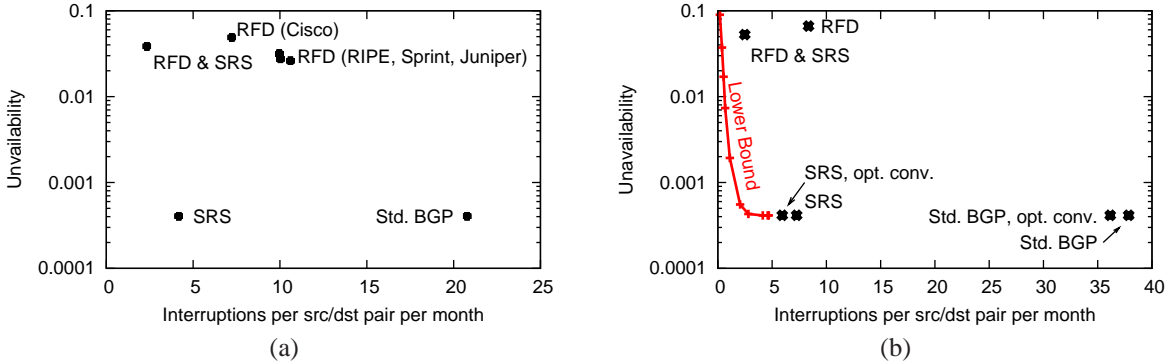
## 5.2 Results

This section presents the results of our simulation and our lower-bounds analysis 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 5.2.1).
- 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 5.2.2).
- $SRS(\infty)$  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 5.2.3).
- 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 5.2.4).
- SRS only slightly increases mean path length (Section 5.2.5), and can obtain significant improvements in stability under limited deployment scenarios (Section 5.2.6).

### 5.2.1 Effect of batching

Figure 1 shows performance of various strategies both without and with batching. Recall from Section 2.1 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. 1(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.



**Figure 1: Performance of various strategies in the stability-availability space, with batching off (a) and on (b). SRS’s delay parameter is fixed at  $\infty$ . Unless otherwise specified RFD refers to flap damping with Cisco standard parameters.**

### 5.2.2 Convergence overhead

Figure 1(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.

**Origin events:** ISPs may withdraw and announce prefixes either intentionally or due to configuration error, triggering long sequences of path hunting which are relatively rare in 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. [24]. Like [24], 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 of 1.14 while withdrawals incurred an overhead of 6.00 on average. In addition, we found that failing a single link adjacent to the destination incurred an average overhead of 3.90, while repairing the single link incurred an average overhead of 1.03.

**Policy misconfiguration:** Misconfigured ISPs may introduce oscillations and instability which can lengthen convergence periods. Here, we select a small fraction of ASes to violate our default configuration based on the Gao-Rexford policies described in [12]. We do this by selecting a random subset of ASes to “misbehave” by selecting routes based on a uniform-random preference ordering among next-hops. We found that having a fraction of ASes misbehave in this manner can increase convergence overhead; for example, convergence overhead 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 routing convergence times [20]. 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  $\alpha$ , which controls the variance of the distribution, and varying mean. Results are shown in Table 2. 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 [14], we also found that enabling MRAI increases convergence time, but reduces control overhead.

**Table 2: Effect of varying link delay.**

$\alpha$	Convergence overhead	
	mean=4 ms	mean=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

### 5.2.3 Stability-availability tradeoff

Figure 1 shows performance of various strategies in the stability-availability space. Comparing the points in Figure 1(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 a factor of 2.9, but sacrifices two “nines” of availability, and the other parameter values used by Sprint and Juniper and recommended by RIPE have substantially similar performance. One might expect the tradeoff between availability and stability to be fundamental. However, 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 2 explores the pattern of interruptions in more detail with a complementary CDF over all measured end-to-end paths. 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.

