R-BGP: Staying Connected In a Connected World

Nate Kushman nkushman@mit.edu
Srikanth Kandula kandula@mit.edu
Dina Katabi dk@mit.edu
Bruce M. Maggs bmm@cs.cmu.edu

Abstract

Many studies show that, when Internet links go up or down, the dynamics of BGP may cause several minutes of packet loss. The loss occurs even when multiple paths between the sender and receiver domains exist, and is unwarranted given the high connectivity of the Internet.

Our objective is to ensure that Internet domains stay connected as long as the underlying network is connected. Our solution, R-BGP works by pre-computing a few strategically chosen failover paths. R-BGP provably guarantees that a domain will not become disconnected from any destination as long as it will have a policy-compliant path to that destination after convergence. Surprisingly, this can be done using a few simple and practical modifications to BGP, and, like BGP, requires announcing only one path per neighbor. Simulations on the AS-level graph of the current Internet show that R-BGP reduces the number of domains that see transient disconnectivity resulting from a link failure from 22% for edge links and 14% for core links down to zero in both cases.

1 Introduction

It has long been known that during convergence, BGP, the Internet interdomain routing protocol, causes packet loss and transient disconnectivity. For example, Labovitz et al. show that a route change generates, on average, 30% packet loss for as long as two minutes [21]. Wang et al. report that a single routing event can produce hundreds of loss bursts, and some bursts may last for up to 20 seconds [34]. Both popular IP addresses with a lot of traffic as well as unpopular addresses suffer temporary disconnectivity because of BGP dynamics [24, 30]. Furthermore, BGP causes much of the lasting transient failures that affect Internet usability; our recent paper [20] shows that half of VoIP outages occur within 15 minutes of a BGP update.

BGP often loses connectivity even when the underlying network continuously has a path between the sender and the receiver. Indeed, in the above studies, the underlying network continuously has such a path. The Internet topology is known for its high redundancy, even when considering only policy compliant interdomain paths [13, 35]. Hence, transient disconnectivity due to protocol dynamics is unwarranted. The objective of this work is *to ensure that Internet domains are continuously connected as long as policy compliant paths exist in the underlying network.*

Past work in this area has focused purely on shrinking convergence times [9, 18, 28, 33]. Such approaches, however, are intrinsically limited by the size of the Internet and the complexity of the BGP protocol.

We take a fundamentally different approach. We focus on protecting data forwarding. Instead of trying to reduce the period of convergence, we isolate the data plane from any harmful effects that convergence might cause. Specifically, while waiting for BGP to converge to the preferred route, we set the data plane to forward packets on pre-computed failover paths. Thus, packet forwarding can continue unaffected throughout convergence.

Our failover design addresses two important challenges:

(a) Ensuring Low Overhead: The size and connectivity of the Internet make a naive advertisement of alternate failover paths unscalable. Announcing multiple paths to each neighbor could lead to explosion of the routing state, and announcing even a single failover path per neighbor could lead to excessive update traffic. Instead, in our design, a domain announces only one failover path to one strategic neighbor.

(b) Guaranteeing Continuous Connectivity: The real difficulty in using failover paths lies in ensuring connectivity while progressing from the failover state to the final converged state. Inconsistent state across Internet domains can cause forwarding loops, or lead domains to believe that no path to the destination exists even when such a path does exist. We address the consistency problem by annotating BGP updates with a small amount of information that prevents transient routing loops and ensures that forwarding is never updated based on inconsistent state.

Our solution, R-BGP (Resilient BGP), needs only a few simple and practical changes to current BGP. It precomputes a few strategically chosen failover paths and maintains enough state consistency across domains to ensure continuous path availability. R-BGP has these properties:

- Two domains are *provably* never disconnected by the dynamics of interdomain BGP, as long as the underlying network has a policy compliant path between them.
- Like BGP, R-BGP advertises only one path per neighbor, and thus the number of updates it produces is on a par with BGP.

We evaluate R-BGP using simulations over the actual Internet AS topology. Our empirical results show that, when a link fails, R-BGP reduces the number of domains temporarily disconnected by the dynamics of interdomain BGP from 22% for edge links and 14% for core links down to zero in both cases. Even in the worst case when multiple link failures affect both the primary and the corresponding failover path, R-BGP avoids 80% of the disconnectivity seen with BGP. Furthermore, R-BGP achieves this performance with message overhead comparable to BGP and reduced convergence times.

Ensuring that routing protocols recover from link failures with minimal losses is an important research problem with direct impact on application performance. Significant advances have been made in supporting sub-second recovery and guaranteed failover for intradomain routing in both IP and MPLS networks [11, 27, 31, 32], but none of these solutions apply to the interdomain problem. The main contribution of this paper is to provide immediate recovery guarantees for interdomain routing, using scalable, practical and provably correct mechanisms.

2 BGP Background

The Internet is composed of multiple networks, called domains or autonomous systems (ASes). ASes use the Border Gateway Protocol (BGP) to exchange interdomain information on how to reach a particular address prefix. A BGP update contains the full path to the destination expressed in terms of AS-hops. Each BGP router selects the best route for each destination prefix, called the *primary route*, and *advertises only its primary path* to its peer routers, sending them advertisements *only when routes change*.

BGP is a policy-based protocol. Rather than simply selecting the route with the shortest AS-path, routers use policies based on commercial incentives to select a route and to decide whether to propagate the selected route to their neighbors. The policies are usually guided by AS relationships, which are of two dominant types: customer-provider and peering [13]. In the former case, a customer pays its provider to connect to the Internet. In peering relationships, two ASes agree to exchange traffic on behalf of their respective customers free of charge. Most ASes follow two polices for routing traffic: “prefer customer” and “valley-free”. Under the “prefer customer” routing policy, an AS always prefers routes received from its customers to those received from its peers or providers. Under the “valley-free” routing policy, customers do not transit traffic from one provider to another, and peers do not transit traffic from one peer to another.

Finally, BGP comes in two flavors: routers in different ASes exchange routes over an eBGP session, whereas routers within the same AS use iBGP.

3 Related Work

Most prior work on improving BGP performance addresses control plane issues, such as reducing convergence time and the number of routing messages [9, 18, 28, 33]. A common tactic is to prevent paths that are not going to be useful from being explored during convergence. For example, BGP-RCN [29] augments a BGP update with the location that triggered the update, which, upon a withdrawal, saves BGP from exploring paths that also traverse the same troubled location, and hence are likely to be down. Another proposal [9] sends additional withdrawal messages to purge stale information from the network as quickly as possible. Our approach differs from the above prior work because instead of worrying about shrinking the convergence time, it protects data forwarding from the harmful effects of BGP convergence by providing failover paths.

Prior work on failover paths is mainly within the context of *intradomain* routing. For example, in MPLS networks, it is common to use MPLS fast re-route which routes around failed links using pre-computed MPLS tunnels [27]. In IP networks, there are a few proposals for achieving sub-second recovery when links within an AS fail [11, 22, 32], this work does not extend to the interdomain context because it assumes a monotonic routing metric, ignores AS policies, and requires strict timing constraints.

Lastly, prior work on failover paths in the interdomain context does not provide a general solution for continuous connectivity. The authors of [8] have proposed a technique for immediate BGP-recovery for dual-homed stub domains. Their solution, however, does not generalize to other types of domains, and requires out-of-band setup of many interdomain tunnels. Also, the authors of [35] propose a mechanism that allows neighboring ASes to negotiate multiple BGP routes, but in contrast to our work they do not use these routes to protect against transient disconnectivity or provide connectivity guarantees.

4 Why Not Reduce Convergence Time?

There are two broad ways to address packet loss caused by BGP convergence: limit how long BGP convergence lasts, or ensure that BGP convergence does not cause packet-loss. Much prior work has focused on the former approach, i.e., shrinking BGP’s convergence times [9, 18, 28]. We chose to explore the second approach for two reasons:

(a) Fast enough convergence is unlikely: Given the size of the Internet, it is difficult, if not impossible, to design an interdomain routing protocol that converges fast enough for real-time applications. The convergence time of BGP is limited both by the time it takes routers to process messages, and by rate-limiting timers instituted to reduce the number of update messages. There’s an inherent trade-off between these two however: set the timers too low and convergence is limited by the time it takes routers to process the additional updates; set the timers too high and convergence is limited by the time it takes for the timers to expire. Currently, routers use a timer called the Minimum Route Advertise Interval (MRAI) to limit the time between back-to-back messages to the same neighbor for the same destination to 30 seconds, by default. Griffin et. al. [14] show that we cannot expect a net gain in convergence time by reducing this default, yet real-time applications such as

VoIP and games cannot handle more than a couple seconds of disconnectivity, thus even a single MRAI of outage can be devastating for them [17].

(b) Focus on convergence limits innovation: Imposing strict timing constraints on convergence stifles innovations in interdomain routing. For example, it is desirable for interdomain routing to react to performance metrics by moving away from routes with bad performance [35]. However, such an adaptive protocol will likely spend longer time converging because it explores a larger space of paths and changes paths more often. A mechanism to protect the data-plane from loss during convergence will be a key component of any research into richer routing and traffic engineering options.

The rest of this paper details the problem and presents R-BGP (Resilient BGP), a few simple and practical modifications to BGP that ensure continuous AS-connectivity. We present R-BGP in the context of a single destination; this keeps the description simple but also complete, since BGP is a per-destination routing protocol. Further, we first describe the problem and solution at the AS level, referring to each AS as one entity and ignoring router-level details. Then in §7, we discuss router-level implementations.

5 Transient Disconnectivity Problem

Ideally, when a link fails, the routing protocol would immediately re-route traffic around it. But, in reality, BGP takes a long time to find another usable route, creating a *transient disconnectivity*, during which packets are dropped.

Consider the example in Fig. 1, where MIT buys service from both Sprint and a local provider Bob, who in turn buys service from AT&T and Joe. Traffic sent to MIT flows along the dashed arrows in the figure.

In BGP, an AS advertises only the path that the AS uses to reach the destination. Since all of Bob's neighbors are using his network to route to MIT, none of them announces a path to Bob, and Bob knows no alternate path to MIT. Thus, if the link between Bob and MIT fails, Bob can no longer forward packets to MIT and has to drop all packets, including those from AT&T and Joe. Eventually, Bob will withdraw his route to MIT, resulting in AT&T advertising the alternate path through Sprint. Now that Bob knows again a path to MIT, it resumes packet forwarding.

This is an example of a transient disconnectivity. Specifically, Bob has suffered temporary disconnectivity to MIT even when the underlying AS-graph contains an alternate path. Note that the harm of transient disconnectivity is not limited to the AS without a path. In this example, AT&T also suffers transient disconnectivity to MIT even though it knows of an alternate path.

