Reliable Interdomain Routing Through Multiple Complementary Routing Processes

Yong Liao[†], Lixin Gao[†], Roch Guerin[§], Zhi-li Zhang[††]

ECE Department[†]
University of Massachusetts
Amherst, MA 01003, USA
(yliao,lgao)@ecs.umass.edu

ESE Department[§]
University of Pennsylvania
Philadelphia, PA 19104
guerin@ee.upenn.edu

CSE Department[††]
University of Minnesota
Minneapolis, MN 55416
zhzhang@cs.umn.edu

ABSTRACT

Maintaining reliable paths is a core responsibility of any routing system, but successfully carrying it out can be challenging, especially for inter-domain paths. As a case in point, the current *de facto* inter-domain routing protocol, BGP, commonly experiences extended periods of unreliable routing involving transient routing loops or loss of connectivity. Addressing this issue while preserving BGP's benefits, be it its flexibility in accommodating policies or its operational maturity, is a long-standing goal shared by this paper. We approach it by applying to inter-domain routing a common concept in the design of highly reliable systems, namely, the use of redundancy, which we introduce in a manner that maximizes compatibility with the existing BGP protocol. The basic idea is to run in parallel in each AS several (two) slightly extended BGP routing processes that produce *complementary* routing choices, so that in the presence of network instabilities a working path is always available to any destination. This paper provides a detailed description of the design, formally establishes its properties, and compares it to previous proposals with a similar goal. The benefits of the scheme, including as part of an incremental deployment, are demonstrated using actual BGP data and realistic simulations.

1. INTRODUCTION

With the increasing popularity of time-sensitive or interactive Internet applications such as VoIP, video streaming, on-line gaming, etc, it has become ever more important for the Internet routing system to provide "reliable" end-to-end paths. As basic as this requirement is, it has proven challenging, because the distributed nature of Internet routing decisions, something that scalability mandates, introduces unavoidable latency when reacting to network changes (such as link failures of node failures). This has been particularly evident in inter-domain routing, where the shortcomings of the *de facto* standard routing protocol, BGP, are well known [1]. For instance, BGP may take as long as 30 minutes to converge after certain routing events [2], and during those periods "transient" routing loops and loss of network reachability frequently occur. Recent measurement studies [3,4] have shown that 55% to 85% of short-lived routing failures are due to transient routing failures during BGP convergence, and that transient loops account for up to 90% of all packet losses.

Researchers have sought to address this challenge and proposed several approaches to improve inter-domain routing reliability. One approach is to speed-up BGP convergence; hence limiting the duration and thereby impact of transient routing loops and failures [5–9]. In particular, faster convergence will occur if obsolete routing information is rapidly removed across routers, e.g., by propagating additional information such as *root cause information* (RCI) that can be used to invalidate routes affected by a common failure. Another approach is to compute backup paths that can supplement the "best path" selected by BGP. As with approaches to speed up BGP convergence, enabling the selection of *good* backup paths calls for making additional routing information available. This need for additional information introduces overhead and modifications that can affect the odds of successful deployment.

In this paper, we seek to improve inter-domain routing reliability with minimal changes or added complexity to the current routing system. Our goal is to use BGP pretty much "as-is," and in particular without resorting to RCI, to preserve current operational knowledge and expertise and minimize deployment hurdles. In realizing this goal, our basic idea is to have each AS run multiple (two) very slightly extended BGP processes that exploit the AS-level path diversity of the Internet to compute *complementary* paths. Specifically, each process selects paths to ensure that across all network events that affect routing, at least one routing process maintains a "reliable" end-to-end path, i.e., a path free of routing failures or loops. In other words, the routing processes complement each other across the space of possible network events. As we demonstrate in the paper, in addition to being feasible with min-

imal changes to BGP, this approach offers protection against a broader range of routing events than existing alternatives.

Although the intuition behind computing complementary paths is straightforward, translating it into reality in the context of inter-domain routing is challenging. Not only do distributed computations have to be coordinated, but the resulting paths need to be *policy compliant*. This in itself, has been shown to be a hard problem[1]. Our goal is, therefore, to develop an approach that *allows the distributed computation of disjoint AS paths, while accommodating existing policy constraints and relying on the BGP protocol with as few changes as possible.* In tackling this problem, we first identify possible simplifications brought about by the current Internet structure and common routing policies. In particular, we establish that complementary routing solutions can be obtained by focusing only on the *"downhill"* portion of paths, i.e., the segments that extend from provider ASes to customer ASes towards the destination. This affords some simplifications, but the problem remains hard, and we introduce a simple heuristic whose performance we demonstrate through extensive experiments on the current inter-AS topology. Once complementary routes are available, it remains to specify how to *use* them, and in particular identify which one is free of problems and should be used at any one time. We propose a simple approach to this problem and argue its effectiveness.

In summary, the paper's main contributions are twofold: (i) it devises a simple and practical scheme for significantly enhancing the reliability of inter-domain routing; and (ii) it does so in a manner that leverages existing experience with BGP protocol, and which can be incrementally deployed with minimum disruption.

The rest of the paper is organized as follows. Section 2 provides background information on inter-domain routing and reviews related works. Section 3 discusses the motivation and basic design principles behind our multi-process routing scheme. Details on its design and realization are given in Sections 4 and 5. Section 6 is devoted to an extensive evaluation of the scheme and its performance. Section 7 concludes the paper.

2. BACKGROUND AND RELATED WORK

In this section we first provide a brief overview of BGP [2] and lay out our assumptions. We then introduce the transient routing problems faced by BGP, outline representative proposals for limiting or eliminating them, focusing particularly on R-BGP [11], and discuss their limitations. Other related works are also briefly touched on.

2.1 BGP and Transient Routing Problems

A basic requirement for any routing system is *safeness,* i.e., the ability to converge to a stable state for any initial state and combination of routing events. BGP is an incremental path vector protocol that accommodates a wide spectrum of routing policies, so that without constraints on their generality its safeness cannot be guaranteed [12]. In practice, however, neighboring ASes usually engage in bilateral agreements, also called *AS relationships*, which determine and constrain their routing policies. The two most common ones are i) *customer-provider* relationships where a customer AS pays a provider to transit its traffic, and ii) *peer-peer* relationships where two ASes agree to swap traffic of their respective networks (and their customer ASes) for free. Because of the economic interests defined by the relationships, ASes typically follow two common routing policies, *prefer-customer*–an AS always selects customer routes (routes learned from a customer) whenever available, and *valley-free*–an AS does not advertise provider/peer routes (routes learned from a provider or a peer) to other providers and peers. Assuming that the customer-provider relationships between ASes are *acyclic*[3], which holds in practice, the BGP protocol has been shown [13] to be *safe*, if every AS adopts these two policies. This paper assumes that these two policies are adopted by all ASes across the Internet.

The safeness of BGP notwithstanding, it only implies that routing "eventually" converges. However, during convergence affected ASes can experience *transient* loss of reachability, commonly referred to as *transient routing failures* [3]. Moreover, inconsistency in routing information across ASes during convergence can also result in *transient routing loops*. We use an example, shown in Fig. 1, to illustrate these transient routing problems. In the figure, AS 1 is the destination, with the selected path to that destination shown next to each AS. If the link between AS 6 and AS 1 fails, AS 6 loses its path and sends a withdrawal to AS 5. AS 5 will not announce its alternate path to AS 6 until its MRAI (Minimum Route Advertisement Interval) timer expires. AS 6, therefore, experiences a transient failure until AS 5 announces its alternate path. Similarly, if the link between AS 2 and AS 1 goes down, AS 2 sends withdrawal messages to AS 3 and AS 4. Both of them will then switch to their alternate paths, with AS 4 using path 4:3:2:1 and AS 3 using path 3:4:2:1. There is a transient loop between AS 3 and AS 4 until one of them announces its route to the other.

[1]Computing disjoint inter-domain paths was investigated in [10], which established that for common policy constraints computing disjoint paths in a ToR (Type-of-Relationship) graph is NP-hard unless the graph is acyclic.

[2]In this paper we focus on eBGP, which controls exchanges of routing information between ASes.

[3]Namely, the provider of any AS cannot be a customer of that AS's customer, or a customer of a customer, and so on.

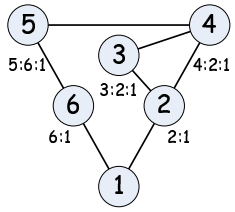


Figure 1: An example to show transient routing problems of BGP.

The relatively slow convergence of BGP after many network events, therefore results in frequent transient routing problems that are major contributors to network performance degradations [3]. Hence limiting or eliminating the impact of transient routing failures has become the focus of much recent research activity.