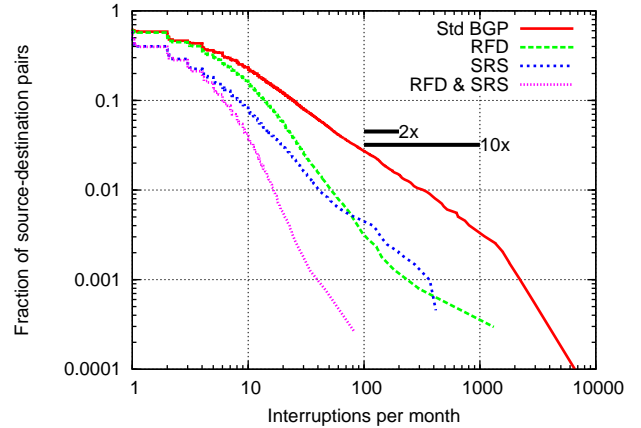
Figure 1(b) also compares performance with lower bounds on the optimal policies. SRS performs surprisingly close to optimal among strategies which achieve the highest availability, with an interruption rate just 55% higher than our lower bound, which uses knowledge of the future. Factoring out SRS’s convergence overhead, it would be just 27% worse than optimal. These results indicate that the SRS heuristic does a 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. 2.1).

Finally, the “Lower Bound” curve in Figure 1(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.

#### 5.2.4 Stability-deviation tradeoff

The above results, which set SRS’s delay parameter to  $\infty$ , assume SRS is permitted to use a large amount of flexibility in the choice of paths. 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 3(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 3(b) shows that even with batching



**Figure 2: Complementary CDF of interruptions per month over all measured source-destination pairs. SRS’s delay parameter is fixed at  $\infty$ .**

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 [26], so a deviation of less than one percent may be acceptable.

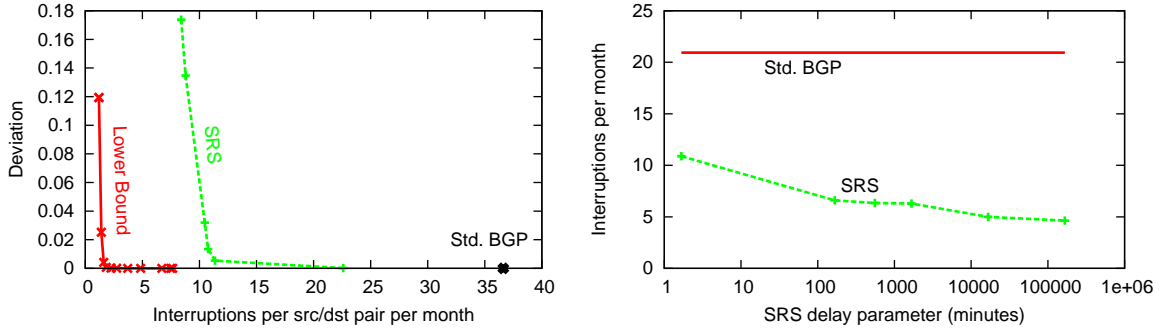
We also note there may be cases where it is useful to move beyond the knee to obtain greater stability. For example, if an ISP has the ability to route multiple classes of traffic along different routes, it would be possible to send the most stability-sensitive flows (e.g., real-time voice traffic) along 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 3(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. We believe the true optimal is actually much closer to the upper bound. Our basic lower-bounds technique in both tradeoff spaces relies on 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 algorithm does not take into account. Providing a substantially better lower bound appears to be a nontrivial problem which we leave to future work.