In practice, transient disconnectivity is typical whenever routes change, and might last for a few minutes [21, 34]. This delay stems from several causes. First, a usable path might not be available at the immediate upstream AS

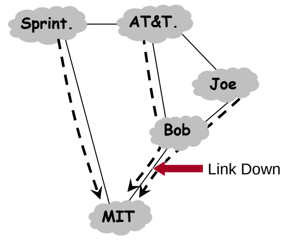


Figure 1: Transient Disconnectivity: When the link between Bob and MIT goes down, packets to MIT that are sent from Bob's domain and also those from upstream domains will be dropped until the alternate path through Sprint is advertised and BGP re-converges.

forcing the withdrawal to percolate through many ASes until reaching an AS that knows an alternate path. Further, searching for a usable path involves discarding many alternatives. For example, AT&T might first switch to the customer route "Joe→Bob→MIT" but will have to discard it when Joe reacts to Bob's withdrawal by withdrawing his route. Finally, this delay is exacerbated because a link failure creates a flurry of update messages for all destination prefixes that were using the link, thus delaying processing at nearby routers [7].

6 The Design of R-BGP

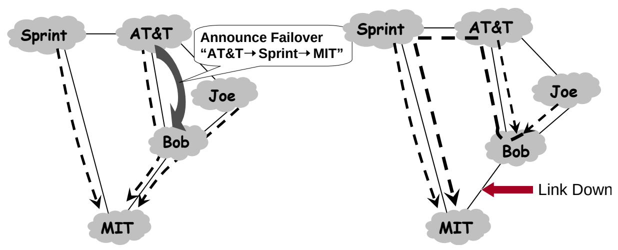
Our goal is ambitious; we want to ensure continuous connectivity between any two ASes as long as the underlying AS graph is connected. More specifically, our aim is for R-BGP to provide:

Continuous Connectivity: *If an AS has a policy compliant path both before and after a BGP routing event, then the AS should not become disconnected from the destination at any time during convergence due to the dynamics of interdomain BGP (eBGP).*

To achieve our goal, we use failover paths. Failover as an idea to solve the transient disconnectivity problem is conceptually simple; instead of searching for a new path to the destination after a link fails, as is the case in current BGP, pre-compute an alternate path *before* the link fails.

For example, in Fig. 1, with current BGP, Bob drops MIT's packets when its link to MIT fails. During BGP convergence, Bob learns of Sprint's path to MIT, uses it, and stops dropping packets. In contrast, in a failover solution, AT&T advertises to Bob the path "AT&T→Sprint→MIT", labeled as a failover path, as shown in Fig. 2a. As long as Bob's link to MIT is operational, the failover path is not used, and traffic follows standard BGP; but, if the link between Bob and MIT fails, Bob immediately diverts MIT's traffic to the failover path, i.e., he diverts the traffic to AT&T, who will forward it along the failover path to Sprint. As shown in Fig. 2b, the failover path saves Bob, Joe and AT&T from experiencing transient disconnectivity to MIT.

The conceptual simplicity of the failover idea hides three significant challenges.



(a) Announce Failover Path (b) Forward onto failover path

Figure 2: Failover Paths in Action: AT&T announces a failover path to Bob. When the link Bob→MIT goes down, Bob immediately forwards data onto the failover path ensuring all packets, even those from Joe and AT&T are not dropped.

- How to select and disseminate failover paths that ensure *continuous connectivity* without undue overhead?
- How to prevent inconsistent state across ASes from leading to transient loops during convergence?
- How to know that we have converged to the final state, i.e., we can stop using failover paths?

In the following three sections, we explain these challenges and the mechanism we use to address each of them.

6.1 Advertising Only a Few Failover Paths

The Internet is highly connected, with many alternate paths between a pair of ASes. Naively announcing failover paths can easily lead to an explosion of routing messages and state overhead in routers. How do we find a few strategic failover paths that achieve our goal of continuous connectivity regardless of which link fails?

6.1.1 To Whom Should Failover Paths Be Advertised?

In practice, a given AS is always incented to advertise a failover path to the neighboring domain through whom it is routing, because if this neighboring domain is left without an available path, it will drop the given AS's packets. ASes are less incented, however, to offer failover paths to other neighbors. In Fig. 2, AT&T has an incentive to advertise a failover path to Bob because if it does not, Bob may be left without a path and drop AT&T's packets to MIT. It is less incented, however, to offer a failover path to its competitor Sprint. Thus, with *R-BGP, an AS advertises at most one failover path per destination and only to the next-hop domain along its primary path*.

Advertising failover paths adds little update message overhead because each domain advertises at most one path to each neighbor, just like current BGP. To see this, recognize that an AS should not advertise its best path to the neighbor currently used to reach that destination, since this path would be a loopy (unusable) path to this neighbor. BGP's *poison-reverse* policy ensures that a withdrawal be sent in this case. R-BGP replaces this poison-reverse withdrawal with an advertisement of the failover path, keeping the overhead at a minimum.

The above rule also simplifies the implementation of failover paths in a way that is secure and has little forwarding overhead. If an AS were to announce multiple paths to a neighbor, it will need an additional signaling mechanism, such as marking the packets, to identify which path to use to forward packets coming from that neighbor. Such signaling mechanisms can be expensive, as the IP header has no free bits, and may require additional security mechanisms. Since R-BGP offers the failover path only to the neighbor used to reach the destination, only packets from this neighbor are forwarded on the failover path. This requires no additional signaling, and no additional security mechanisms to prevent abuse. Finally, though the failover path advertisement rule may appear too restrictive, we will prove that it is enough to achieve *continuous connectivity*.

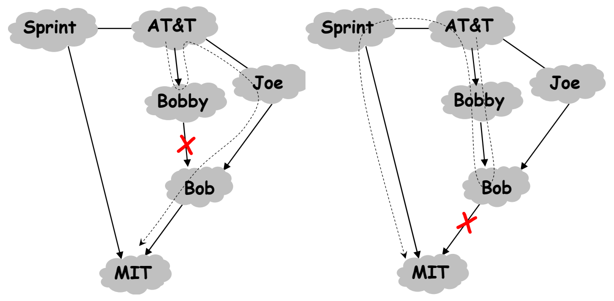
6.1.2 Which Failover Path to Advertise?

The above rule specifies the neighbor to which an AS advertises a failover path; it does not, however, answer the question–which failover path to choose in order to ensure continuous connectivity?

At first, it might seem that an AS should advertise the second-best route as a failover path, but it is likely that there is significant overlap between the primary and the second-best path, causing both paths to be unavailable at the same time. Consider the modified scenario in Fig. 3, where we inserted a new ISP, Bobby, between AT&T and Bob. Solid arrows represent primary paths to MIT and dotted lines represent failover paths. In this new scenario, AT&T's primary path goes through Bobby then Bob. Its second-best path is via Joe because ASes usually apply a "prefer customer" policy, and Joe is AT&T's customer whereas Sprint is AT&T's peer. Assume AT&T advertises its second-best path to Bobby, its next-hop AS on the primary, as a failover path. This failover path is useful if the link between Bobby and Bob fails, in which case Fig. 3a shows the path taken by the packets after the failure. The second-best path does not help, however, if the link between Bob and MIT fails. If AT&T had advertised the path via Sprint instead, it would have been possible to protect against either of the two failures. This leads us to the simple intuition – the more disjoint the failover path is from the primary, the more link failures it can protect against. Thus, our strategy for failover paths is:

Mechanism 1 - Failover Paths: *Advertise to the next-hop neighbor a failover path that, among the available paths, is the one most disjoint from the primary.*

We note a few important subtleties. First, a domain must check all paths it knows, *including failover paths*, to pick the one most disjoint from its primary. For example, in Fig. 3b, Bobby's most disjoint path is the failover path he learned from AT&T, and hence he advertises this path to Bob. Note that it may not be policy compliant to advertise this path to the next-hop neighbor. The failover path, however, will only be used for a short period during



(a) Second Best as Failover (b) Most Disjoint as Failover

Figure 3: Which failover path to advertise? Choosing the path that is most link-disjoint from the primary path makes it less likely that a link failure will take down both the primary and the failover paths.

convergence, and it is used to guarantee connectivity to the advertising AS. Thus, we believe most ASes are willing to advertise such paths. Regardless, we show experimentally, that even using only policy compliant paths still eliminates most transient disconnectivity.

Second, note that the disjointness of two paths is defined in term of their shared suffix. In particular, any destination based routing protocol, including BGP, creates a routing tree to reach the destination. Once two paths going to the destination meet at an AS, they do not diverge again because no AS will announce multiple routes to the same destination. This means that at convergence two paths to the destination can only have a common suffix. The smaller the length of this common suffix, the less likely the two paths will fail simultaneously. Lastly, if multiple paths are equally disjoint from the primary path, then the normal BGP algorithm is used to choose between them.

6.1.3 Is This Enough?

The mechanism from the previous section maximizes connectivity while advertising only a few failover paths. But is this enough? Will every AS know a failover path for every link that can fail?

Let us go back again to the Bob-AT&T example introduced in Fig. 2 at the beginning of this section. When the link between Bob and MIT fails, Bob knows a failover path through AT&T. But Joe does not know any failover path; neither AT&T nor Bob sends through Joe, and thus none of them offers Joe a failover path.

We claim that it is not necessary for every AS to know a failover path for every link that can fail in order to achieve our goal. In fact, it suffices if *each AS is responsible only for the links immediately downstream of it*. The intuition for this is simple: if the AS immediately upstream of a failed link knows a failover path, packets of all upstream ASes are automatically protected. In the above example, as long as Bob knows a failover path for when the link Bob-MIT goes down, Joe's packets will see no loss. Further, in this example Joe is responsible for knowing a failover path only if the link between Joe and Bob fails, and AT&T is responsible for knowing a failover path only if the link between AT&T and Bob fails.

Thus, in order to achieve the first step in ensuring *continuous connectivity*, we need only show that Mechanism 1 ensures the AS immediately upstream of the down link always has a failover path on which to send packets. We show this by proving:

Lemma A.4. *If any AS using a down link will have a path to the destination after convergence, then R-BGP guarantees that an AS that is using the down link and adjacent to it knows a failover path when the link fails.*

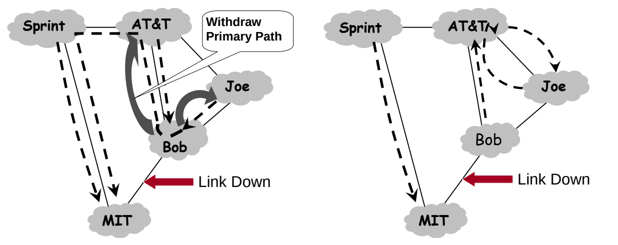
The formal proof is in the appendix, but the intuition is simple. Let Bob be the AS immediately upstream of the failing link. One of two cases will apply.