2.2 Existing Proposals and Their Limitations

One of the main reasons behind BGP's slow convergence is the so-called *path exploration* phenomenom [14]. As alluded to earlier, its impact can be limited through the availability of additional information such as RCI on the cause of route changes [8,9]. However, as shown in [9], correctly implementing RCI to expose path dependencies calls for careful design that can add significant complexity to BGP. For example, one approach is to annotate the *AS_PATH* attribute in BGP update messages with the ingress/egress router information of each AS on the path. Apart from the additional network bandwidth and router memory overhead, changes in RCI may also induce more routing update messages, thus increasing BGP dynamics. More importantly, these solutions in themselves do not completely mitigate the impact of transient routing problems.

Nevertheless, extensions such as RCI open the door for improvements to BGP's slow convergence, and the R-BGP scheme of [11] relies on it to eliminate transient problems *under network instability involving a single link failure*. The basic idea behind R-BGP is to preselect a *fail-over* route that is used when the primary route fails. For illustration purposes, consider the configuration of Fig. 1. Under R-BGP, each AS selects a "most disjoint" route as its fail-over route and advertises it *only* to the *next-hop* AS on its best route, e.g., AS 4 selects route 4:5:6:1 as its fail-over route and announces it to AS 2 only. In the case of single link failures, R-BGP uses RCI to expose path dependencies and avoid transient loops. For instance, when link (1:2) fails, upon receiving withdrawals from AS 2, both AS 3 and AS 4 rely on RCI to remove their alternate routes, 3:4:2:1 and 4:3:2:1, and therefore do not attempt to use them. Note though that this results in AS 3 being left with no valid routes. To "compensate" for this complete loss of connectivity induced by RCI, R-BGP allows AS 3 to continue forwarding packets along its old primary route 3:2:1, namely, to AS 2. In turn, AS 2 forwards packets along its fail-over route 2:4:5:6:1. Hence, in this scenario, R-BGP successfully eliminated transient problems caused by the single link failure.

Extending this success to other failure scenarios is unfortunately not immediate, and in particular it can be shown that R-BGP cannot handle scenarios involving the failure of multiple links *adjacent to a single AS*; a reasonably common and realistic occurrence. For example, multiple links adjacent to the same AS can fail at the same time due to the crash of a border router. Or a policy change in an AS can lead to route withdrawals to several neighbors. We illustrate this through an example using again the network of Fig. 1. Suppose that AS 2 connects to AS 1 and AS 4 via a single border router, and connects to AS 3 through another router. Suppose the former router crashes, so that *both* the primary path 2:1 *and* the fail-over path 2:4:5:6:1 are now invalid. Upon receiving withdrawals from AS 2 with embedded RCI, AS 3 invalidates all its routes but continues forwarding packets to AS 2 in conformance with R-BGP's rule. Those packets are, however, dropped at AS 2 since its fail-over route is unavailable. Hence both AS 3 and AS 2 lose connectivity to AS 1, in spite of the availability of an alternate route for both of them (3:4:5:6:1 for AS 3 and 2:3:4:5:6:1 for AS 2). Note that AS 3's inability to learn about that alternate route is caused by R-BGP's limitation to have an AS announce its fail-over route only to the next-hop on the current best route, e.g., AS 4 announces its fail-over route only to AS 2.

As another example, suppose that AS 3 and AS 4 are connected to AS 2 via a single border router in AS 2 which crashes. Upon upon detecting the event, AS 4 invalidates its primary path 4:2:1 and switches to path 4:3:2:1. Likewise, AS 3 invalidates its primary path 3:2:1 and switches to path 3:4:2:1. Therefore, despite the availability of RCI we have a transient loop between AS 3 and AS 4 until both receiving the withdrawals from each other. Hence availability of the RCI mechanism does not remove all inconsistencies, thus transient routing loops may still occur under certain common failure scenarios.

Limitations in existing solutions to exploit the rich path diversity available in the Internet topology, and more importantly the significant added cost implied by the additional mechanisms such as RCI that they require, are the principal motivations behind this paper's attempt at designing a new solution to improve the reliability of inter-domain routing. Before we proceed with the description of our proposed scheme, we complete this section by quickly mentioning a few other relevant works with a similar goal.

2.3 Other Related Work

Achieving reliable routing by using multi-path/multi-topology routing has long been a goal of network researchers. Shaikh et al. considered how to efficiently compute multiple equal cost routes in [15]. The RRL (Resilient Routing Layer) scheme is proposed in [16], in which a set of fully connected sub-topologies are created so that at least one sub-topology is still connected if any network element (link or router) is removed. In [17], the authors propose to use multiple routing configurations to achieve fast recovery from single link and node failures. The OSPF-MT (Multi-Topology) routing scheme is proposed in [18], in which each interface of a router can be configured to belong to a set of topologies. Then OSPF computes multiple topologies and finds paths to IP prefixes for each MT independently.

Various multi-path/multi-topology schemes have been proposed for routing in wireless networks. In [19], the authors proposed a routing protocol which finds two node disjoint paths using a spanning tree rooted at the destination node to construct a set of cycles which include all nodes. A centralized algorithm called *XCT* algorithm is proposed in [20] which adopts paths augmentation to compute two node disjoint paths for each node. Ramasubramanian et al. extended the *XCT* algorithm into a distributed algorithm in [21]. Besides, they demonstrated in [22] that their distributed algorithm runs in linear time in the number of links in the network.

Qiu et al. proposed the indicative re-routing in [23], but their scheme only reduces the chance of transient loop or failure. Xu et al. proposed a multi-path inter-domain routing scheme in [24]. Their focus is on how to explore the richness of connectivity in the current Internet and provide value-added services. The consensus routing proposed in [25] aims to take advantage of a distributed protocol to build a consensus view of the network. However, when network changes are in the form of node/link failures, consensus may not always be established in time to prevent transient failures.

3. MULTI-PROCESS ROUTING: BASIC DESIGN PRINCIPLES

In this section, we present key aspects of the approach we propose to improve the reliability of inter-domain routing, and establish several properties that motivate its subsequent investigation. Before we move on to these key aspects, we first introduce *routing events* that multi-process routing protocol aim to handle.

3.1 Routing Events

Routing events are the underlying network events (or "root causes") such as a link failure or recovery, BGP session reset or re-establishment, router crash or recovery, policy change. Based on the underlying causes and their manifestations, we classify routing events into three classes: a *route withdrawal event* triggers one or multiple ASes (incident to the event) to withdraw (or replace) affected routes, e.g., due to failures; a *route addition event* triggers one or multiple ASes to announce new routes, e.g., due to recovery; and a *route change event* triggers an AS to announce route updates (but no withdrawals) due to a policy change. Note that under our definition, a single route event can trigger *multiple* concurrent route updates (which may originate from different ASes). For instance, as shown by the example in Section 2.2, the crash of a border router connected to several neighbor ASes may trigger a route withdrawal from each of these ASes[4]. Instead of treating these route withdrawals as separate "failure events," we consider them as caused by a single (route withdrawal) event. Another example is a policy change within an AS (a route change event) that affects multiple neighbor ASes. An important goal of our paper is to ensure the reliable delivery of packets against all disruptions associated with a *single* routing event as defined above. Note that this is a more stringent requirement than that of previous similarly motivated proposals, e.g., R-BGP, which as illustrated earlier did not deal with some multiple correlated "route failures" triggered by a single underlying routing event.

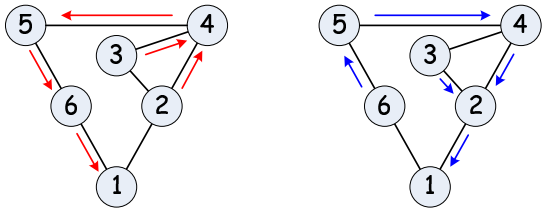
3.2 Multiple Routing Processes

Given the above set of route events, our approach to avoiding the transient loops and failures can give rise to combine a control plane mechanism and a closely related data plane mechanism.

In the control plane, we seek to provide robustness to any single network event, by identifying two distinct sets of routing decisions in each AS that are *complementary* in how they are affected by single network events. This is realized by having multiple slightly modified BGP routing processes running in parallel in each AS. Those processes can, for example, be differentiated through the use of different TCP port numbers. Paths selected by different routing processes should satisfy a key property, namely, *node disjointness*, i.e., not share any common AS nodes except the source and destination (recall that our focus is on AS-level path, with each AS a "node" in the path). This ensures that they are not affected by the same sets of events. The use of separate routing processes that select distinct (disjoint) paths, provides considerably more flexibility than relying on a single routing process that selects one best path, which must then be supplemented by finding a good fail-over path among the remaining available paths. For example, as shown in Fig. 2 and using the same AS topology as in Fig. 1, complementary pro-

[4]These ASes will detect the failure of their respective BGP session with the said router, but may not know the underlying root cause!

cesses in each AS compute two disjoint paths, the red and blue paths, to destination AS 1. The availability of these two node-disjoint paths ensures that one of them remains operational in the presence of any single node (AS) and multiple link (from the same AS) failures. In contrast, as shown in in Section 2.2, such "multiple" failures disrupt routing even when a protection scheme such as R-BGP is in use.