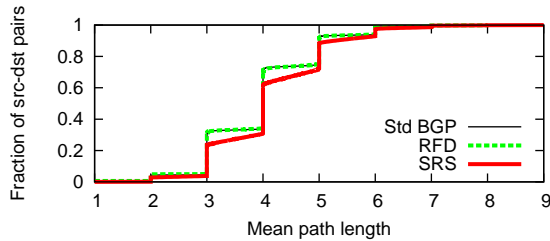
#### 5.2.5 Path length

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 4 shows the CDF of mean path length over source-destination pairs. SRS has a mean path length of 4.14 AS-



**Figure 3: (a): Performance of SRS in the stability-deviation tradeoff space with *batching on*. (b): SRS’s interruption rate as a function of its delay parameter, with *batching off*. SRS’s delay parameter varies from 100 sec to effectively  $\infty$  (a value longer than the one-month trace).**



**Figure 4: CDF of mean path length over all measured source-destination pairs.**

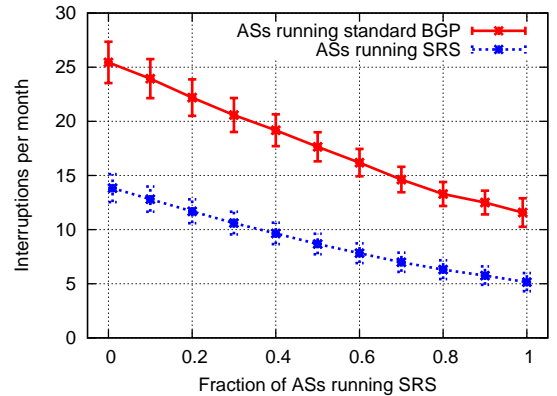
level hops, or just 4.2% greater than Standard BGP’s 3.97 hops. We note that BGP itself empirically incurs an AS-level path inflation of 49.8% itself due to policies [5]. Hence an additional inflation of 5% (resulting in a combined inflation of roughly 56.4%) may be tolerable to some ISPs, while others 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. 4). 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% longer than Standard BGP.

### 5.2.6 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.

Figure 5(a) shows the performance of SRS with delay  $\delta = \infty$  under only partial deployment, at between 1% and 100% of ASes. The set of ASes running SRS is selected randomly in each trial. We measure the interruption rate separately at those nodes running and not running SRS. The leftmost points in the plot show that the first ASes deploying



**Figure 5: Interruption rate under partial deployment of SRS, with delay parameter  $\delta = \infty$  and batching off.**

SRS would see a significant drop in their own interruption rate of roughly  $1.8\times$ . As more ASes deploy SRS, the interruption rates decrease further, so that when all ASes implement SRS, we achieve a  $5\times$  reduction in the interruption rates, as shown in previous plots. This is due to the distributed nature of BGP path selection: each AS running SRS effectively stabilizes routes on behalf of other ASes that route through them.

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 [34]), 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.

## 6. EXPERIMENTAL RESULTS

Our experimental results use 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 simulations by correlating them with observed data plane performance.

## 6.1 Methodology

Our experimental evaluation is based on the BGP implementation in the Quagga software router [28]. 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 6.

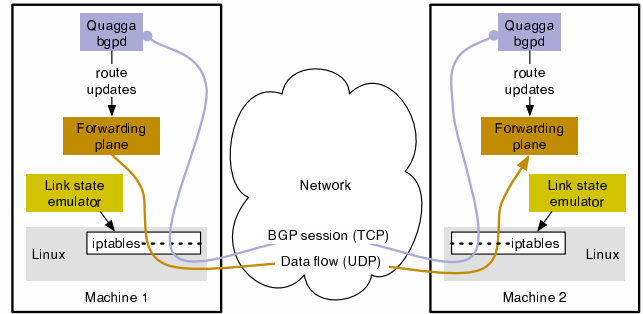
**Route selection policies:** We altered Quagga’s decision process to, with the appropriate command-line argument, run SRS with delay  $\delta = \infty$  or  $\delta = 10$  min. We also tested several strategies already implemented in Quagga: what we have referred to as Standard BGP; Quagga’s default 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. 6, 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.

**Emulating failures:** We emulate link failures by leveraging Linux’s iptables facility, with which we can block UDP and TCP traffic between pairs of software routers at appointed times. This allows us to emulate a trace of link state failures 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 than 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.