- *An AS upstream of Bob knows of a path p that does not use the failed link:* In this case, the most disjoint path at this upstream AS must not contain the down link because any path that contains the down link has a longer common suffix with the primary path through Bob and consequently is less link disjoint than *p*. As the failover path percolates from the upstream AS toward Bob, it can be replaced only with *more* disjoint paths, which necessarily do not contain the down link. Hence, Bob will be advertised a failover path that does not traverse the failed link.
- *No AS upstream of Bob knows of paths that do not use the failed link:* In this case, no AS using the down link knows an alternate path to the destination, and since ASes that do not use the down link will not change their advertisement as a result of the link going down, Bob and all other ASes using the down link will not have an alternate path, even after convergence.

6.2 Converging Without Routing Loops

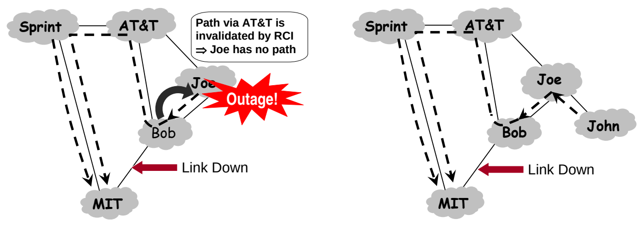
We have shown that announcing most disjoint failover paths guarantees that, whenever a link goes down, the AS immediately upstream of the down link knows a failover path and can avoid unnecessary packet drops. We now focus on the aftermath; specifically how to ensure that no AS *unnecessarily loses connectivity at an intermediate stage of convergence* either due to routing loops (§6.2) or because a usable path is no longer available (§6.3).

To see how a routing loop can be formed during convergence, we re-visit the previous example. In Fig. 4a, the link Bob→MIT is down and Bob has switched over to the failover path through AT&T. Now, Bob has no path to announce to AT&T or even to Joe because with normal BGP policies the path through one provider (AT&T here) is not advertised to another provider. Hence, Bob withdraws his route to MIT as in Fig. 4a. Unfortunately, these BGP updates do not indicate the reason for the withdrawal; so both AT&T and Joe believe that the other might be still



(a) Bob withdraws path (b) AT&T and Joe route via each other

Figure 4: Example of a Routing Loop



(a) Without Mechanism 3, Joe drops packets when Bob withdraws his path (b) Bob doesn't send John a withdrawal before he hears from AT&T

Figure 5: Avoiding disconnectivity while moving from a failover state to a converged state

be able to route through Bob, and thus together they mistakenly create a routing loop. In short, AT&T attempts to route along "Joe→Bob→MIT" while Joe attempts to route along "AT&T→Bob→MIT" causing the loop. Eventually, normal BGP will fix the loop but packets to MIT will be stuck in the loop and suffer drops until then.

We observe that the routing loop could be avoided if AT&T and Joe could determine that Bob's withdrawal renders the old paths through each other unavailable. This is possible if Bob includes in its update to Joe and AT&T *Root Cause Information (RCI)* indicating that the link between Bob and MIT is no longer available, preventing Joe and AT&T from attempting to route on any paths that use this link. This leads to our second mechanism:

Mechanism 2 - Root Cause Information: *Include in each update message Root Cause Information indicating which other paths will not be available as a result of the same root cause event.*

We defer the details of implementing RCI until §7.3. The idea of including in each update its root cause, however, is not new. Prior work [23, 29] uses root cause information to reduce BGP convergence time and number of messages. R-BGP benefits from the reduced convergence time and reduced number of messages provided by RCI, but is novel in using RCI to prevent routing loops during convergence. Assuming the valley-free and prefer-customer policies, we prove that using RCI eliminates transient routing loops:

Lemma A.9. *Consider a network that is in a converged state at time t, when a link fails, and converges again at $t+\tau$. At no time between t and $t+\tau$ do the forwarding tables contain any loops.*

The intuition underlying the proof is that loops occur because at least one AS tries to use an out-of-date route. RCI allows an AS to locally purge out-of-date routes, preventing the creation of transient loops. The details of the proof are in the Appendix.

6.3 Ensuring a Usable Path Throughout Convergence

We have just shown that if an AS has a usable path it will be loop free, but we have not yet shown that an AS will continue to have such a usable path throughout convergence. Continuing with our example in Fig. 5a, when Bob withdraws his path, Joe uses RCI to determine that the old path offered by AT&T is invalid. Joe concludes that he has no available path and starts to drop packets, even though AT&T will eventually advertise him a new path through Sprint. How do we prevent ASes, like Joe, from transiently losing their path? To solve this issue, we take the following approach:

Mechanism 3a - Use Old Primary Paths: *When left without a usable primary path, the AS immediately upstream of a down link forwards along the failover path, and all other ASes continue to forward along their old primary and failover paths.*

Thus, even though the path through Bob has been withdrawn, Joe can temporarily use the withdrawn path to forward packets to the destination.

Using old paths in this way, however, raises another question: How long can Joe use this old primary path? Ideally, Joe would *use the old primary path until one of his neighbors announces a new path or he knows that he will not have a path after convergence*. The first part is simple; If a neighbor announces a path, Joe just moves to the new path because, as we have just proven, RCI guarantees that this new path will be loop-free and will not traverse the down link.

The second part has a catch though– in current BGP an AS cannot tell whether a neighbor will eventually announce a path to him. This decision must be handled carefully, not waiting long enough may cause premature packet-loss, but waiting too long can create a deadlock state leaving an AS indefinitely forwarding along an old path. To precisely determine whether an AS will have a path after convergence requires an individual AS to ascertain a global property of the network using only the local information available to it. Griffin et al. have proven that even with access to the entire link state of the Internet, and all policies of all ASes, it is an NP-complete problem just to determine if the network will converge, without even attempting to determine the state to which it will converge [15].

The Internet has an inherent structure to it, however, and we take advantage of this structure to allow an AS to determine when it will know an available path. In particular, most paths on the Internet are valley-free [12], that is all ASes on the path pursue economic interests by offering the path only if it goes to or from a customer. Focusing on this common case of valley-free paths, allows R-BGP to provide the global guarantee of continuous connectivity, while using only local information available to each AS. We use the following Mechanism to communicate the required information to allow an AS to locally determine the global property of whether it will eventually have an available path:

Mechanism 3b - Ensuring Convergence: *An AS stops forwarding internally originated traffic along withdrawn primary paths or failover paths when explicit withdrawals have been received from all neighbors. An AS delays sending a withdrawal to a neighbor until it is sure it will not offer this neighbor a valley-free path at convergence.*

To see how this works, suppose another neighbor John is using Joe to get to MIT as shown in Fig. 5b. After Bob withdraws his route from Joe, Joe knows no route to MIT, until AT&T announces a new path. Joe waits until AT&T withdraws its current path, which contains the down link, or replaces it by a new path before sending a withdrawal to John, thus ensuring continuous connectivity for John. In contrast, it sends a withdrawal to AT&T as soon as it hears the withdrawal from Bob, since it knows it will not have a valley-free path to offer AT&T at convergence. This prevents a deadlock where Joe and AT&T are waiting on each other to send withdrawals.

More generally, an AS knows it will not offer a valley-free path to a non-customer once it has heard withdrawals or advertisements of non-valley free paths from all customers. Additionally, it knows it will not offer a valley-free path to a customer once it has heard withdrawals or non-valley-free advertisements from all neighbors. To identify paths which are valley-free, advertisements include an additional bit indicating whether or not a path is valley-free.

Since valley-free paths are also loop-free, enforcing that delayed withdrawals only follow valley-free paths allows R-BGP to ensure continuous connectivity in the common case when ASes are following valley-free and prefer-customer policies, and still avoid deadlock regardless of policies. Formally, Mechanism 3b ensures:

Theorem A.10. *Regardless of policies, in a converged state, no AS is deadlocked waiting to send an update, and no AS is forwarding packets along a withdrawn or failover path.*

Lastly, note that to ensure continuous connectivity, ASes continue to forward traffic received from their neighbors.

6.4 R-BGP: Intuition and Guarantees

Together, the preceding three mechanisms ensure continuous connectivity between two ASes as long as the underlying graph is connected. To understand this, recognize that when a link fails, the ASes most affected are those using a path through the down link. If we call the AS upstream of the down link the *failover AS*, then Mechanism 1 ensures that the failover AS will have a failover path when the link first goes down, allowing it to initially protect all traffic sent by the affected ASes. If we refer to alternate paths that do not contain the down link as *safe paths*, then Mechanism 2 ensures no loops throughout convergence by allowing an AS that changes its path as a result of the down link to avoid unsafe paths. Finally, Mechanism 3 ensures that, throughout convergence, all affected ASes protect their traffic by forwarding it on their old path to the failover AS, until they, or some AS between them and the failover AS, learn a safe path to the destination. Thus, the combination of all three mechanisms allows us to prove:

Theorem A.11. *Consider a network that is in a converged state at time t, when a link goes down, and converges again at $t + \tau$. Assuming the valley-free, and prefer-customer policies, if an AS A knows a path to destination p at times t and $t + \tau$, then at any time between t and $t + \tau$ the forwarding tables contain a path from A to p.*

The proof of this theorem is in the Appendix. This last theorem is the culmination of R-BGP design. It proves that R-BGP achieves the goal stated at the beginning of this section. Specifically, R-BGP ensures that any two domains stay connected as long as the underlying AS-graph has a valley-free policy compliant path that goes between them.

7 Implementation & Protocol Details

Our discussion so far considers each AS to be a single entity, but ASes are composed of many routers. Here, we describe a router level implementation of R-BGP; specifically, how R-BGP forwards packets, works with route reflectors, and communicates RCI. Fig. 12 in the appendix has pseudo-code for R-BGP's update algorithm.

7.1 Packet Forwarding

Packet forwarding with R-BGP is similar to normal forwarding except that when packets arrive at a given router, they may be traveling along either the primary path or the failover path, and the router will need to forward the packet differently in each case. This differentiated forwarding requires: a) detecting whether a packet is on the failover path or the primary, b) storing the next hop for both the primary and failover paths, and c) forwarding the packet to the appropriate next hop.