(a) The red paths (b) The blue paths

Figure 2: Complementary paths to the same destination.

3.3 Downhill Node Disjointness

Having stated our overall goal, namely, computing node disjoint paths, we pause to point out that realizing it while preserving BGP's policy-based distributed computations is non-trivial, i.e., recall [10]. In section 4, we present our approach based on using two essentially standard BGP processes together with some simple coordination rules within each AS. Besides computational challenges, requesting full path disjointness can also limit the choices available to routing processes running in an AS. Fortunately, full path disjointness is not necessary to realize paths that are complementary in the sense that they are not affected by the same route event. As we shall see, the hierarchical structure of the Internet together with the common policies guiding permissible routing choices, allow us to "simplify" this requirement. Before presenting this simplification, we introduce two definitions needed to characterize AS paths and establish the result.

Because of the valley-free routing policy, an AS path usually goes through a sequence of customer-to-provider links, possibly followed by a peer-to-peer link, and finally a series of provider-to-customer links. We define the *uphill portion* of an AS path as a sequence of customer-to-provider links followed by a peer-to-peer link (if it exists) and the ASes at the two ends of each link, except the AS whose next hop is its customer. The *downhill portion* of an AS path is a sequence of provider-to-customer links, together with the ASes at the two ends of each link.

Using the above definition of uphill and downhill portions of a path, we will demonstrate that complementary routing processes only have to ensure node disjointness in the *downhill portions* of their paths. Specifically, they need to produce *downhill node disjoint paths,* as formally stated below:

DEFINITION 3.1. *Two AS paths are* downhill node disjoint paths *if the downhill portions of both paths do not have any AS in common.*

3.3.1 Route withdrawals

To establish that downhill node disjointness is sufficient to ensure that the routing processes are complementary under any single routing event, we first show that it is sufficient for any single route withdrawal event.

The reason why node disjointness is needed only in the downhill portion is that under the constraints of common routing policies, network events in the uphill portion of a path will not trigger transient loops or failures during BGP convergence. In other words, a link or AS failure or a policy change in a higher tier AS (provider) does not create transient failures or loops at an AS while its BGP process adapts to the changes. This is more formally expressed in Lemma 3.1.

LEMMA 3.1. *A route withdrawal event in the uphill portion of an AS path to a destination does not produce transient routing loops or failures during BGP convergence.*

The proof of Lemma 3.1 is in Appendix A.1, but the basic ideas are as follows. If AS V uses a provider route to reach a destination p, V should have no peer or customer route to p. Hence, during a change of the route to p at AS V, e.g., after a route withdrawal, AS V cannot forward packets to p originating from its own customers "back" to lower tiers ASes because V never learned a route from them (note that since V does not have customer path to p, it will not advertise p to its peers/providers, and so should never receive a packet addressed to p from them). Hence, transient routing loops will not occur. Since V uses provider paths, the failure of V affects only V's customers who use a provider path (via V). For those customers who have another path, their providers (other than AS V) should announce that path to them before. So they will not lose connectivity.

Lemma 3.1 states that complementary routing processes only need to focus on the downhill portions of the paths they select. As a result, Section 4, which introduces our design for complementary routing processes, focuses on selecting paths that are downhill node disjoint.

Before proceeding with characterizing another set of events we need not be concerned with when it comes to their impact on routing, we highlight some implicit assumptions in the above discussion. Specifically, we have assumed that withdrawals propagate quickly, which is standard in current BGP implementations, and that

transient problems (loops or failures), therefore, arise primarily because of latency introduced by timers that delay the propagation of route announcements. We believe this to be a reasonable assumption and an accurate reflection of when and why problems do arise during BGP convergence. Hence, problems in the uphill direction will not impact routing whenever a feasible alternate exists, i.e., it will be discovered and used very rapidly.

3.3.2 Route additions and changes

We now proceed to identify two other categories of events, namely, route addition and route change events, for which it can also be established that transient problems cannot arise. Intuitively, adding a link gives BGP more choices in selecting paths, which cannot cause routing failures (note that we are considering eBGP). Provided that adding the link does not violate the cycle-free property of AS relationships, no transient loop can arise during convergence either. Similarly, an AS changing its best path selection will not result in transient failures, since all ASes still have their paths. The reason for the absence of transient loops is that the path change remains compliant with common routing policies. So that the direction (either uphill or downhill) of each AS' path does not change. We formally state those properties in Lemma 3.2. The complete proof is in Appendix A.2

LEMMA 3.2. *No transient routing loop or failure will occur after a route change event or route addition event.*

Lemma 3.2 highlights that route addition and route change events are not events one really needs to be concerned with in designing complementary routing processes.

3.4 From Control Plane to Data Plane

As mentioned earlier, our approach relies on both a control plane and a data plane component. Complementary routing processes offer the possibility of uninterrupted packet delivery in the presence of failure, by ensuring that one of the paths is always unaffected by the failure. Hence, by simply switching to the unaffected path, packets can avoid transient failures or loops. The simplicity of this statement not-withstanding, it hides several challenges beyond the design of complementary routing processes. First and foremost are the criteria that trigger switching from the current path to one computed by another routing process. According to the properties we have identified via Lemma 3.1 and Lemma 3.2, only route withdrawal events can cause transient problems. Therefore, we can embed the *"event type"* information into each update message[5] to assist in the detection of a possible failure on the current path and the identification of a complementary one. A closely coupled issue is the translation of the path switching decision into corresponding packet forwarding actions. We expand on these issues in Section 5, while the next section is devoted to our design of complementary routing processes. The discussion in the rest of this paper is limited to two complementary routing processes in each AS, a *red* process and a *blue* process that correspondingly compute *red* and *blue* paths. Those two paths must in turn be complementary under all *single* AS-level changes.

[5]Note that this calls for just "one bit" of information, which is a far cry from what is required to support RCI.

4. A MULTI-PROCESS ROUTING PROTOCOL

We first introduce our scheme, the *SelecTive Announcement Multi-Process routing protocol (STAMP),* and then formally establish its properties.

4.1 The STAMP Protocol

Each AS has two routing processes, red and blue, running in parallel. The red (or blue) process selects path among those announced by the red (or blue) processes of neighboring ASes.

As indicated in Section 3.3, our design goal for STAMP is to ensure that red and blue paths are downhill node disjoint. A straightforward approach is to have the red and blue processes locally coordinate their choices, and simply select the two most downhill node disjoint paths among the candidates they receive from neighbors. However, such "local" greedy decisions may not be good "globally". That is, a local greedy choice might result in neighboring ASes not having any downhill disjoint paths. This problem can be eliminated through the use of more sophisticated algorithms, e.g., [10,21], but this comes at the cost of significant added complexity, and more importantly major changes to the BGP protocol; something we want to avoid. Therefore, we opt for a simpler alternative, where we perform selective announcements of paths by the two processes to control their propagation, while essentially preserving the standard BGP path selection process. This ensures that we impose only minor changes to existing BGP implementations, and as we will demonstrate through analysis as well as experiments, is quite effective in achieving our goal of providing downhill node disjoint paths.

Next, we describe the rules that govern these selective path announcements. Note that since we only require disjointness in the downhill portion, path announcements to peers and customers are not selective. Therefore, selective path announcements are required only to providers. Let us first consider how a multi-homed AS (i.e., an AS with multiple providers) announces its own prefixes. The AS selects a subset of its providers as blue providers, and announces its prefixes to these providers

through its blue process only, and to rest of its providers through its red process only. This "splitting" of the colors with which the origin AS announces its prefixes to the providers ensures that the red and blue paths for each prefix reach it through different last hop providers. For a single-homed origin ASes, the splitting occurs at their first multi-homed direct/indirect provider. Note that in spite of the splitting, all prefixes are announced to all providers and only their color is affected. As we shall see, because STAMP does not impose any constraint on an AS on which colored path is its "primary" path for a prefix, the splitting of announcements on the basis of colors should have a minimal or no effect on the ability of an AS to use the same path as BGP would have selected to reach a prefix.