The second environment we use is a synthetic Erdős-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



**Figure 6: A diagram of our software router, showing two nodes with a flow of data from Machine 1 to Machine 2.**

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.

## 6.2 Results

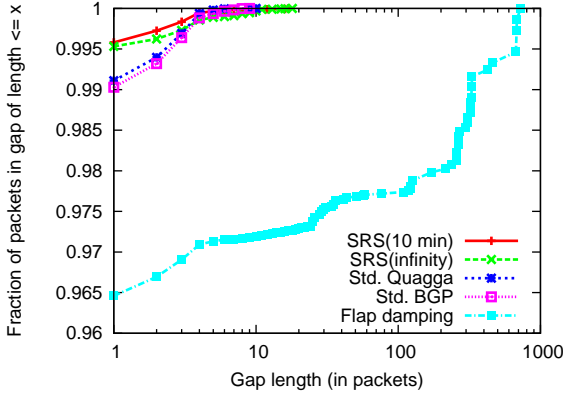
In Section 6.2.1 we show that SRS can substantially improve data plane performance over other strategies; interestingly, a delay of  $\delta = 10$  minutes performs better than  $\delta = \infty$ , potentially due to faster convergence as a result of slightly shorter paths. In Section 6.2.2 we correlate our simulator’s interruption rate metric with packet loss in the software router.

### 6.2.1 Software router performance

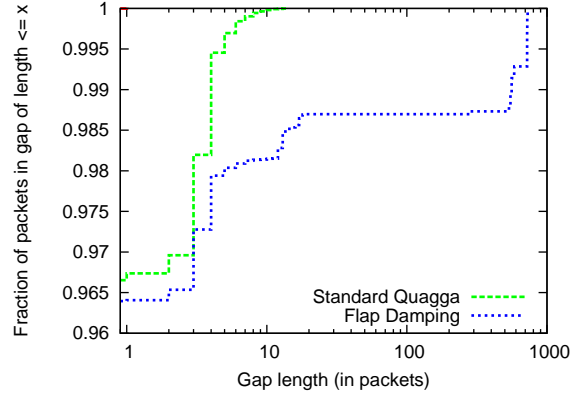
Table 3 classifies packets by their fate. The columns respectively indicate packets that were received, dropped because there was no route to the destination, sent along a virtual link which was down (i.e., dropped by iptables), dropped after exceeding the maximum hopcount, or dropped for unknown reasons (presumably, dropped by the underlying physical network).

Figure 7 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.

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 bi-



(a)



(b)

**Figure 7: Gap lengths in the software router experiments under (a) the Abilene environment and (b) the synthetic random graph environment. Packets are spaced at  $\approx 5$ -second intervals.**

Strategy	Sent	Recv	Dropped			
			No rte	Link ↓	> 30 hops	?
<b>Abilene Environment</b>						
<i>SRS</i> (10 min)	515900	513103	161	2636	0	0
<i>SRS</i> ( $\infty$ )	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
<b>Random Graph Environment</b>						
<i>SRS</i> ( $\infty$ )	810000	809981	0	0	0	19
Std Quagga	810000	782888	178	26532	43	359
RFD	810000	780812	14001	15186	1	0

**Table 3: The fates of packets in the software router experiments.**

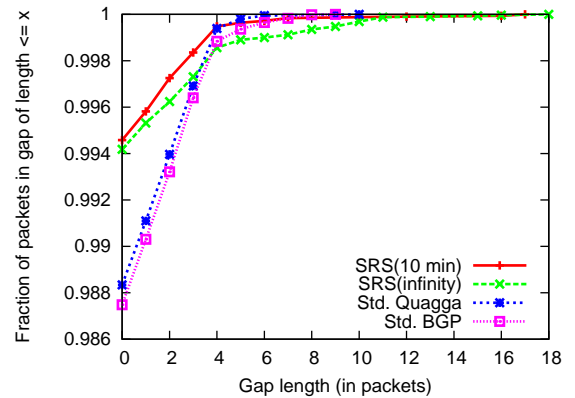
modal failure pattern.