(a) Different Virtual Connections for Primary and Failover

To allow the router to distinguish whether the packet is traveling along the primary or the failover path, R-BGP utilizes two "virtual" layer-two connections between each pair of BGP-speaking routers, one for primary path traffic, and one for failover traffic. The simplest way to implement virtual connections is through virtual interfaces and virtual

LANs (VLANs). When the two routers are not physically connected, we use MPLS or IP tunnels. Thus, traffic sent out the primary virtual interface on one router, will arrive on the primary virtual interface on the other router, and similarly for the failover virtual interfaces. Supporting additional VLANs and virtual interfaces like this comes with little to no overhead, and is already heavily used both for configuring peering relationships between ISPs at public peering points and for configuring VPN customers [3].

(b) Storing Primary and Failover Forwarding Information

Routers can easily support the storage of separate forwarding entries for the primary and failover path through separate forwarding tables. Current routers already support separate forwarding tables for separate virtual interfaces [1]. While this simple model doubles the required forwarding memory, modern routers can accommodate such need. Since they need to support many VPNs, next-generation backbone routers are designed to handle a few million routes. In contrast, the largest Internet routing tables currently have only 200-300 thousand external entries [16], and this is expected to grow to only 370,000 in the next 5 years [25]. Any additional dollar cost associated with the added memory should not be prohibitive either because the forwarding memory typically represents less than 10% of the overall dollar cost of a router line card [1].

We can optimize the forwarding memory overhead by combining the primary and failover tables entries into a single integrated forwarding table. Most high speed router architectures have a level of indirection between the destination IP look-up, and the forwarding entries which contain the next-hop information used to forward the packet. The memory in the forwarding table is typically dominated by the IP look-up portion, because this is usually stored as a tree in a relatively memory inefficient way in order to facilitate fast look-ups. Note that R-BGP does not increase the number of entries (prefixes) in the forwarding table, rather it stores two pieces of information for each prefix – the primary and the failover next hop information. Hence, both primary and failover forwarding entries can be merged into a single table, eliminating the overhead of storing a second tree. The look-up tree need only be extended to store at each leaf two indices into the table storing next-hop information, one for the primary next-hop information, and one for the failover next-hop information.

Furthermore, it is possible to completely eliminate any need for additional memory on the line cards of the router. To do so, one stores the failover table on a specialized dummy line card in the router with no physical interfaces of its own. All packets arriving on any failover virtual interface on the router would be sent to this dummy line card, which would perform the look-up in its copy of the failover table, and ensure the packet is forwarded accordingly. One line card per router should provide sufficient capacity as long as multiple links connected to a given AS do not fail at the same time. This is similar to line cards built to handle tunnel encapsulation and decapsulation at line speed [2, 4].

(c) Forwarding Process

The path followed by a packet utilizing a failover path has at most three segments: (s1) the packet travels along a prefix of the old primary path along which packets were traveling before the link went down, (s2) the packet reaches the router immediately connected to the down link, who forwards the packet along its failover path, and (s3) the packet reaches a router that knows a primary path not containing the down link, and the packet is forwarded along that primary path.

When a router in (s1) receives a packet, it will receive the packet along a primary virtual interface. Since the primary virtual interface is associated with the primary forwarding table, the packet arrival will trigger a look-up in this forwarding table. For routers in (s1), the result of such a look-up will be the primary virtual interface towards the next-hop router on the primary path.

This continues until the packet reaches (s2), i.e., until it reaches the router immediately upstream of the down link. Since packets arrive at this router along a primary virtual interface they again trigger a look-up in the primary forwarding table. This router uses Bidirectional Forwarding Detection [19] to quickly detect the link failure and it populates all entries in its primary forwarding table that use the down link with the associated failover path entries instead. Thus, on this router, a look-up in the primary forwarding table will result in a failover virtual interface.

Thus, the packet will continue to be forwarded along failover virtual interfaces, with all look-ups performed on failover forwarding tables, until it reaches (s3), i.e., until it reaches a router that knows a primary path that does not contain the down link. This router will have an entry in its failover forwarding table that contains a primary virtual interface. From this point on to the destination, the packet will be forwarded along primary virtual interfaces, with look-ups performed only in primary forwarding tables.

7.2 Advertising Failover Paths with Route Reflectors

Large ASes utilize route reflectors to reduce overhead. Rather than have each router in the AS connect to every one of the BGP border routers (full-mesh), the route reflectors act as a central point where all externally learnt routes are stored and propagated to other routers. This works identically with failover paths; a border router that learns a failover path advertises the route to the route reflector, which in turn advertises its best failover path to the other routers.

7.3 Communicating Root Cause Information

BGP update messages are triggered by interdomain link state changes, intradomain link state changes, or configuration changes, which we call the root cause of the update event. Here, we focus on interdomain link state changes in the context of a single destination, but the same applies to other causes and destinations.

Root Cause Information (RCI) is created and forwarded whenever a link changes state. Note that when a link changes state, at most one of the the two routers adjacent to the link, i.e., whichever router uses the link for the given destination, will generate an update to its primary path. This *root cause router* attaches its AS number, a router identifier and the new link state to all updates generated as a result of the link state change. This information is called *Root Cause Information (RCI)*. Routers that change their routes as a result of receiving such an update message, copy the RCI in the received update into the update messages they generate. This allows any router whose paths are affected by this link state change to learn the unique root cause that triggered the series of update messages and purge *other* routes that include the down link from its routing table.

Encoding the appropriate link state change information is non-trivial, however, since updates triggered by different root cause events may propagate at different speeds. For example, if a link flaps (i.e., a link goes down and then immediately comes back up), the link down update may arrive at a router after an update from another neighbor announcing the link up event, causing the router to mistakenly assume the link is still down.

To solve this problem, R-BGP uses a monotonically increasing sequence number per BGP router and includes in the RCI both the identifier and the sequence number of the *root-cause router*. Further, in a router's BGP RIB, the AS-Path information includes for each AS in the AS-PATH, the *egress router* for that AS, and that router's sequence number. Upon receiving an update with RCI, the router purges a route if any of the route's AS-PATH entries matches both the AS and the router identifier in the update's RCI, and has a lower associated sequence number than the update's RCI. When an interdomain link goes down, all ASes on the affected AS-paths will receive updates with the root-cause router's RCI and will mark these paths as withdrawn. This is sufficient to prevent interdomain loops (Lemma A.9).

8 Experimental Evaluation of R-BGP

We evaluate R-BGP using simulations over the best known estimate of the Internet AS-graph.

8.1 Obtaining Internet Graphs and AS Policies

The most important test of an interdomain routing protocol is how well it performs in a full scale Internet setting. Thus, we evaluate R-BGP on a 24,142 node AS-level graph of the Internet generated from BGP updates at Routeviews [6] vantage points. Additionally, we use the best known policy inference algorithms [10] to annotate inter-AS links as either customer-provider, provider-customer, or peer-peer (see §2).

The algorithms used to produce the policies have some limitations. First, the algorithms sometimes produce provider-customer loops. It is well known that such loops do not exist in the Internet, and could lead to persistent BGP loops [12]. Thus, we eliminate them by finding the AS in the loop with the fewest neighbors, and removing the edge between this AS and the next AS in the loop (its customer). Second, when one ISP spans multiple AS numbers, the algorithm in [10] assign these ASes a *sibling* relationship. We treat sibling relationships as peer-peer, since treating them as anything else also leads to provider-customer loops.

8.2 Simulator

BGP implementations in existing simulators such as SSFNET [5] and ns-2 [26] use so much memory that they cannot, even on our 8GB server, scale to the 24000-node AS graph of the Internet. To simulate the full Internet, we had to write our own BGP-specific simulator. Our simulator is optimized for performance, yet implements all the important characteristics of BGP convergence. In particular, it implements sending and receiving of update and withdrawal messages, detailed message timing including the MRAI timer, and the full BGP decision process including relationship-based route preferences.

8.3 Compared Protocols

We compare current BGP with three variants of R-BGP that differ only in the choice of the failover path.

- *Most-Disjoint Failover Path:* When ASes advertise the path most disjoint from their primary as a failover path, R-BGP guarantees that no unnecessary drops occur due to transient inter-AS disconnectivity. Though the most-disjoint path may not be policy compliant, ASes have an incentive to advertise it for failover. First, the cost is little–a failover path is only used for a short time until BGP converges. Second, as explained in §5, an AS that does not announce the most disjoint failover path risks having its own packets dropped upon a route change since a downstream AS may have no usable path.
- *Most Disjoint Policy Compliant Failover Path:* We also evaluate the performance of R-BGP when an AS advertises the most disjoint policy-compliant failover path. In particular, we would like to quantify how useful R-BGP would be if we limit failover paths to be policy compliant. If the likelihood of transient disconnectivity is very low, then it is probably sufficient to stick to policy compliant paths even for failover.
- *Second Most Preferred Failover Path:* It is also natural to ask what would happen if each AS advertises its second best path as its failover path, again limiting to only policy compliant paths. After all, when the primary path fails, an AS uses the second best path.

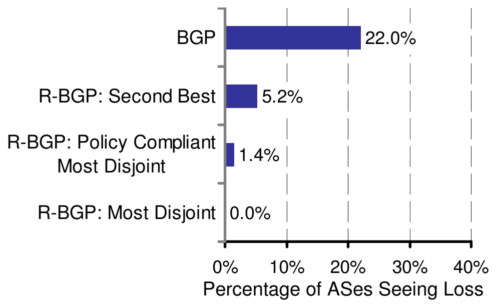


Figure 6: Singe Edge-Link Failure: Percentage of AS sources that will have a policy compliant path after convergence, but see transient disconnectivity to a dual-homed edge domain when one of its links fails.

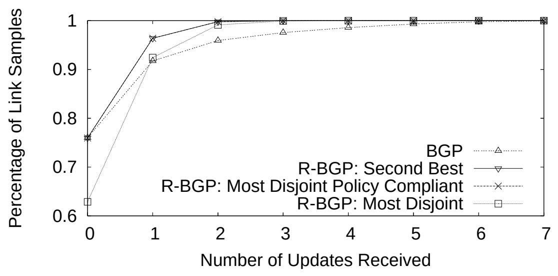


Figure 7: Number of Updates Per Link: CDF of the number of updates sent on each interdomain link. The measurements are from 18,400 simulations, where one of the two links of a dual-homed destination domain is taken down.

8.4 Experimental Setup

To ensure R-BGP works in all cases, we evaluate its performance for both edge link and core link failures.

(1) Dual-Homed Edge Domains: It is common in today's Internet for edge domains to be multi-homed. Multi-homing is sought after to improve resilience to access link failure. But how effective is such a backup approach in preventing transient disconnectivity? To answer this we look at the effect of taking down a link connected to a dual homed edge domain. For each of the 9200 dual-homed edge domains in our AS graph, we run a simulation, in which we take down one of the domain's two access links, and ask how many source ASes will experience transient disconnectivity to the domain. Specifically, we compare the performance of BGP and the R-BGP variants using the following metric: Among the sources that will be connected to the dual homed edge domain *after* BGP converges, what fraction will see disconnectivity during convergence? We also use this scenario to determine the performance of the various protocols when multiple links are taken down, and when a link comes back up at the same time that another goes down. Further, we quantify the relative overhead of the various versions of R-BGP by measuring the number of routing messages exchanged and the time to converge.