Having described how announcements initially proceed from the origin AS, we turn next to how transit ASes announce paths to prefixes they do not originate, i.e., receive from customers. The goal is for them to announce either red or blue routes to providers so that it is impossible for any direct/indirect provider to have both red and blue paths going through them. Clearly, if a transit AS has only red (blue) customer routes, it announces a red (blue) route to all providers. When a transit AS has both red and blue customer routes, it then has to decide which process announces to providers. One approach is to give the blue one a higher priority and announce only the blue route to providers when there are both red and blue customer routes. However, such a strict priority can severely affect the odds that a red path can propagate to all ASes. To reduce the likelihood of such an outcome, we introduce two measures. First, we require that in its initial split between blue and red providers, the origin AS selects a *single* blue provider. Second, to facilitate the propagation of red paths to as many providers as possible while ensuring the propagation of at least one blue path, we impose that this one blue provider to be a "locked" blue provider. This locking is indicated through a new path attribute, *Lock*, which takes a value of 1 when set. When the origin AS of a prefix announces it to its unique locked blue provider, it sets *Lock*=1. Upon receiving a blue path with *Lock*=1, the provider proceeds to announce this blue path to all its own providers. However, only one such announcement has its *Lock* attribute set to 1 (to the locked provider), and all others will have their *Lock*=0. Second, a transit AS that received a blue path announcement with its *Lock*=0 is not required to propagate it further if it also has a red path for the same prefix.

The main purpose of the *Lock* attribute is to ensure at least one provider propagates the blue path, and that red paths are propagated whenever a blue path is received with *Lock*=0. Of course, if an AS has only a blue path, that path will be propagated anyhow. This ensures that all customer paths are propagated to providers.

Since route exchanges between ASes are asynchronous, an AS has to decide which process announces to providers based on what routes it has received so far. Therefore, it can happen that the AS needs to backtrack it decisions. If the red routing process of the AS announced route to providers and its blue path learns a customer route with *Lock*=1. The red process should withdraw its routes from providers and let the blue process proceed (the red process does not need to withdraw routes announced to customers and peers since only announcing to providers is selective). Similarly, if the blue process of an AS has customer routes with *Lock*=0 only and latter the red process learns a customer routes, the blue process should withdraw its routes from providers and let the red process proceed.

4.2 Properties of STAMP

Having described STAMP, we next establish its safeness before showing that the two paths discovered by STAMP are *complementary*, in the sense that under any single routing event (as defined earlier), at least one of the processes does not experience transient problems. We will also present a necessary and sufficient condition for all ASes to have both red and blue routes.

To show that STAMP is safe, we first note that the only difference between a routing process in STAMP and a BGP process is that a STAMP routing process selectively announces routes to providers. Selectively announcement only limits the route announcement. As such, each STAMP routing process is safe as long as the prefer-customer and valley-free policies are followed. Therefore, the conclusion that STAMP is safe readily follows from BGP's safeness.

In exploring the complementarity of the red and blue processes, note that the two processes never announce their best routes to the same providers. Hence, if both the red and blue routing processes of an AS have paths to a prefix, the paths must be downhill node disjoint paths. Based on Lemma 3.1, red and blue routing processes are, therefore, complementary under any one route withdrawal event. Similarly, Lemma 3.2 tells us that the red and blue routing processes are complementary under any route addition or change event. Therefore, we have Theorem 4.1, whose proof can be found in Appendix A.3.

THEOREM 4.1. *Under any one routing event, the red and blue routing processes in STAMP are complementary.*

Theorem 4.1 not withstanding, we note that while selective announcement is effective in ensuring that the two processes are complementary, it does not guarantee that all ASes have both red and blue routes to a given

prefix. Nevertheless, since each AS has a locked (blue) provider, we know that a blue route must propagate to all ASes (although obviously not always as a customer route). In other words, STAMP ensures that all ASes have a blue route for a given prefix. The same guarantee, however, does not apply to red routes. This is because a locked blue route can block red routes from propagating upward to providers (and eventually reaching a tier-1 AS). Next, we formally study under what conditions ASes can be guaranteed to have both red and blue paths for a destination prefix. In Section 6.1 we perform an experimental study over today's Internet topology to evaluate the actual likelihood of ASes having both red and blue paths when using STAMP.

In order to make sure that all ASes have both red and blue routes, we need to guarantee that they have a red route. Since STAMP already announces its red routes to all customers and peers, we only need to ensure that the red route is propagated to a tier-1 AS. Therefore, we have the following Lemma:

LEMMA 4.1. *For any destination prefix, all ASes have downhill node disjoint red and blue routes, if and only if at least one tier-1 AS has a red customer route.*

Next, we derive conditions for ensuring that at least one tier-1 AS has a red customer route. First, we derive a necessary condition for a red customer route to be propagated to a tier-1 AS. From Lemma 4.1, one can see that the sub-graph between the destination AS and the tier-1 ASes is crucial in determining whether all ASes can have red and blue routes. We formally define this sub-graph as the *AS hierarchy graph* as follows. For a destination AS u, the *AS hierarchy graph of* u, $\mathcal{G}(u)$, is a sub-graph of the AS topology graph that includes only the direct/indirect providers of u and the customer-provider (directed) links between them. In order to ensure that a red route is propagated to a tier-1 AS, it is necessary that there are at least two node-disjoint paths from u to tier-1 ASes, i.e., the min-cut of the AS hierarchy graph is at least two. We formally state this in Theorem 4.2. The proof is in Appendix A.5.

THEOREM 4.2. *For any destination AS, u, if all ASes have both red and blue routes to u, then the minimum cut of $\mathcal{G}(u)$ is at least two.*

Although a min-cut of at least two is a necessary condition for all ASes to have both red and blue routes, it is obviously not sufficient. Recall that the locked blue provider is determined in a distributed manner. The wrong selection of a locked blue provider might block all red customer routes. We therefore derive next a necessary and sufficient condition for all ASes to have both red and blue routes. That is, if the locked blue providers do not block all possible paths that a red route could propagate towards a tier-1 AS, then because the red route propagation has the next highest priority after the propagation of the locked blue route, this ensures one of the remaining open paths to a tier-1 AS is discovered. Hence, all ASes will have both red and blue routes. We formally state this in Theorem 4.3. The proof is in Appendix A.6.

THEOREM 4.3. *For any destination AS, u, if u is connected with at least one tier-1 AS after removing from $\mathcal{G}(u)$ all locked blue direct and indirect providers of u, then all ASes have both red and blue routes to u.*

Intuitively, given the Internet topology, the necessary condition in Theorem 4.2 is extremely likely to hold. We confirm this for the current Internet topology in Section 6.1 (it is true for around 98.5% of destination ASes). We also expect the sufficient condition in Theorem 4.3 to be satisfied in most cases, even under random selection of locked blue providers. In fact, as shown in Section 6.1, under a random selection of locked blue providers, around 92.0% of destination ASes satisfy the sufficient condition. In addition, we will show that this percentage can be further improved to about 98.2% simply by having the origin AS "intelligently" select its locked blue provider. This is close to the maximum possible figure of 98.5%, which accounts for the fact that *irrespective* of the scheme used to select blue and red routes, it is impossible to find downhill node disjoint paths for about 1.5% of all ASes.

Before proceeding with describing in the next section how red and blue routes are used to forward packets, we would like to emphasize that although STAMP performs selective announcement, STAMP can always ensure that one of its routing process selects a customer route if there is a customer route. That is, the selective announcement does not limit the chance that an AS can use a customer route. This is because one of the routing processes always announces its route to providers. Therefore, as we will see in the next section, we will leave each source AS to determine which routing process to use when sending the packets.

5. DATA PLANE DESIGN

Once STAMP has computed routes, how these are used in forwarding packets is obviously of importance. In this section, we present a data plane design that addresses this issue.

5.1 Packet Forwarding

Under normal conditions, packets should be forwarded consistently using routes computed by routing process of the same color. For that purpose, we define a *color* bit in the packet header, indicating whether the packet should be forwarded using the red or blue routes. This color bit can be set by the source AS. The color bit might be changed by a transit AS upon detecting that

the corresponding route is currently experiencing instabilities (more on this below). Such a change should, however, be performed only once to avoid potential loops [17]. We therefore define another bit in the packet header, a *change* bit, to record whether the *color* bit of a packet has been changed or not. Two of the DS (Differentiated Services) bits in IP header could be used for that purpose.