Despite its lower overall drop rate, *SRS*( $\infty$ ) has a slightly longer tail in the gap length distribution (see Fig. 8), 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 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 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 min) presents a promising alternative.

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 7 that these cause exceedingly long periods of outage in both environments.

### 6.2.2 Correlation with simulation

The goal of this section is to determine how well our simulator’s interruption rate metric predicts data plane perfor-

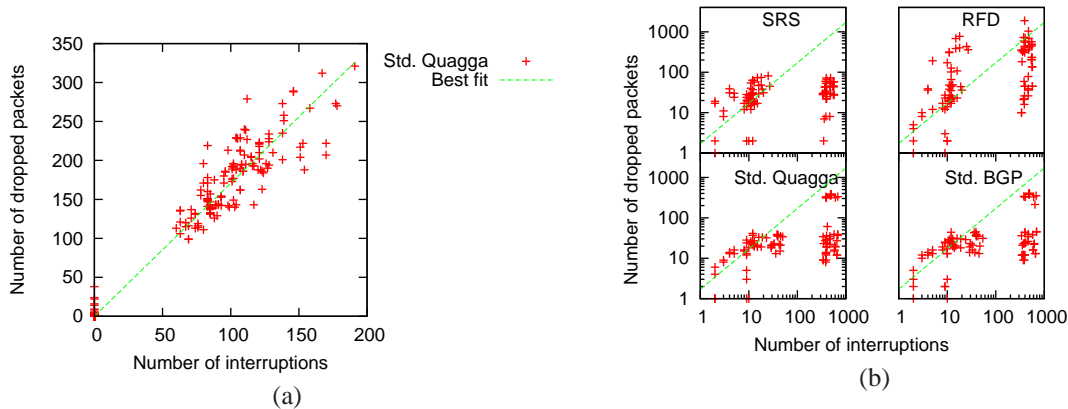


**Figure 8: Fig. 7(a), zoomed in.**

mance. 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 9(a) plots packet loss rate vs. number of interruptions for the 600 source-destination pairs (note 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  $\approx 1.7$  packets or  $\approx 8.5$  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



**Figure 9: Correlation between number of interruptions in simulations, and packet loss in experiments, for (a) the Random Graph environment and (b) the Abilene environment.**

hopcount (i.e., a loop). As long as a working path actually exists, each of these conditions 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 9(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, compared with 10 persistently 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 requires 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.

## 7. 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 [7, 23]. Loop-free convergence [4] 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 [16]. In contrast to these studies, the present paper refrains 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 [15] 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 bounding convergence overhead.

By selecting multihomed connections with low loss, performance and resilience can be improved [3]. 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 [36] 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 [23], and as we have seen, can severely impact availability. Duan et al [8] 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.

## 8. CONCLUSION

In this paper, we have characterized the space of techniques for improving stability in BGP. Our experimental and

large-scale simulation results indicate that *Stable Route Selection* strategies—preferring stable paths, rather than shutting off unstable paths—hold promise for mitigating Internet routing instability without sacrificing availability and with controllable deviation from operators’ preferred paths. We are currently pursuing a wide-area deployment of SRS in PlanetLab.