(2) Core Link Down: While the above scenario looks at access links, many more ASes can be affected when core links fail and so it's important to confirm that R-BGP avoids transient disconnectivity in these scenarios as well. We define a core link as a connection between two non-stub domains. In this scenario we compare the various protocols on the following metric: Among the AS pairs that were using the down core link, and will be connected *after* BGP converges, what fraction will see disconnectivity during convergence? For each of the 200 links that we tested, we ran 24142 simulations, one for each possible destination AS, and averaged the results across links.

9 Empirical Results

Our empirical results confirm that in both the dual-homed edge domain scenario, and the down core link scenario, R-BGP prevents disconnectivity when a single link goes down. Additionally, even when multiple links go down simultaneously, R-BGP avoids almost 80% of the disconnectivity seen with BGP. Further, R-BGP achieves this performance with message overheads comparable to BGP and surprisingly improves convergence times.

9.1 Dual-Homed Domains Resilience to Link Failures

We first consider the dual-homed edge scenario. In particular, we explore the following question: How many source ASes are transiently disconnected when a dual-homed AS loses one of its access links? Fig. 6 reports the percentage of ASes that temporarily lose connectivity, averaged over each link for all 9200 dual-homed domains. The figure shows that, in BGP, 22% of the ASes experience transient disconnectivity when one of the links to a dual-homed destination domain fails. R-BGP using most disjoint failover paths reduces this number to zero, confirming the analytical guarantees.

The figure also shows that all variants of R-BGP significantly increase resilience to transient disconnectivity. R-BGP performs adequately when working within the confines of current policies; using policy compliant most disjoint paths for failover allows disconnectivity in only 1.4% of the cases. This means that even if ASes are not willing to temporarily provide transit for their non-customers, R-BGP still avoids almost all disconnectivity. Announcing the second best path as a failover path, though does not perform as well, allowing 5.2% of the domains to be temporarily disconnected. This is because often there is much overlap in the best and the second-best paths, reducing the number of link failures that the failover paths cover. Still, even an R-BGP variant that uses the second best path for failover is significantly better than current BGP.

9.2 Number of Updates and Convergence Time

Since R-BGP needs to maintain and update failover paths in addition to primary paths, one may be concerned that R-BGP may delay convergence or exchange a large number of update messages. Our results show that the number of update messages in R-BGP is comparable to BGP, and surprisingly R-BGP converges faster than BGP.

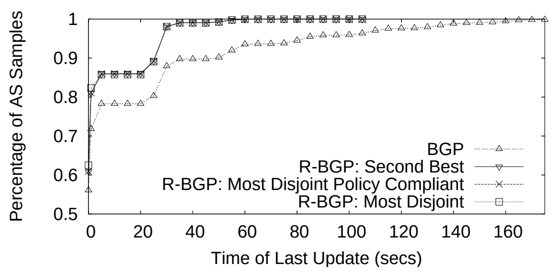


Figure 8: Convergence Time: CDF of the convergence time taken over AS samples. Convergence time is measured as the interval between the failure and the last time an ASes primary path forwarding table is updated. The measurements are from 18,400 simulations, where one of the two links of a dual-homed destination domain is taken down.

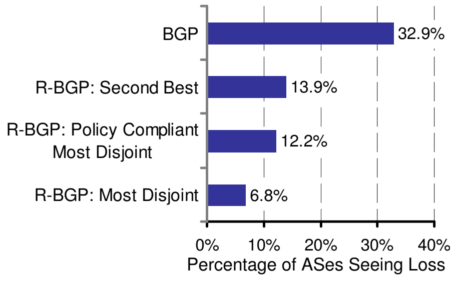


Figure 9: Simultaneous Link Failures: Percentage of ASes which have a policy compliant path after convergence, but see transient disconnectivity to a dual-homed edge destination, in the worst case scenario where one of its access links is taken down at the same time as a link on the failover path of the AS immediately upstream of the down link.

Again, considering the dual-homed edge domain scenario, Fig. 7 plots the cumulative distribution of the number of updates exchanged across each link in the graph, when routes converge after one of the two links of a dual-homed destination is brought down. The figure shows that 92% of interdomain links see at most one update during convergence for all protocols. R-BGP with most disjoint failover paths sends fewer messages than the other variants. Further, all variants of R-BGP send a number of messages comparable to BGP.

It might look surprising that R-BGP sometimes sends fewer updates than current BGP. This is due to the Root Cause Information (RCI) mechanism described in §6.2. In particular, when a link fails, current BGP may move to an alternate route that contains the same failed link and advertise this new route to neighboring ASes. RCI provides an AS with enough information to locally purge all routes that traverse the failed link, thus preventing such useless updates and significantly reducing path exploration. One may wonder whether RCI significantly decreases the number of messages and the failover advertisements eat most of this decrease, making the overall number of messages on par with BGP. This however is not the case. The number of messages with just RCI is only a bit smaller, but we omit these results here to enhance readability.

Fig 8 plots the cumulative distribution of the convergence time for each protocol when one link of a dual-homed domain is brought down. It shows that, on average, R-BGP tends to converge faster than BGP. Again this is because RCI eliminates unproductive path exploration during convergence. In our simulations, all R-BGP variants never took longer than 106s to converge, whereas BGP needed as long as 323s in certain cases. Again, RCI alone only does slightly better than R-BGP–it always converges within 96s, indicating that the overhead of the failover paths adds little to the convergence time.

9.3 Multiple Simultaneous Events

Our focus so far, including our analytical guarantees, has been on the case when only a single link goes down at a time. It is less likely for multiple link-down or link-up events to happen simultaneously especially at the interdomain level. Still, we show empirically that, even in this case, R-BGP can significantly reduce the chances of transient disconnectivity. To simplify interpretation, we again show this in the dual homed edge domain scenario.

9.3.1 *Failure of Both Primary and Failover Paths*

Rather than selecting the two failed links randomly, we simulate one of the worst cases for R-BGP, namely simultaneous link failures on both the primary and failover paths. As before, we pick the first link from among the two access links of a dual-homed edge destination. We pick the second link randomly from among the links on the failover path that would have been used to compensate for the failure of the first link. We perform a total of 9200 $\times$ 2 $\times$ 4 simulations, i.e., for each of the links of the dual-homed ASes, we randomly fail four different links on the failover path, one at a time.

Fig. 9 shows that R-BGP reduces the number of disconnected sources from 32.9% to 6.8%, avoiding 80% of the disconnectivity seen with BGP. The intuition is that even though the failover path of the first failed link is broken, the failover path of the second failed link may still be operational, thus avoiding disconnectivity.

9.3.2 *Changing Failover Path During Failure*

Can ongoing convergence on the failover path hinder the ability to ensure connectivity when the primary path fails? Our simulations show that R-BGP avoids transient disconnectivity and that no adverse interaction happens.

Specifically, we simulate the following worst case scenario for R-BGP. We pick one of the two access links of a dual-homed AS and bring down its preferred failover path by bringing down a link on that path. This triggers a change in the failover path. After the network has converged, we fail the access link of the dual-homed AS. At the same time, we bring up the previously failed link of most preferred failover path. This triggers a change in the failover path while it is in use.

Fig. 10 shows that despite ongoing convergence on the failover path, no packets are dropped when the primary

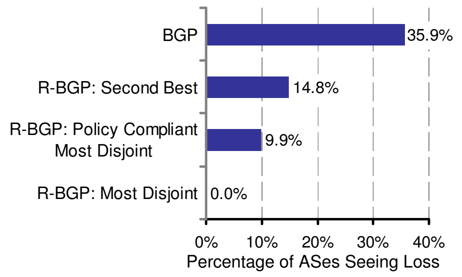


Figure 10: One Up One Down: Percentage of sources which have a policy compliant path with both links down, but see transient disconnectivity to a dual-homed edge destination, when one of it's links is taken down at the same time as bringing up a link on the default failover path of the AS immediately upstream of the first down link.

path goes down. In contrast, BGP causes over 35% of the ASes to lose connectivity to the dual-homed AS. Surprisingly, this number is significantly larger than the percentage of AS disconnectivity caused by BGP when a single link goes down. The fact that two events occurred together increases BGP transient disconnectivity despite that one of the events is a link up. This surprising fact is consistent with prior results that show that BGP might experience disconnectivity even when a single link comes up [34].

9.4 Resilience to Core Link Failures

Finally, we want to ensure that R-BGP performs well during core link failures in addition to access link failures. Thus, using the methodology described in §8.4, we test disconnectivity when a randomly chosen core link in the network fails. Figure 11 reveals that R-BGP using most disjoint failover paths eliminates packet loss for core link failures. Further, we see that core link failures cause proportionally less loss than edge links for all four protocols, because the highly connected nature of the core ensures that alternative paths are available to BGP. However note that even though a smaller fraction of ASes may see loss when a core link fails, failures on such links cause significantly more total packet loss as they carry much more traffic.

Additionally, note that R-BGP using most disjoint policy compliant paths sees the largest relative improvement, as we look at core link failures instead of edge link failures. To see why this is the case, note that if an AS remains disconnected after convergence, it cannot know a policy compliant failover path before the link fails but, if most disjoint paths, regardless of policy, are used the AS is guaranteed to have a failover path (Lemma A.4). Such persistently disconnected ASes and any AS that uses a path through such an AS will see transient disconnectivity when R-BGP restricts to most disjoint policy compliant failover paths. In our experiments, we see that persistently disconnected ASes tend to be long haul backbones like Abilene and GEANT that are primarily connected to stub ASes and don't have providers of their own. Such specialized long haul backbones are used more commonly by access net-

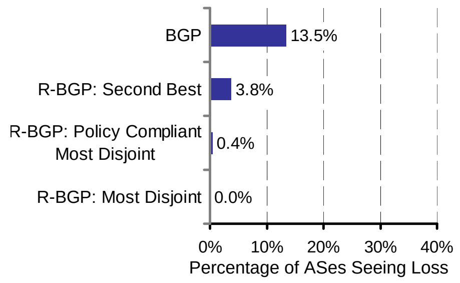


Figure 11: Single Core-Link Failure: When a random core-link is taken down, the fraction of source destination pairs using that link which see transient disconnectivity.

works that do not have a long haul backbone of their own in order to connect between regional offices. Since paths through a core link are less likely to involve this type of long haul backbone AS, a correspondingly smaller fraction of paths using a core link are affected by whether the most disjoint path is restricted to be policy compliant.