Allowing the source AS to determine the color of its packets has a number of implications. First, transit ASes can receive packets from either color for a given destination, and must then be able to forward them using either route. There is clearly a cost to such a capability; one that goes beyond examining extra bits in packet headers and that extends to storing additional information in forwarding tables, i.e., the next hop(s) associated with routes of different colors. While this certainly represents an overhead, it can be kept relatively small through the use of intelligent data structures. Second, it is possible to ensure that whenever there is a customer route to deliver a packet, the source AS can choose the color that provides a customer route. Furthermore, it is possible to ensure that any transit AS does not have to use a provider route to deliver packets whenever a customer route exits, if the source AS selects its default route carefully. Basically, if the source AS uses peer route or customer route to reach a destination, all intermediate AS should use their customer routes, according the prefer-customer and valley-free routing policies. The source AS, therefore, does not need to do anything special in selecting which route as the default one. When both routing processes of the source AS have provider routes, the source AS first selects the "most preferred" next hop provider. For instances, both the red and blue processes of the source AS can set the local preference of one provider higher than other providers, so that both processes will select the routes announced by that provider AS. The source AS then finds the AS from which its red and blue AS paths split (let us call this AS the divergent AS). If the divergent AS's next hop AS is its provider or peer in one process and the divergent AS's next hop AS is its customer in the other process, then the source AS should pick the color of the second routing process.

5.2 Switching between Routing Processes

The previous section dealt with implementing the forwarding decisions given the routing process (or color) of the packet. However, it is possible to change the routing process used (or color) if the current process potentially experiences transient problems. In this section, we focus on how this switching is performed. Recall that our goal is to always switch to a routing process without transient problems. Although the idea is straightforward, we need to address three issues in designing such a switching mechanism; (1) detecting problematic behavior in routing processes, (2) identifying the routing process that does not experience transient problems, and (3) determining when to switch back to the original routing process.

5.2.1 Detecting potential transient problems

ASes adjacent to where a routing event originated, e.g., a link/node failure/recovery, can easily detect and identify the root cause of the event. For example, a link addition/route change will not cause transient problems but a link failure would, and the first AS heard of the event has the necessary information to tell whether the event can lead to transient problem. ASes not directly adjacent to where the routing event occurred, need to rely on update messages to infer potential transient problems. According to Lemma 3.2, transient problems arise only when a routing process loses some routes. Therefore, in order assist in "recording" if the routing event that originally triggered an update was associated with a loss of a route, we add a new path attribute ET (Event Type) to update messages. The ET attribute is 1-bit of information that indicates whether the message was caused by losing a route (ET=0) or not (ET=1).

In STAMP, updates can be sent not only because of events that affect one process, but also because of interactions between processes [6]. Rules for setting the ET bit must, therefore, account for these different scenarios. Withdrawal messages all have ET=0, irrespective of their cause. If a process generates an update message for a route because of an *adjacent* link/node failure/recovery or policy change, i.e., it is the origin AS of the event, the update message has ET=1 if it is a recovery; and ET=0 if it is a failure. If a process announces an update because itself received an update message M from one of its neighbors that resulted in a change of its best route or a new route, the process copies M's ET attribute into the update messages sent to its neighbors. Finally, if a routing process of an AS announces a route to a neighbor, because the other process withdrew its route, the update message has ET=1.

5.2.2 The switching mechanism

Given the availability of the ET attribute, we have the following mechanism for switching routes. If an AS is using the best route computed by one process and that process loses that route (receives an update message with ET=0 or the adjacent link/node fails), the AS switches to the route selected by the other process. If both processes receive update messages with ET=0, either process that still has a route can be used.

[6]For example, an AS used to announce an unlocked blue route to providers, but after learning a red customer route, it should announce this red route to providers and its blue process must withdraw its blue route.

5.2.3 Switching back to the primary routing process

Changing the packet color should be "temporarily". Once the routing process having transient problems converges and has a new path to the destination, an AS should stop changing the color of packets to that destination. Although the intuition behind this statement is simple, implementing it is not trivial because it is hard to decide when a routing process has converged or not. We adopt a heuristic in which a *switch back* timer `t` is used. If a routing process has not had dynamics for `t` seconds, we assume it has converged. Instead of setting `t` to a constant value, we can use mechanisms such as an exponential backoff algorithm to dynamically adjust `t`.

5.3 Effectiveness of the Switching Mechanism

With the above switching mechanism and the downhill node disjoint paths computed by STAMP, we can achieve reliable packet forwarding. In case of a single routing event, the only disruption in packet forwarding can be as short as the time needed for ASes adjacent to the routing event (such as a link failure) to detect that event. The effectiveness of this switching mechanism is formally established in Theorem 5.1.

THEOREM 5.1. *In case of a single routing event, no packet will be looped or blackholed after ASes adjacent to where the routing event occurs detect that event.*

The proof of Theorem 5.1 is in Appendix A.7. Here we briefly outline the basic ideas behind it. According to Theorem 4.1, for one routing event, at least one routing process in STAMP has no transient problems. Also note only losing routes creates transient problems (Lemma 3.2). If only one routing process of an AS receives updates with ET=0, the other one should not have transient problems. If both routing processes of an AS receive updates with ET=0, the routing event must either 1) occurs in the uphill portions of both red and blue paths of that AS (e.g, those two paths share a link in the uphill portions and that link fails), or 2) that AS has multiple link failures. In the first case, neither process has transient problems (Lemma 3.1). In the second case, suppose the AS has both red and blue customer routes. One of its routing process (e.g., red) must have a provider route since only one process announces to providers (here the blue one). Even if the AS loses all customer routes (the first links of both red and blue paths fail), one process still has a provider path which does not include the failed links. So the routing process that still has a route will not have any transient problem.

6. EXPERIMENTAL EVALUATION

Given that STAMP may not always succeed in discovering blue and red paths at all ASes even when they exist, we first evaluate STAMP's performance along that dimension. Next, we compare STAMP with other schemes under various failure scenarios by simulations, and finally we study its benefits in the context of partial deployments.

In order to carry out meaningful and realistic evaluations, we conduct our experiments using BGP routing tables collected by the RouteViews project [26], which we use to construct an AS topology of the Internet. We infer the underlying AS relationships using Gao's algorithm [27].

6.1 Evaluating STAMP's Performance

Given our reconstructed AS topology, we proceed to evaluate the odds for ASes to have both red and blue paths to destination prefixes when using STAMP.

6.1.1 The metric

Can STAMP ensure that all ASes have both blue and red paths to a destination? This depends on the AS topology as well as on how the locked blue provider is selected at each AS. Assume that the locked blue provider is selected randomly among all providers of an AS. Given the AS topology, we can then compute the odds that all ASes have both blue and red paths to a destination. Let Φ_m be the probability that all ASes have both red and blue routes to multi-homed AS m, and denote λ as the number of all possible paths from m to any tier-1 AS. If path $l_i, 1 \leq i \leq \lambda$, is selected as the "locked blue path" from m to a tier-1 AS and a disjoint path from m to another tier-1 AS exists, we say that l_i is a "good" locked blue path since we know that STAMP can still find a red path. If there are $\lambda^{'}$ good locked blue paths, Φ_m=$\frac{\lambda^{'}}{\lambda}$. For a single-homed AS s, Φ_s=Φ_m if m is the first multi-homed (direct/indirect) provider of s.

6.1.2 Destinations without downhill node disjoint paths

For destination AS k, if Φ_k=0, then the min-cut of $\mathcal{G}(k)$, the AS hierarchy graph of AS k, is one. That is, no matter how locked blue providers are selected, we cannot ensure that all ASes have both red and blue path to k (Theorem 4.2). The number of ASes with Φ_k=0 is shown in TABLE 1. We can see that only a small number of ASes have AS hierarchy graphs with a min-cut of one. This means that the Internet AS topology is to a large extent diverse enough to provide downhill node disjoint paths to most ASes.

	# of destinations with Φ=0	percentage
single-homed	125	0.46%
multi-homed	289	1.1%
overall	414	1.56%

Table 1: Number of destinations without downhill node disjoint paths

6.1.3 The distribution of Φ_k across all destinations

In Fig. 3, we plot the CDF (Cumulative Distribution Function) of Φ_k for all destinations. We see that less than 10% of destinations have $\Phi_k \leq 0.7$. Conversely, more than 75% of destination ASes have a probability greater than 0.9 that all other ASes can reach them through both red and blue paths. On average, all ASes have both red and blue paths to any destination AS with probability 0.92.

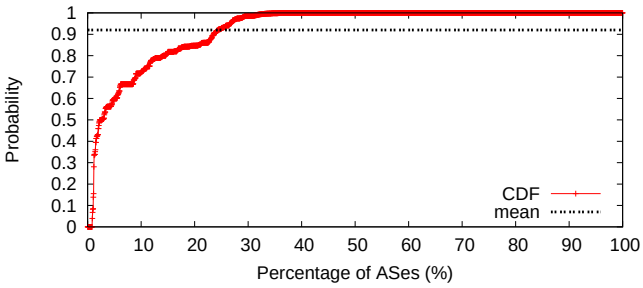


Figure 3: The CDF of Φ_k.