## 9. REFERENCES

- [1] Abilene observatory data collections. <http://abilene.internet2.edu/observatory/>.
- [2] S. Agarwal, C. Chuah, S. Bhattacharyya, and C. Diot. Impact of BGP dynamics on router CPU utilization. In *Passive and Active Measurement Workshop*, April 2004.
- [3] A. Akella, J. Pang, B. Maggs, S. Seshan, and A. Shaikh. A comparison of overlay routing and multihoming route control. In *ACM SIGCOMM*, 2004.
- [4] S. Bryant and M. Shand. A framework for loop-free convergence. In *IETF Internet Draft*, 2006. draft-bryant-shand-lf-conv-frmwk-03.
- [5] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, I. Stoica, and S. Shenker. ROFL: routing on flat labels. In *ACM SIGCOMM*, 2006.
- [6] H. Chan, D. Dash, A. Perrig, and H. Zhang. Modeling adoptability of secure BGP protocols. In *ACM SIGCOMM*, 2006.
- [7] J. Chandrashekar, Z. Duan, J. Krasky, and Z. Zhang. Limiting path exploration in BGP. In *IEEE INFOCOM*, 2005.
- [8] 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.
- [9] C.-T. Ee, V. Ramachandran, B.-G. Chun, K. Lakshminarayanan, and S. Shenker. Resolving inter-domain policy disputes. In *ACM SIGCOMM*, 2007.
- [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. B. Godfrey, S. Shenker, and I. Stoica. Minimizing churn in distributed systems. In *ACM SIGCOMM*, 2006.
- [14] T. Griffin and B. Premore. An experimental analysis of BGP convergence time. In *ICNP*, 2001.
- [15] T. Griffin, F. Shepherd, and G. Willfong. The stable paths problem and interdomain routing. *IEEE/ACM Transactions on Networking*, 10(2), April 2002.
- [16] 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.
- [17] G. Huston. 2005—a BGP year in review. In *21st APNIC Open Policy Meeting*, February 2006.
- [18] G. Huston. Damping BGP, 2007. <http://www.potaroo.net/presentations/2007-12-02-dampbgp.pdf>.
- [19] N. Kushman, S. Kandula, and D. Katabi. Can you hear me now?! it must be BGP. In *Computer Communication Review*, 2007.
- [20] C. Labovitz and A. Ahuja. Experimental study of internet stability and wide-area backbone failures. In *Fault-Tolerant Computing Symposium (FTCS)*, 1999.
- [21] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. Delayed Internet routing convergence. In *ACM SIGCOMM*, 2000.
- [22] T. Li. Router scalability and Moore’s law. In *Workshop on Routing and Addressing, Internet Architecture Board*, October 2006.
- [23] Z. Mao, R. Govindan, G. Varghese, and R. Katz. Route flap damping exacerbates Internet routing convergence. In *ACM SIGCOMM*, 2002.
- [24] Z. M. Mao, R. Bush, T. Griffin, and M. Roughan. BGP beacons. In *IMC*, 2003.
- [25] D. Meyer, L. Zhang, and K. Fall. Report from the IAB workshop on routing and addressing. In *Internet-Draft*, April 2007.
- [26] A. Odlyzko. Internet pricing and the history of communications. In *Internet Services*, L. McKnight and J. Wroclawski, eds., MIT Press, 2001.
- [27] 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.
- [28] Quagga software routing suite. <http://quagga.net>.
- [29] Y. Rekhter and T. Li. A border gateway protocol 4 (BGP-4). In *RFC1771*, March 1995.
- [30] Route Views project. <http://routeviews.org>.
- [31] P. Smith and C. Panigl. RIPE routing working group recommendations on route-flap damping. In *Document ID ripe-378*, May 2006.
- [32] SprintLink BGP dampening policy. [https://www.sprint.net/index.php?p=policy\\_bgp\\_damp](https://www.sprint.net/index.php?p=policy_bgp_damp).
- [33] J. Stewart. *BGP4: inter-domain routing in the Internet*. Addison-Wesley, New York, 1999.
- [34] L. Subramanian, S. Agarwal, J. Rexford, and R. Katz. Characterizing the internet hierarchy from multiple vantage points. In *IEEE INFOCOM*, 2002.
- [35] 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.
- [36] C. Villamizar, R. Chandra, and R. Govindan. BGP route flap damping. In *RFC2439*, November 1998.
- [37] 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.
- [38] F. Wang and L. Gao. On inferring and characterizing internet routing policies. In *IMC*, 2003.
- [39] 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.