10 CONCLUSION

This paper shows that transient disconnectivity during BGP convergence is unwarranted and can be easily avoided. Our approach, called R-BGP, uses a small number of precomputed failover paths to protect data forwarding from the pathological effects of BGP dynamics. R-BGP performs a few modifications to current BGP that are easy to implement and deploy in today's routers. Simulations on the AS-level graph of the current Internet show that R-BGP reduces the number of domains that see transient disconnectivity resulting from a link failure from 22% for edge links and 14% for core links down to zero in both cases. Further, R-BGP achieves this loss reduction with message overheads close to current BGP, and, surprisingly, reduces convergence times.

11 ACKNOWLEDGMENTS

We thank both Bruce Davie and Garry Epps of Cisco Systems, and John Scudder of Juniper Networks for answering our repeated questions on router implementation details. We thank Hariharan Rahul for extensive feedback on various versions of the paper. We thank our Shepherd Alex Snoeren for constructive comments on various versions of the paper. We also acknowledge the support of Cisco Systems and the National Science Foundation under NSF Career Award CNS-0448287. The opinions and findings in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation or Cisco Systems.

REFERENCES

[1] Private Discussions with Bruce Davie and Garry Epps of Cisco.

[2] Cisco GRE Tunnel Server Card. `http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/120newf%t/120limit/120s/120s21/gre.htm`.

[3] Equinix Peering FAQ. `https://ecc.equinix.com/peering/faqs.htm`.

[4] Juniper Networks Tunnel Services PIC. `http://www.juniper.net/products/modules/100092.pdf`.

[5] SSFNet. `www.ssfnet.org`.

[6] Univ. of Oregon Route Views. www.routeviews.org.
[7] S. Agarwal, C.-N. Chuah, S. Bhattacharya, and C. Diot. Impact of BGP Dynamics on Router CPU Utilization. In *PAM*, 2004.
[8] O. Bonaventure, C. Filsfils, and P. Francois. Achieving sub-50ms Recovery upon BGP Peering Link Failures. In *Co-Next*, 2005.
[9] A. Bremler-Barr, Y. Afek, and S. Schwarz. Improved BGP Convergence via Ghost Flushing. In *INFOCOM*, 2003.
[10] X. Dimitropoulos, D. Krioukov, M. Fomendov, B. Huffaker, Y. Hyun, and kc claffy. As relationships: Inference and validation. *CCR*, 2007.
[11] P. Francois, C. Filsfils, J. Evans, and O. Bonaventure. Achieving Sub-Second IGP Convergence in Large IP Networks. *CCR*, 2005.
[12] L. Gao and J. Rexford. Stable Internet Routing Without Global Coordination. In *SIGMETRICS*, June 2000.
[13] L. Gao and F. Wang. The extent of AS Path Inflation by Routing Policies. In *Glob. Internet Symposium*, 2002.
[14] T. Griffin and B. Premore. An Experimental Analysis of BGP Convergence Time. In *ICNP*, 2001.
[15] T. G. Griffin and G. T. Wilfong. An analysis of BGP convergence properties. In *Proceedings of SIGCOMM*, pages 277–288, Cambridge, MA, August 1999.
[16] G. Huston. Growth of the bgp table - 1994 to present. http://bgp.potaroo.net,2006.
[17] International Telecommunication Union. Connection Integrity Objective for International Telephone Service. E.855., 1988.
[18] L. Jiazeng et al. An Approach to Accelerate Convergence for Path Vector Protocol. In *Globecom*, 2002.
[19] D. Katz and D. Ward. Internet Draft. Bidirectional Forwarding Detection. draft-ietf-bfd-base-05.txt.
[20] N. Kushman, S. Kandula, and D. Katabi. Can you hear me now?! it must be bgp. In *CCR*, 2007.
[21] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. Delayed Internet Routing Convergence. In *SIGCOMM*, 2000.
[22] S. Lee et al. Proactive vs. Reactive Approaches to Failure Resilient Routing. In *INFOCOM*, 2004.
[23] J. Lou, J. Xie, R. Hao, and X. Li. An Approach To Accelerate Convergence for Path Vector Protocol. In *Globecom*, 2002.
[24] Z. Mao, R. Govindan, G. Varghese, and R. Katz. Route-flap Damping exacerbates Internet Routing Convergence. In *SIGCOMM*, 2002.
[25] D. Meyer, L. Zhang, and K. Fall. Internet Draft. Report from the IAB Workshop on Routing and Addressing, Dec 2006.
[26] The Network Simulator. www.isi.edu/nsnam/ns.
[27] E. Osborne and A. Simha. *Traffic Engineering with MPLS*. Cisco Press, 2002.
[28] D. Pei et al. Improving BGP Convergence Through Consistency Assertions. In *INFOCOM*, 2002.
[29] D. Pei et al. BGP-RCN: Improving BGP Convergence through Root Cause Notification. *Computer Networks Journal*, June 2005.
[30] J. Rexford, J. Wang, Z. Xiao, and Y. Zhang. BGP Routing Stability of Popular Destinations. In *IMW*, 2002.
[31] E. Rosen, A. Viswanathan, and R. Callon. Multi-protocol Label Switching Architecture. RFC 3031.
[32] M. Shand and S. Bryant. IP Fast Reroute Framework, Oct 2005.
[33] L. Subramanian, M. Caesar, C. T. Ee, M. Handley, M. Mao, S. Shenker, and I. Stoica. HLP: A Next-generation Interdomain Routing Protocol. In *SIGCOMM*, 2005.
[34] F. Wang, Z. M. Mao, J. W. L. Gao, and R. Bush. A Measurement Study on the Impact of Routing Events on End-to-End Internet Path Performance. In *SIGCOMM*, 2006.
[35] W. Xu and J. Rexford. Multi-path Interdomain Routing. In *SIGCOMM*, 2006.

APPENDIX

A FAILOVER PROOFS

An important step in our design is to formally prove that the proposed protocols achieve their goals.

```
When an update is received:
a) Store AS-Path and Egress Router Sequence Number List in RIB-In
b) Mark withdrawn and out-of-date all RIB-In entries whose Egress
     Router Sequence Number List contains an entry with same AS
     and Egress Router as RCI, but with a lower sequence number
c) Compute new primary using the BGP decision algorithm over non-
     failover paths
d) Compute, over all paths, most disjoint path from the primary path,
     and set as the new failover path
e) If primary path changes and is non-empty, increment sequence
     number, send new primary path to all non-next-hop neighbors
     for whom it is policy compliant, and update primary forwarding
     table with new primary
f) If primary or failover changed and new primary path is non-empty
     send an update containing new failover, regardless of policy, to
     new primary next-hop neighbor, and update failover forwarding
     table with new failover
g) When no customer paths are marked out-of-date, if the primary
     path is not empty send withdrawals to all non-customers for
     whom the primary path is not policy compliant; otherwise the
     primary path is empty, so send withdrawals to all non-customers.
h) If new primary is empty path, no path is marked out-of-date path
     from any neighbor, no peer is offering a loopy path, and no
     provider is offering a loopy or backup path, then update for-
     warding table to stop forwarding locally originated packets, and
     send withdrawals to all neighbors to whom withdrawals have not
     already been sent, but continue forwarding along the old path
     packets received from neighboring ASes
```

Figure 12: Pseudo-code for the algorithm

We model the network as a directed graph in which each node represents a single-router autonomous system (AS). We analyze the scenario in which BGP is in a converged state at time t, then a link e goes down, and finally BGP converges again at time $t + \tau$. We assume that between times t and $t + \tau$ no other events occur, i.e., no other links come up or go down. We also assume that during this time no customer-provider-peer relationships between ASes change, and that no AS changes its policy regarding which routes are preferred and to whom those routes are advertised. Without loss of generality, we analyze the available paths to one prefix p during the transition.

The union of the paths specified in the router forwarding tables at time t from all origins to a particular destination p form a directed "forwarding tree" rooted at p with all edges pointing toward the root. (The paths form a tree because BGP ensures that there are no cycles in the forwarding tables at convergence.) We visualize the forwarding tree with the root p at the bottom, so that all paths flow down to p. We use Q to denote the AS that is "upstream" of the failing link e in the tree, i.e., the link connects Q to its downstream parent in the forwarding tree.

In Theorems A.10 and A.11, we prove the two important properties of R-BGP. First, if there is a path from an AS A to prefix p at time t and there is a path from A to p at time $t + \tau$, then in any snapshot of the forwarding tables taken between times t and $t + \tau$ there is a path from A to p. Second, R-BGP does this without ever deadlocking, i.e., it never reaches a state where a set of ASes are all waiting on each other before sending an update or withdrawal.

At various points in the analysis we make assumptions about routing policy. In order to state these assumptions, we must introduce several definitions. First, we model every pair of directly connected ASes as joined by two directed edges. Further, every directed edge in the network has a label indicating the relationship between its endpoints, either customer-to-provider, provider-to-customer, or peer-to-peer. Oppositely directed edges have reversed labels. We also assume that there are no directed cycles consisting entirely of customer-to-provider edges and no directed cycles consisting entirely of provider-to-customer edges, i.e., no AS is an indirect customer of itself.

We use the term path to refer to a sequence of directed edges (or list of ASes) advertised from one AS to another as a routing path, and the term forwarding path to denote a sequence of directed edges traversed by data traffic. A path or forwarding path in the network is *valley-free* if an edge labeled either provider-to-customer or peer-to-peer can only be followed by edges labeled provider-to-customer. (Thus in a valley-free forwarding path, no AS on the path forwards packets from a non-customer to a non-customer.) As a consequence, a valley free path consists of three ordered parts, although any of them may be empty. The path starts with a sequence of customer-to-provider edges, perhaps contains a single peer-to-peer edge, and then ends with a sequence of provider-to-customer edges. We say that an AS A observes a *valley-free policy* if it never advertises a path from a non-customer to a non-customer.

Lemma A.1. *A valley-free path cannot contain a loop.*

Proof. A simple case analysis rules out loops. First, if the loop contains any provider-to-customer link, then by the valley-free property of the path, the loop must consist entirely of provider-to-customer links. But, this violates the above assumption that there can be no loop in the network consisting solely of customer-to-provider links or solely of provider-to-customer links. Second, if the loop contains a peer-to-peer link, then all of the following links must be provider-to-customer links by the valley-free property of the path – specifically there is at least one provider-customer link in the loop. But, as argued in the first case, no valley-free path with a provider-customer link can contain a loop. Finally, the loop cannot consist entirely of customer-to-provider links, by our assumption. □

We also observe that the reverse of any valley-free path is also valley-free and thus loop free.