6.1.4 Smart selection of first hop blue provider

In the previous experiment, each AS selects its locked blue provider randomly. However, if the origin AS (or the first multi-homed provider if the origin AS is single-homed) "intelligently" selects its blue provider, the chance that all ASes have red and blue paths to that destination can be improved. Specifically, assume there are v_a paths from k to tier-1 ASes via k's provider a; and $v_a^{'}$ of them are "good" locked blue paths. The probability that all ASes have red and blue paths to k when k selects a as its locked blue provider is then $\Phi_k^a = v_a^{'}/v_a$. Assuming that AS k can compute Φ_k^a for all its providers, something that can be done off-line relatively easily and periodically since topology does not change often, it can select the lock provider u that maximizes Φ_k^u.

We investigate the benefits of such an approach in TABLE 2, which reports the number of ASes for which $\max(\Phi_k^u)$=1, i.e., are guaranteed to be reachable by all ASes through node disjoint red and blue paths. Note that while the table reports a total percentage of 97.3%, when accounting for the fact that about 1.5% of ASes cannot have two node-disjoint paths no matter how they are selected (TABLE 1), this improves to 98.8% of all ASes for which this is feasible.

	# of ASes with $\max(\Phi_k^u)$=1	percentage
single-homed	10160	38.1%
multi-homed	15794	59.2%
overall	25954	97.3%

Table 2: Number of ASes with $\max(\Phi_k^u)$=1

6.2 Performance under Failure–Comparison to Other Schemes

The previous experiments focused on evaluating STAMP's ability to provide protections again *any* single routing event/failure, which does not account for the actual impact of each possible failure scenario, e.g., some failures may not have an impact even for ASes for which STAMP did not succeed in identifying both red and blue paths. In order to better assess STAMP's actual benefits in the presence of failures, we developed an event-driven simulator to replicate routing dynamics. Our simulator is lightweight and highly efficient, which can simulate networks with thousands of ASes. We implemented BGP, R-BGP, and STAMP in the simulator. For all protocols, both processing and transmission delays are modeled by a random variable uniformly distributed in $[10ms, 20ms]$. The BGP MRAI timer is peer-based and its value is set to 30 seconds multiplied by a random factor uniformly distributed within $[0.75, 1.0]$.

6.2.1 Single link failure

We simulate routing convergence after a multi-homed AS fails one of its provider links. The destination AS is randomly selected across 100 simulation instances. The average (across all 100 scenarios) number of ASes having transient problems is shown in TABLE 3. BGP has more than 6,000 ASes experiencing transient problems. Although R-BGP handles single link failure very well, it requires RCI mechanism, which as argued earlier adds significant complexity to the routing system. Nevertheless, we include it as a benchmark against which to compare STAMP. Note that without RCI, R-BGP results in over 2,000 ASes being affected in some ways by failures. STAMP has about 350 ASes that experience transient problems. Considering that the actual Internet is likely to be more densely connected than the partial AS topology derived from BGP tables, STAMP should perform better in practice.

	BGP	R-BGP no RCI	R-BGP	STAMP
# of ASes	6604.7	2097.6	0	357.2
percentage	24.76%	7.86%	0%	1.34%

Table 3: Number of ASes having transient problems in single link failure

6.2.2 Multiple link failures

Next, we consider scenarios where multiple links fail simultaneously (or policy changes affect multiple ASes). We distinguish between two cases: i) the two failed links are not connected to the same AS; and ii) the two failed links are connected to the same AS, which corresponds the case that one router within an AS fails or an AS changes it policy resulting withdrawing route from two neighbors. In the first case, an origin AS fails one of its provider links and another randomly selected indirect

provider link (multi-hop away from the origin AS). In the second case, an origin AS fails a link to one of its providers and that provider also fails one of its own provider links.

The average number of ASes experiencing transient problems are presented in TABLE 4. When the two failed links are not connected to the same AS, both STAMP and R-BGP perform similarly, while when the two failed links are connected to the same AS, STAMP experiences about half fewer problems than R-BGP. This is because multiple link failures at the same AS correspond to a "single" routing event for STAMP; something against which its node-disjoint path selection offers protection. Note that when R-BGP is not afforded the benefit of RCI, its performance again degrades significantly.

6.2.3 *Single node (AS) failure*

The other scenario we consider is node (AS) failure, which means all links attached to that AS fail or an AS withdraws its route from all neighbors. The experiment results are presented in TABLE 5. In case of single node failure, both BGP and R-BGP have a considerable number of ASes experiencing transient problems. The reason for R-BGP's poor performance in is that R-BGP heavily relies on the provider of the origin AS to detour the traffic. If the provider AS fails, a large number of ASes will have transient problems. Here, STAMP performs similar to the single link failure because we let the origin AS "branch" its announcement to different providers so even one provider totally fails, other ASes still have another path computed by one of their routing processes.

	BGP	R-BGP no RCI	R-BGP	STAMP
# of ASes	7721.8	3376.5	2504.2	327.5
percentage	28.94%	12.7%	9.39%	1.23%

Table 5: Number of ASes having transient problems in single node failure

6.3 Evaluating Partial Deployment

The previous sections demonstrated STAMP's ability to provide most ASes in the Internet downhill node disjoint paths to most destinations with minimal impact to the operation of the current BGP protocol, and more importantly to eliminate a majority of transient problems appearing in BGP convergence. In this section, we turn to the important practical issue of incremental deployment. Specifically, the previous results assume that all ASes in the Internet deploy STAMP, but this is unlikely to occur, at least not immediately. A natural question is, therefore, to ascertain how much of these benefits remain if STAMP is deployed only in a fraction of all ASes. The natural candidates for such an initial deployment are tier-1 ASes, since they are the core of the Internet and responsible for delivering a significant percentage of the Internet traffic. We conducted a set of experiments to evaluate STAMP's "coverage", i.e., to how many destination ASes tier-1 ASes have two downhill node disjoint paths, if only tier-1 ASes adopt STAMP.

Since we assume that no customer AS feed tier-1 ASes with "colored" routes, the tier-1 ASes need to assign colors to their routes. We assume each tier-1 AS T acts according to the following rules in assigning colors to its customer routes: If T has two disjoint customer routes to a destination, T randomly assigns colors to them (one red and one blue) and announces them to other tier-1 ASes; If T does not have two disjoint customer routes to a destination, T randomly assigns a color to its best route and announces it to other tier-1 ASes.

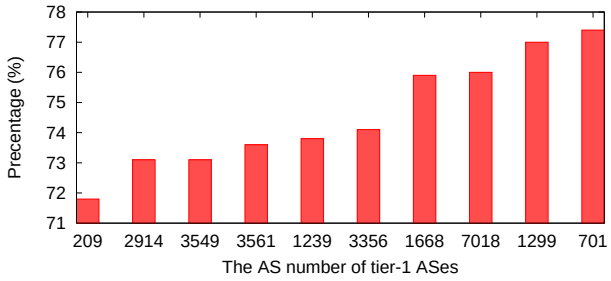


Figure 4: Percentage of non-tier-1 ASes to which each tier-1 AS have two downhill node disjoint paths under incremental deployment of STAMP (at tier-1 ASes only).

Ten ASes were selected as tier-1. For each of them, we count the number of destinations for which they have two downhill node disjoint red and blue routes. The results are shown in Fig. 4. Note that even under our simple random color assignment rule, all tier 1 ASes have two downhill disjoint paths to more than 70% of destination ASes. On average, tier-1 ASes have two downhill disjoint paths to about 75% of destination ASes.

From the simulation studies, we can conclude that STAMP performs much better than standard BGP in all failure scenarios we considered. Although R-BGP handles single link failure very well, it requires RCI information, which is not trivial to implement. Without RCI, R-BGP performs much worse than STAMP in all failure scenarios in our simulations. For multiple link failures, depending on the locations of those failed links, STAMP performs comparable or better than R-BGP. In case of single node failure, STAMP performs much bet-

(a) two failed links are not connected to the same AS

	BGP	R-BGP no RCI	R-BGP	STAMP
# of ASes	10314.5	4242.6	861.4	845.7
percentage	38.66%	15.9%	3.22%	3.17%

(b) two failed links are connected to the same AS

	BGP	R-BGP no RCI	R-BGP	STAMP
# of ASes	12071.2	3803.4	761.4	366.8
percentage	45.2%	14.3%	2.85%	1.49%

Table 4: Number of ASes having transient problems in multiple link failures

ter than other schemes.