We say that an AS A observes a *prefer-customer policy*, if A always prefers a path that is advertised by a customer of A over a path that is advertised to A by a provider or peer.

The last definition is more subtle. We say that an AS A follows a *widest-advertisement* policy if the following holds. For a prefix p, if A's policy allows it to advertise a primary path to a neighbor B that it learned from another neighbor C, then whenever A knows of any path through C, it must advertise some path to B. Note that this latter path advertised to B need not have been learned from C. Recognize that this really limits an ASes policies in only two ways: (1) if C changes the path it's advertising to A, this cannot cause A to stop advertising it to B, and (2) if A moves to a path that it prefers more than the path through C, then it must also advertise this more preferred path to B.

We note that all three policies are consistent with each other. In particular the prefer customer policy is consistent with the other two policies because it limits an ASes preference policy which determines the path it will choose as its best path, while the valley-free and widest-advertisement policy limit an ASes advertisement policy, which determines whether the AS will advertise its chosen best path to a given neighbor. The valley-free and widest-advertisement polices are also consistent with each other. To see this, recognize that if AS B from the above description is a customer, then A can advertise any path to it and still maintain a valley-free policy. Additionally, if B is not a customer, then C must be a customer path if A is following a valley-free policy. However, if C is a customer and A moves to a path through some other neighbor, D, while the path through C is still available, then if it is following the prefer-customer policy, D must also be a customer. And since the valley-free policy allows customer paths to be advertised to all neighbors, the widest-advertisement policy is consistent with the valley-free policy as long as the prefer-customer policy is also in place.

We note two important issues. First, all of our assumptions comply with the commercial incentives of ASes and most ASes follow these policies for most of their paths in today's Internet [12]. Second, we only require the policy assumptions to hold for the set of paths involved in the convergence. Specifically, if policies inconsistent with these assumptions are either not exercised, or utilized only by ASes or prefixes not involved in the convergence, then the proofs will still hold. With our terminology settled, we now prove several lemmas on our way to the theorems.

Lemma A.2. *If at time t the primary path from an AS A to p passes through e, but A knows of a primary or failover path that does not use e, then A's failover path to p does not use e.*

Proof. The proof follows from the fact that BGP establishes destination based routes. Thus, any two paths to p that share e must share a common suffix beginning with Q (the AS upstream of e). On the other hand, if a path does not contain e, then it cannot share this entire suffix. Hence this path must be more disjoint from A's primary path than any path that contains e. □

Lemma A.3. *If at time t there is any AS A whose primary path to destination p passes through link e, but who knows of a primary or failover path that does not use link e, then, at time t, Q, the AS upstream of e, knows of a failover path that does not use e.*

Proof. As A must be upstream of Q, let $U_0, \dots, U_n$ denote the prefix of A's primary path to p, where $U_0 = A$, and $U_n = Q$. We prove by induction that for $0 \leq i < n$, U_i's failover path to p does not contain e, and hence U_i advertises a failover path that does not contain e to its successor, U_{i+1}, on the primary path. For the base case, by assumption and by Lemma A.2, U_0's failover path does not contain e. This failover path is advertised to its successor U_1 on U_0's primary path to p. For the inductive step, each U_i, $i > 1$, learns of a failover path that does not contain e from its predecessor U_{i-1}. Hence, U_i knows of at least one path that does not contain e, and by Lemma A.2, U_i's failover path, which U_i advertises to U_{i+1}, does not contain e. Finally, $Q = U_n$ learns of a failover path from its predecessor U_{n-1}. □

The previous lemma shows that if there exists an AS A upstream of Q that knows a failover path at t, then Q will know a failover path at t. The following lemma assumes the widest advertisement policy and shows that if there exists an AS A upstream of Q at t who will know some path at $t + \tau$, then Q will know a failover path at t.

Lemma A.4. *If any AS using a down link will have a path to the destination after convergence, then R-BGP guarantees that an AS that is using the down link and adjacent to it knows a failover path when the link fails.*

Proof. To prove this, we first inductively prove that at time t there must exist some AS V whose primary path goes through e, but who knows an alternate path that does not go through e. We then combine this with lemma A.3 to prove that at time t, Q must know a failover path that does not contain e.

First let us establish some notation. Suppose that some AS A uses a path through e to destination p at time t and knows of a path $P_0, P_1, \dots, P_k$ to p at time $t + \tau$, where $P_0 = A$ and P_k hosts the prefix p. Note that $P_0, P_1, \dots, P_k$ does not contain e, because e is down at $t + \tau$. Let i denote the largest index such that at time t the primary path from P_i to p passes through e. (Such an index must exist since the path from $A = P_0$ goes through e at time t.) Now, proving that P_i will know some primary or failover path that does not go through e at time t will complete the proof as P_i would satisfy the above constraints for V.

The key to the proof is to show that since each AS in the sequence $P_{i+1}, P_{i+2}, \dots, P_k$ advertises a path to its predecessor at time $t+\tau$, it must also do so at time t. The proof is by induction on the path, starting at P_k and moving towards P_{i+1}. For the base case, we note that since P_k hosts p, advertises a path to P_{k-1} at time $t+\tau$ and we assume that ASes do not change their policies between t and $t + \tau$, it must advertise a path for p to P_{k-1} at time t. Since P_k hosts p, clearly its path does not go through e. For the inductive case, P_j, where $i+1 \leq j < k$ receives a path from P_{j+1} that does not use e at t, and since it advertises to P_{j-1} the path received from P_{j+1} at time $t+\tau$, P_j must also advertise one at time t by the widest advertisement policy. By choice of i, P_j's primary path to p at time t does not go through e, so the path P_j advertises to P_{j-1} does not use e.

Now we apply Lemma A.3. Note that by the definition of i, P_{i+1} does not use a path through e at time t. Since, at time t, P_i uses a path through e and knows of a path from P_{i+1} that does not, by Lemma A.3, at time t AS Q knows of a failover path that does not use e. □

Lemma A.5. *Between time t and $t + \tau$, no AS besides Q changes its primary or failover path to p to one that uses e, and no AS besides Q changes either its primary or failover path at all until it knows a primary path that does not contain e. Furthermore, before Q learns a path to p that does not contain e, its only change is to set its primary path to be equal to its failover path.*

Proof. At time t BGP is in a converged state when link e goes down, implying that at time t there are no outstanding updates in the network for prefix p. The first updates issued by Q carry RCI indicating that link e has failed. All updates generated in response to these initial updates also carry the RCI. Between times t and $t+\tau$ there are no other events, so there are no sources of other updates. Thus all updates for destination p in the network between t and $t+\tau$ are indirectly generated by failure of e, and contain RCI indicating e has failed. Hence, Mechanism 2 ensures that no change to a primary or failover path made in response to an update can use e, and that no AS will update either its primary or failover path until it learns a primary path that does not contain e. The only exception to this is AS Q who moves to forwarding all packets on its failover path from the time it learns that e is down until it learns an alternate primary path not containing e. □

The next two lemmas build up together to prove in Lemma A.8 that once an AS learns a valley free path not containing link e, it will continue to know such a path, and all packets it forwards will follow such a path. We then use this to help prove loopfreeness in Lemma A.9 and path existence in Theorem A.11.

Lemma A.6. *Assuming the valley free, prefer-customer, and widest-advertisement policies, if at time t', such that $t \leq t' \leq t + \tau$, an AS P is offered a valley-free primary path that does not contain e from a customer C, then from t' until time $t + \tau$, C must continue to offer a valley-free primary path to P that does not contain e. Furthermore, any packets that P forwards through C must follow a valley free path that does not contain e.*

Proof. Suppose that at time t' AS P is offered a valley-free path that doesn't contain e from a customer C. First observe that any valley-free path offered from a customer to a provider consists of only provider-to-customer links. Hence the entire path from P to the AS that hosts p consists of provider-to-customer links and does not contain e.

We first prove that C must continue to offer a valley free primary path that does not contain e. Our proof is by induction along the reverse path from p to P. We show that for all time steps between t' and $t+\tau$, each AS on the path offers a valley-free customer path to its provider. As the base case, the AS that hosts p will never withdraw the path that it offers to its provider. To show the inductive step, observe that an AS C on the reverse path can only stop advertising a customer path (that does not use e) to its provider P for one of four reasons (which we will now rule out): (0) C moves to a path that contains e, (1) C loses its path and knows of no path at all, (2) C moves to a path that contains the provider P, or (3) C moves to a new best-path that it does not wish to offer to the neighbor. By Lemma A.5, C cannot move to a path that contains e, preventing (0). By induction, C will continue to know a customer path, preventing (1). Also, since C uses a prefer-customer policy, it cannot prefer a path through P to the path it knows through a customer, preventing (2). Finally, the widest-advertisement policy requires C to continue advertising a path to P if it knows of a path from the neighbor whose path it was advertising to P at time t'. By induction C continues to know a path from this customer, preventing (3).

We now prove that any packets P forwards through C must follow a valley free path that does not contain e. Let the path advertised by C to P at time t' be $P_0, P_1, \ldots P_k$ with $P_0 = P, P_1 = C$ and P_k hosting the prefix p. If P were to forward through C and none of the ASes $P_i, 0 \leq i \leq k$ were to change their primary paths before $t+\tau$, P's packets will trivially follow a valley-free path. Suppose, on the other hand, that P_j is the first AS starting from P that changes its primary path. By the argument above, we have shown that P_{j+1} will continue to offer P_j a valley-free customer path until $t+\tau$, so by the prefer customer policy, P_j could only have changed its primary path to that offered by another customer! Consequently, P_j will forward P's packets along a valley-free customer path. □

Lemma A.7. *Assuming the valley free, prefer-customer, and widest-advertisement policies, if at time t', such that $t \leq t' \leq t+\tau$, an AS A is offered a valley-free primary path that does not contain e from a provider or peer, P, then at each time step from t' to $t+\tau$, either P offers to A a valley-free primary path that does not contain e, or A knows of a customer path that does not contain e. Furthermore (in either case), any packets that A forwards through P must follow a valley free path that does not contain e.*

Proof. This proof follows the along the same general lines as that of Lemma A.6.

Again, we start by proving that P will continue to offer a valley free path not containing e, or A will know a customer path not containing e. Suppose that P is a peer of A. Then a valley-free path from P can only be a path that is offered to P from one of P's customers, C. By Lemma: A.6, P will continue to know a customer path from C until step $t+\tau$. By the prefer customer policy, P will not move to a path through A and thus by the widest advertisement policy P must continue to offer a path to A. By Lemma A.5, the path from P cannot use e.

Now suppose instead that P is a provider to A. In the most interesting case, P stops offering a path to A because it decides to use a path through A. This can happen only if A switches to a customer path and advertises it to P. But then by Lemma A.6, A will know of a valley-free customer path (that does not use e) until step $t+\tau$.

If at time t' P offers A a path through a customer C of P, then by Lemma A.6 C will continue to offer a path to P until step $t+\tau$. By the widest-advertisement and prefer customer policies, then, P will continue to offer a path through one of its customers to A until step $t+\tau$.

If at time t' P offers a valley free path to A from a peer R of P, then by the same analysis as that in which P is a peer of A, R will continue to know a customer path by Lemma A.6, and by prefer customer and widest advertisement will continue to advertise such a path to P until step $t+\tau$. By the widest advertisement policy, P must therefore continue to offer a path to A until step $t+\tau$.

Finally, suppose that at time t' P offers a path to A that it learns from a provider. In this case, we follow the path up from A along customer-provider links until reaching an AS G that passes a path down to its customer that it has learned from either a customer or a peer (rather than a provider). As we have just shown, G will continue to receive a path from its customer or peer. Now we prove inductively, moving down the path from G to P, that each AS continues to offer a path (that does not use e) to its customer.

To complete the proof, we can show, by induction starting at A, that any packets A forwards through P must follow a valley free path to the destination that does not contain e. The inductive proof is similar to that for Lemma A.6, by showing that any AS through whom a packet from A is forwarded will forward only along valley free paths. □

Lemma A.8. *Assuming the valley free, prefer-customer, and widest-advertisement policies, if at time t', such that $t \leq t' \leq t+\tau$, an AS knows a valley-free primary path that does not contain the down link e, then it will continue to know at least one valley-free primary path not containing e from time t' until time $t+\tau$, and any packets it forwards will follow a valley free path that does not contain e.*

Proof. This follows clearly from the previous two lemmas.

□

The following lemma assumes the widest advertisement policy and proves that no loops will form during convergence.

Lemma A.9. *Consider a network that is in a converged state at time t, when a link fails, and converges again at $t + \tau$. Assuming the valley-free, and prefer-customer policies, at no time between t and $t + \tau$ do the forwarding tables contain any loops.*

Proof. We say that a valley-free path that does not contain the down link is a *safe path*. Lemma A.5 proves that any AS forwarding packets continues to forward along its old path until it learns a safe path. Thus the path that a packet travels can be broken into three phases, although it does not necessarily include either of the first two phases. In (1), the packet travels along ASes that do not know a primary path, and are following Mechanism 3a which directs them to continue to forward along their old primary path. The packet continues like this until it reaches Q, after which it follows (2) a failover path until it reaches an AS that knows a safe path. Note, it may reach an AS that knows a safe path before it reaches Q, in which case phase (2) is skipped. Finally, it follows (3) a safe path to the destination. Lemma A.8 proves that once the packet reaches an AS that knows a safe path, it will continue to be forwarded only to ASes that know a safe path, i.e. once a packet reaches an AS that knows a safe path it jumps to phase (3). Additionally, once a packet reaches Q and is forwarded on a failover path, Mechanism 3a ensures that it can only leave the failover path once it moves to phase (3) by reaching an AS that knows a safe path, and thus it can never return to phase (1). Thus once a packet reaches a given phase it cannot return to an earlier phase and so we must only show that each phase individually is loop free in order to show that the entire path is loop free. Note that in a later phase a packet may revisit ASes already traversed in an earlier phase, but it will never loop indefinitely since a given phase cannot contain a loop within that phase. We know that phase (1) of the path is a prefixes of a previously converged paths, because Lemma A.5 ensures that any AS not in phase 3 continues to use its old converged primary path. Similarly, phase (2) is just the reverse of the prefix of a previously converged path, by Lemma A.5 ASes continue to use their old failover paths as well. Since converged BGP paths cannot contain loops, these phases of the path must be loop free. Lastly, while the safe path in phase (3) may change during convergence; Lemma A.8 guarantees that the forwarded packets will always traverse some valley-free path, and Lemma A.1 proves that valley free paths must be loop free.

□

The following theorem proves that as long as the AS graph does not contain any directed cycles containing only customer-to-provider edges, then regardless of the policies of the ASes involved R-BGP cannot deadlock in a state where some AS is still waiting to send out a withdrawal, or continues to forward along a withdrawn path.

Theorem A.10. *Regardless of policies, in a converged state, no AS is deadlocked waiting to send an update, and no AS is forwarding packets along a withdrawn or failover path.*

Proof. Mechanisms 3a and 3b allow an AS to delay withdrawals and continue forwarding packets on its old primary path if the AS might get a usable path from one of its neighbors. We prove that regardless of the actual policies used by the ASes themselves, if any AS is in this delayed withdrawal state, then there must be some other AS it is waiting on which can immediately send an update without waiting. We prove this by showing that any sequence of ASes waiting on each other must be valley-free and thus by Lemma A.1 cannot contain a loop. We define a dependency path to denote a sequence of ASes $a_1, a_2, \ldots, a_n$, such that AS a_i is waiting to hear an update from AS a_{i-1} before sending an update to AS a_{i+1}. If an AS, a_i in the dependency path is following a valley-free policy, then either a_{i-1} or a_{i+1} must be a customer. Thus, trivially, if all ASes in the dependency path are following valley free policies, then clearly such a dependency path must also be valley free and thus loop free by Lemma A.1. The interesting case occurs if some AS is not following a valley free advertisement policy. Regardless of the policies involved, however, any AS on such a dependency path must be one of two types: a *delayed AS* following mechanism 3b, delaying withdrawals to its neighbors, or an *unaware AS* that hasn't sent a withdrawal because it has not yet heard any update resulting from the down link. In the following, we show that for neither type of AS can the dependency go from a non-customer to a non-customer, i.e., the valley-free invariant must be maintained.

Mechanism 3b directs a *delayed AS*, a_i, to delay a withdrawal to neighbor a_{i+1}, only if it has some other neighbor a_{i-1}, such that a_i may still offer a_{i+1} a primary through a_{i-1}. Further, Mechanism 3b directs a_i to delay a withdrawal only if it may offer a valley-free path to a_{i+1}, thus either a_{i-1} or a_{i+1} must be a customer of a_i. Delayed ASes may be waiting on other delayed ASes, or unaware ASes.

An *unaware AS* has not yet heard any update indicating that the link is down, thus has not sent any update in response to the down link. If however, an unaware AS is to eventually send a withdrawal to its neighbor, then its current path must contain the down link. Thus while an unaware AS is not actually waiting on any other AS, it will not send a withdrawal until it hears a withdrawal from the next AS in the path it is currently advertising. This means that no dependency cycle can contain only unaware ASes, since at least the AS immediately upstream of the down link must be aware of link going down.

Thus we can break the loop into segments where each segment contains a series of delayed ASes a_j, a_{j+1}, ..., a_{j+k-1} and then a series of unaware ASes, a_{j+k}, a_{j+k+1}, ..., a_m. By the argument above, each delayed AS, a_i must maintain the valley-free invariant such that either a_{i+1} or a_{i-1} is a customer. Additionally, since delayed ASes only wait on neighbors who are advertising them paths with the valley free bit set, the first unaware AS in the sequence, a_{j+k} must be advertising a valley free path to the last delayed AS, a_{j+k-1}. Since a_{j+k} hasn't yet heard about the down link, it is effectively "waiting" for a withdrawal from the next AS in the path it advertised, a_{j+k+1}. But in order for it to have advertised a valley-free path in the first place, a_{j+k+1} must have advertised it a path with the valley-free bit set, and either a_{j+k-1} or a_{j+k+1} must be a customer. And as we continue through the unaware ASes in the sequence, each AS, a_l, in turn could only have advertised a valley free path if the previous neighbor in the loop, the neighbor whose path a_l is currently using, last advertised a valley free path, and either a_{l-1} or a_{l+1} is a customer. Similarly, the last unaware AS in the sequence, a_m could only have advertised a valley free path if the first delayed AS in the next sequence last advertised it a valley free path, and either a_{m-1} or a_{m+1} is a customer. Thus the valley free property is maintained between segments as well.

Therefore, in no case can the dependence go from non-customer to non-customer, and so the dependency path must be valley-free by the definition of valley-free. By Lemma A.1, valley-free paths never contain a loop, and thus if any AS is waiting to send an update, then, regardless of the policies of the respective ASes, there must be some AS that can send an update without waiting on any other AS. □

As the the culmination of the preceding lemmas, the follow proves, assuming the widest advertisement policy, that R-BGP meets the continuous connectivity guarantee.

Theorem A.11. *Consider a network that is in a converged state at time t, when a link goes down, and converges again at $t+\tau$.Assuming the valley-free, and prefer-customer policies, if an AS A knows a path to destination p at times t and $t+\tau$, then at any time between t and $t+\tau$ the forwarding tables contain a path from A to p.*

Proof. At any given time between t and $t+\tau$, we can break down the path from A to the destination into three phases, just as we did in Lemma A.9. In the first phase, all ASes just continue to forward along their old primary paths, and so no AS will drop the packet for lack of an available path. If the packet reaches AS Q, then by Lemma A.4 we know that Q will have a failover path on which to forward the packet. Then, in phase 2 ASes will continue to forward along their old failover paths and so again, no AS will drop the packet for lack of an available path. Finally, when the packet reaches phase 3, Lemma A.8 proves that the packet will not be dropped on its way to the destination. Additionally, Lemma A.9 proves that no AS will ever drop packets because of a routing loop. Thus, we can guarantee that no packet that A sends to a neighbor between t and $t+\tau$ will ever be dropped by any AS through whom it is forwarded, and thus all such packets must eventually reach the destination. Thus we must only prove that A will continue to forward its own packets.

Recognize that by following Mechanism 3, A will continue to forward its own packets unless it receives a withdrawal from all neighbors. Thus we need only prove that A has at least one neighbor from whom it never receives a withdrawal.

To do this, we prove by induction that between t and $t+\tau$, A will never receive a withdrawal from P, the neighbor of A through whom A will forward at time $t+\tau$. Suppose the path that P offers to A at $t+\tau$ is $P_i, P_{i+1}, \ldots, P_k$, where $P_i = P$ and P_k originates p. Then, following the same inductive argument we used at the end of Lemma A.4, we can see that if P is offering A a path at $t+\tau$, it must be doing so for the entire period between t and $t+\tau$. Note that during various parts of this period the path that it offers may be loopy, contain e, or be a failover path, but P will continue to offer N some path through the entire period.

Thus we have shown that A will never receive a withdrawal from P between t and $t+\tau$, and so A will continue to forward its own packets thus ensuring continuous connectivity. □