6.4 Comparison of protocol overhead

We also use our simulator to study the protocol overhead of STAMP, compared with BGP. We still use the Internet AS-level topology mentioned in Section 6.2. In each simulation instance, we randomly select a multi-homed non-tier-1 AS p assign one prefix to it. First, p announces its prefix out after the simulation begins. We record the number of routing update messages generated during the convergence process and how long it takes for the routing system to converge. After that, we fail the link between the destination AS and one of its providers and record the number of routing update messages generated during routing convergence and the duration of the convergence process. This experiment is repeated 150 times and the average number of routing update messages and convergence time is presented in TABLE 6.

Table 6: Comparison of messages overhead and convergence delay.

(a) announcing a prefix

	# of messages	convergence delay
BGP	337,728	233.5s
STAMP	575,614	215.1s

(b) linkdown event

	# of messages	convergence delay
BGP	625,157	182.8s
STAMP	1,030,775	140.7s

From TABLE 6 we can see that the STAMP protocol generates a little less than twice the number of routing update messages generated by BGP. This result is easy to understand because the STAMP protocol has two slight changed BGP routing processes running in parallel. One way to reduce the number of routing update messages is to use a single message to include the routing update information from both red and blue process. Obviously, by doing that each update message will be larger and we need to include addition information into each message on how to differentiate between the routing update information of different processes. Even our STAMP scheme has more routing update messages, the convergence delay is actually smaller than BGP. Although we do not care much about convergence delay our STAMP scheme, because we do not need to wait for the routing system to converge.

7. CONCLUSION

The paper proposed a multi-process routing solution, STAMP, to mitigate transient problems experienced by today's inter-domain routing. STAMP seeks to accomplish this while requiring minimal changes to the current inter-domain routing protocol, BGP, and its implementations. STAMP is based on running two slightly extended BGP processes in each AS, which compute *complementary* AS routes. The paper establishes that this can be realized by focusing only on the downhill portion of AS paths, and using a simple heuristic for path selection. STAMP was evaluated through extensive experiments, which showed that compared to BGP, it could yield substantial improvements in routing stability. These improvements were comparable, and for some important failure scenarios, better than those of previous proposals that also called for more extensive and potentially complex modifications to BGP. Equally if not more important, STAMP can be deployed incrementally across the Internet, and we showed that its deployment at tier-1 ASes only could already deliver a significant improvement in Internet routing stability.

8. REFERENCES

[1] N. Kushman, S. Kandula, and D. Katabi, "Can you hear me now?!: it must be BGP," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 2, pp. 75–84, 2007.

[2] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian, "Delayed internet routing convergence," in *SIGCOMM*, 2000.

[3] F. Wang, L. Gao, J. Wang, and J. Qiu, "On understanding of transient interdomain routing failures," in *Proceedings of IEEE ICNP*, 2005.

[4] U. Hengartner, S. Moon, R. Mortier, and C. Diot, "Detection and analysis of routing loops in packet traces," in *IMW '02*. New York, NY, USA: ACM, 2002, pp. 107–112.

[5] J. Luo, J. Xie, R. Hao, and X. Li, "An approach to accelerate convergence for path vector protocol," in *Proceedings of Globecom*, 2002.

[6] D. Pei, X. Zhao, L. Wang, D. Massey, A. Mankin, S. Wu, and L. Zhang, "Improving BGP convergence through consistency assertions," in *INFOCOM*, 2002.

[7] A. Bremler-Barr, Y. Afek, and S. Schwarz, "Improved BGP convergence via ghost flushing," in *INFOCOM*, 2003.
[8] D. Pei, M. Azuma, D. Massey, and L. Zhang, "BGP-RCN: improving BGP convergence through root cause notification," *Comput. Netw. ISDN Syst.*, vol. 48, no. 2, pp. 175–194, 2005.
[9] J. Chandrashekar, Z. Duan, Z.-L. Zhang, , and J. Krasky, "Limiting path exploration in BGP," in *INFOCOM*, 2005.
[10] T. Erlebach, A. Hall, A. Panconesi, and D. Vukadinovic, "Cuts and disjoint paths in the valley-free path model," in *CAAN*, 2004.
[11] N. Kushman, S. Kandula, D. Katabi, and B. M. Maggs, "R-BGP: Staying connected in a connected world," in *Proceedings of NSDI*, 2007.
[12] T. G. Griffin and G. Wilfong, "An analysis of BGP convergence properties," *SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 4, 1999.
[13] L. Gao and J. Rexford, "Stable internet routing without global coordination," in *SIGMETRICS*, 2000.
[14] Z. M. Mao and et al, "Route flap damping exacerbates internet routing convergence," in *SIGCOMM '02*, 2002.
[15] A. Shaikh, J. Rexford, and K. Shin, "Efficient precomputation of quality-of-service routes," in *In Proc. Workshop on Network and Operating Systems Support for Digital Audio and Video*, July 1998, pp. 15–27.
[16] A. Kvalbein and A. F. Hansen, "Fast recovery from link failures using resilient routing layers," in *ISCC '05: Proceedings of the 10th IEEE Symposium on Computers and Communications*, 2005.
[17] A. Kvalbein and et al, "Fast IP network recovery using multiple routing configurations," in *INFOCOM*, 2006.
[18] P. Psenak, S. Mirtorabi, A. Roy, L. Nguyen, and P. Pillay-Esnault, "Multi-Topology (MT) Routing in OSPF," June 2007, RFC 4915.
[19] K. Ishida, Y. Kakuda, and T. Kikuno, "A routing protocol for finding two node-disjoint paths in computer networks," in *ICNP*, 1995.
[20] G. Xue, L. Chen, and K. Thulasiraman, "Quality-of-service and quality-of-protection issues in preplanned recovery schemes using redundant-trees," *IEEE JSAC*, vol. 21, no. 8, pp. 1332–1345, 2003.
[21] S. Ramasubramanian, H. Krishnamoorthy, and M. Krunz, "Disjoint multipath routing using colored trees," *Comput. Networks*, vol. 51, no. 8, pp. 2163–2180, 2007.
[22] S. Ramasubramanian, M. Harkara, and M. Krunz, "Distributed linear time construction of colored trees for disjoint multlipath routing," in *IFIP Networking*, 2006.
[23] J. Qiu, F. Wang, and L. Gao, "Bgp rerouting solutions for transient routing failures and loops," in *Proceedings of MILCOM*, 2006.
[24] W. Xu and J. Rexford, "MIRO: multi-path interdomain routing," in *SIGCOMM'06*, 2006.
[25] J. P. John and et al, "Consensus routing: The internet as a distributed system," in *Proceedings of NSDI*, 2008.
[26] "RouteViews Project," http://www.routeviews.org.
[27] L. Gao, "On inferring autonomous system relationships in the internet," *IEEE/ACM Trans. Netw.*, vol. 9, no. 6, pp. 733–745, 2001.

APPENDIX

A. FORMAL PROOFS

A.1 Proof of Lemma 3.1

PROOF. We first introduce the definition of *routing graph*. The routing graph for one destination AS p is a direct graph $G = (V, E)$. A node $v \in V$ represents an AS. A direct edge $\overrightarrow{e} = (u, v) \in E$ if v is the next hop of u to reach p. When BGP is in a stable state, the correspond routing graph is a DAG (Direct Acyclic Graph) and a node has one outgoing edge if there exists a policy compliant path for that node.

We consider the scenario where there is a route withdrawal event caused by a node (AS) failure. Route withdrawal events caused by link failure and policy change can be proved similarly. Suppose the failed AS is u and u is in the uphill portion of an AS path to reach p. Before the failure of u, the BGP system is in a stable state and the routing graph is a DAG.

Since u is the in uphill portion of an AS path, u has only provider path to p and u never announces its path to providers/peers (valley-free policy). The failure of u affect only the direct/indirect customers of u. Let C_s represent those direct single-homed customers of u and C_m represent those multi-homed direct customers of u, which have policy compliant alternate paths after u fails. The routing graph changes from G to $G^{'}$ as following after the failure of u. ASes in C_s remove their outgoing edges. Those ASes do not have alternate paths and there is nothing we can do. AS $v \in C_m$ removes it outgoing edge pointing to u and add a direct edges pointing to another provider $u^{'}$. During this process, AS v does not have transient routing failure because it still has outgoing edge. AS v does not have transient loop either, because the result routing graph $G^{'}$ is still a DAG. Suppose $G^{'}$ has a cycle $\mathcal{C}$ after removing edge $\overrightarrow{e} = (v, u)$ and adding edge $\overrightarrow{e_1} = (v, u^{'})$, we have $\overrightarrow{e_1} \in \mathcal{C}$. Since u never announces its path to providers, any AS using an path going through u must

be a direct/indirect customer of u. Since $\overrightarrow{e_1} \in \mathcal{C}$, $u^{'}$ is a direct/indirect customer of v, which is a contradiction. Therefore, $G^{'}$ must be a DAG.

Recursively, we can prove that any AS v who uses a path with u in the uphill portion will not have transient routing loop or failure, as long as there still exists a policy compliant path from v to p after u fails. □

A.2 Proof of Lemma 3.2

PROOF. We first consider route addition event caused by adding a link between two ASes. Route addition event caused by AS policy change can be proved similarly.

Suppose a link is added between AS u and AS v. Without losing generality, we consider how AS u and its neighbors reach destination p only. If AS u selects v as its next hop and v is a customer, u will announce that path to all of its neighbors. AS u will not withdraw any path. If AS u selects v as its next hop and v is a peer or provider of u, u's old path must be a peer path or a provider path. AS u will announce the new path to its customers. AS u will not send any withdrawal message to its peers and providers, since u did not announce a path to them before (valley free path policy). Since no withdrawal message is sent out, all nodes still have outgoing edges in the routing graph of p. Transient routing failure does not occur.

Now we consider transient routing loop. If u selects the path announced by v as its best path and its is a provider/peer path, u will announce that path to its customers. So, an AS $u^{'}$, which is the direct/indirect customers of u, may switch from one provider path to another provider path. According to the proof in Appendix A.1, no transient routing loop will occur when an AS switches from one provider path to another provider path. Suppose at some time the BGP system is in state $S^{'}$ and the routing graph $G^{'}$ is a DAG. For each direct/indirect provider w of u, it either switches from a provider path to a customer path via u or switches from one customer path to another customer path. For both cases, the other end of w's outgoing edge is changed from x to a customer y. The outgoing edge of AS y should point to a customer of y, otherwise it violates the valley free path policy. So w changing its outgoing edge makes the routing graph transit from $G^{'}$ to $G^{''}$ and $G^{''}$ is still a DAG, otherwise, we can derive that y is a provider of w, which is impossible. Therefore, no transient routing loop occurs.

In case of a route change event, we consider an AS u changes from a customer path to another customer path. AS u changes from a provider (or peer) to another provider (or peer) path can be provided similarly. Note that AS u cannot change from a customer path to a provider (peer) path if that customer path is still available, since it violates the prefer-customer policy.

In the initial state S the routing graph G is a DAG. The outgoing edge of u points to customer v. At some time, node u changes its outgoing edge and points to another customer $v^{'}$. Since no AS loses path, every node in the new routing graph $G^{'}$ has a outgoing edge in this new state $S^{'}$. If there is a cycle in $G^{'}$, that cycle must have edge $(u, v^{'})$ in it. Since u changes to a customer path, we can derive that u is a customer if $v^{'}$, which is not possible. So there is no cycle in $G^{'}$. Similarly, we can prove an AS changes from a provider/peer path to another provider/peer dose not create transient loop/failure either. □

A.3 Proof of Theorem 4.1

PROOF. We define the state of an AS. If an AS is selected by one of its customers at a locked blue provider, we define it is in state LB. If an AS is not selected as a locked blue provider, and it announces blue (or red) path to providers, that AS is in state B (or R).

We first consider the route addition event caused by adding a link between two ASes. Route addition event caused by policy change can be proved similarly. Suppose a link is added between AS u and AS v.

- AS u is a provider of AS v: 1) If u is in state R and v is in state B, AS u announces both red and blue (if u has one) paths to v; AS v announces a blue path to u. Both u and v do not withdraw any path so no transient problem will occur for either of them. 2) if u is in state B and v is in state R, after the link between u and v is added, u withdraws its blue path and announces the red path learned from v. It is equivalent to have a route withdrawal event in blue process and a route addition event in red process. The red process will not have any transient problems according to Lemma 3.2. 3) For other combinations of the states of u and v, neither the red process nor the blue process will withdraw any paths, adding link (u, s) is a route addition event for both routing processes. So none of them will have transient problems.
- AS v is a provider of AS u: We ignore this case because it is equivalent to the previous case.
- AS u and AS v have peer-peer relationship: In this case, neither u nor v has new customer path. Therefore, none of them will change state so that no path will be withdrawn. Both red and blue routing processes will not have transient routing problems.

Second, we consider the route withdrawal event in which an AS u fails all its links. It is equivalent to a node failure event in which node u fails.

- AS u has red customer path and blue provider/peer path: For blue routing process, the u use provider path. Therefore, u is not in the downhill portion

of any red path. According to Lemma 3.1, the blue process will not have any the failure of u will not create transient problems.

- AS u has blue customer path and red provider/peer path: This case if equivalent to the previous case.
- Both red and blue processes of u use provider/peer paths: For both red and blue processes, AS u is in the uphill portion, the failure of u will not cause transient problems.
- Both red and blue processes of u have customer paths: In this case, only one routing process of u announces to providers (assuming it is the red process). The blue process of u announces to peers and customers only. Therefore, u is not in the downhill portion of any blue path. The failure of u will not create transient problems for blue process.

One special case is that an AS u fails multiple connected links and AS u is the source AS we are considering. If at least one routing process of AS u uses provider route, that routing process will not have transient problems since the failed link must belong to the uphill of its AS path. If both routing processes of AS u use customer paths, multiple link failures can make both red and blue routing processes of AS u have route withdraw event in the downhill portion of their AS paths. If AS u still has a customer path in one of its routing process, that routing process must be free of transient problems because multiple links connecting to AS u fail is the *only* event. If both routing processes of AS u have no more customer, AS u still has no transient failure because only one routing process of AS u announces to providers. So the other routing process must also have a provider paths. The absence of transient loop for AS u (AS u is not in a loop) is based on the following fact. Suppose AS u's red process has a provider path and the red routing graph is G_r. After multiple links of AS u fail and AS u lose all customer path, AS u's red process switches to the provider path learned by its red process. Reflected in the red routing graph G_r, AS u's outgoing edge points to a customer before the failure event and its outgoing edge points to a provider after the event. If there is a loop, some customer of u (direct or indirect) must announce its provider path to another provider, which violates the valley-free policy. □

A.4 Proof of Lemma 4.1

PROOF. Since a blue path is always announced towards uphill direction, the blue process of each AS must have a blue path because announcing to peers and customers are not selective. Similarly, if the red process of a tier-1 AS has a red path, the red processes of all ASes can have a red since we do not selectively announce to peers and customers. □

A.5 Proof of Theorem 4.2

PROOF. If the min-cut of $\mathcal{G}_u$ is one, all paths from u to any tier-1 AS must contain the cutting vertex. Therefore, it is impossible to let any tier-1 AS have downhill node disjoint red and blue paths. □

A.6 Proof of Theorem 4.3

PROOF. We define the follow *"coloring"* process. The destination AS u first announces a blue route to its unique locked blue provider v and colors that provider as blue. The locked blue provider v proceeds to announce its blue route to its own providers color them as blue. This coloring process continues all tier-1 ASes have stable blue path and all direct and indirect provider of v have been colored as blue.

Then we remove all vertices among the locked blue path (except AS u) from $\mathcal{G}_u$ (the remaining graph is represented by $\mathcal{G}_u^{'}$) and start another *"coloring"* process. That is, AS u announces a red path all remaining providers and those providers proceed to color their providers. This coloring process will continue until all vertices in $\mathcal{G}_u^{'}$ are colored as red. Therefore, if there exists a path from u to a tier-1 AS in $\mathcal{G}_u^{'}$, that path must be discovered by the second coloring process. □

A.7 Proof of Theorem 5.1

PROOF. According to Theorem 4.1, after one routing event occurs(see Section 3.1 for the definition of "one" routing event), at least one routing process in STAMP has no transient problems. Also because only losing routes creates transient problems (Lemma 3.2). If only one routing process of an AS receives updates with ET=0, the other one should not have transient problems.

If both routing processes of an AS receive updates with ET=0, the routing event must either 1) occurs in the uphill portions of both red and blue paths of that AS (e.g, those two paths share a link in the uphill portions and that link fails), or 2) that AS has multiple link failures. In the first case, neither process has transient problems (Lemma 3.1). In the second case, from the proof of Theorem 4.1 in Appendix A.3, we can see that at least one routing process of the AS should still have a route. Besides, that route is free of any transient problems because the only event is multiple links connecting the AS have failed. Any invalid routes should be removed from that AS's routing table after it detects the link down event. □