Reducing Transient Disconnectivity using Anomaly-Cognizant Forwarding

Andrey Ermolinskiy
Scott Shenker

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2008-120
http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-120.html

September 18, 2008

Reducing Transient Disconnectivity using Anomaly-Cognizant Forwarding

Andrey Ermolinskiy[†], Scott Shenker[†⋆]
[†]University of California at Berkeley, Computer Science Division,
[⋆]International Computer Science Institute

Abstract

It is well known that BGP convergence can cause widespread temporary losses of connectivity resulting from inconsistent routing state. In this paper, we present Anomaly-Cognizant Forwarding (ACF) - a novel technique for protecting end-to-end packet delivery during periods of convergence. Our preliminary evaluation demonstrates that ACF succeeds in eliminating nearly all transient disconnection after a link failure without the use of precomputed backup routes or altering the dynamics of BGP.

1 Introduction

It is widely known that BGP, the core Internet interdomain routing protocol, is susceptible to temporary connectivity failures during periods of convergence. A single event, such as a link failure or a policy change, can trigger a lengthy and complex sequence of route recomputations, during which neighboring ASes exchange updates and converge on a new globally-consistent set of routes. During this process, routers operate upon potentially inconsistent local views, which can lead to the emergence of temporary anomalies such as *loops* and *blackholes*. Both of these are considered undesirable, as they result in temporary losses of connectivity to the set of destinations affected by the event.

In order to prevent explosive growth of control traffic during the convergence process, BGP routers are typically configured to constrain the maximum rate of update propagation via the MRAI timer and [1] recommends setting its value to 30 seconds. Inevitably, limiting the rate of update dissemination lengthens the period of exposure to routing anomalies and several studies have reported prolonged and noticeable bursts of packet loss caused by BGP convergence. It has been shown that a single route change can produce up to 30% packet loss for two minutes or more [11]. Further, [18] reports loss bursts that last up to 20 seconds after a single route failure and up to 8 seconds after a route recovery event.

Today's Internet applications such as online games, streaming video delivery, and VoIP demand continuous end-to-end reachability and consistent performance. A recent study [9] establishes correlation between BGP update traffic and the quality of the end-user VoIP experience, demonstrating that about 50% of bad-quality voice samples occur within 10 minutes of a BGP update.

This problem has received considerable research attention and previous approaches can be broadly categorized into (a) those that attempt to expedite protocol convergence [3, 13, 17] and (b) those that seek to protect end-to-end packet delivery from the adverse effects of convergence. It has been suggested that mechanisms in the former category face an inherent limitation given the current scale of the Internet on the one hand and stringent demands of today's applications on the other. A scalable policy-based routing protocol that converges fast enough for applications such as interactive voice delivery still remains an elusive goal.

The second category of proposals includes mechanisms such as R-BGP [10], which advocates the use of precomputed failover paths for ensuring connectivity during periods of convergence. This scheme offers provable guarantees of reachability for single link failures, but these guarantees come at the cost of additional forwarding state and protocol complexity associated with the maintenance of backup routes and ensuring loop-free convergence. Further, ensuring connectivity in the face of multiple concurrent routing events would require routers to compute and maintain additional link-disjoint paths and the forwarding state requirements would present a serious scalability challenge.

Most recently, Consensus Routing [8] proposes to address transient disconnectivity by requiring BGP routers to agree on a globally-consistent "stable" view of forwarding state. In this context, stability means that a source domain can adopt a route to some destination in a given epoch only if each of the intermediate routers on the path adopts the respective route suffix in the same epoch, which guarantees absence of loops. In each epoch, routers participate in a distributed snapshot and consensus protocol in order to identify the set of "complete" BGP updates that satisfy stability. In contrast to much of prior work directed at reducing the duration of convergence, this scheme intentionally delays the adoption of BGP updates, so as to preserve the stability invariant. In the absence of

a stable forwarding path, consensus routing fails over to a transient forwarding mode that implements a heuristic such as detouring, backtracking, or backup paths.

In this paper, we present *Anomaly-Cognizant Forwarding* (ACF) - a new and complementary approach to improving Internet path availability and reducing transient disconnection. Rather than attempting to eliminate anomalous behavior by enforcing global consistency or shrinking the convergence time window, we accept inconsistent routing state as an unavoidable fact and instead develop a mechanism for *detecting* and *recovering* from such inconsistencies on the data path. While much of prior work has focused on extending BGP to improve its consistency and convergence properties, in this paper we consider a somewhat more disruptive approach that involves adding several fields to the packet header and inspecting them on the forwarding path. Our main hypothesis is that a single nearly trivial extension to conventional IP forwarding suffices to eliminate a dominant fraction of convergence-related disconnectivity. Our approach does not require routers to maintain multiple forwarding tables, nor does it require extending BGP or altering its timing dynamics.

2 Approach Overview

In broad terms, we view inconsistent BGP state and routing anomalies as unavoidable facts and approach the problem by extending the forwarding plane with a small amount of functionality that enables us to detect and recover from these anomalies. Toward this end, we augment the packet header with two additional pieces of state. First, a packet p originating in AS_s and destined to AS_d carries a *path trace* (denoted $p.pathTrace$) - a list of AS-level hops encountered by p on its path toward AS_d. At each hop, the border router inspects this field and appends its own AS identifier. The content of this field enables routers to detect and recover from occurrences of loops via a process that we describe more fully below. Second, each packet carries a *black list* (denoted $p.blackList$) containing an encoding of AS identifiers that are known to have possessed *deficient* routing state for p's destination at some point after packet's origination.

We say that a transit domain AS_t has *deficient* routing state for a destination AS_d at a particular instant in time if at that instant (a) AS_t lacks a valid policy-compliant path to AS_d, or (b) the path adopted by AS_t for destination AS_d results in a routing loop that causes packets to return back to AS_t.

At a high level, ACF packet forwarding proceeds as follows: a router first inspects $p.pathTrace$ and checks it for the presence of its local AS identifier, which would indicate a loop. If no loop is detected, the packet is forwarded as usual along the adopted route. Otherwise, the loop is viewed as evidence of deficient state and the router acts

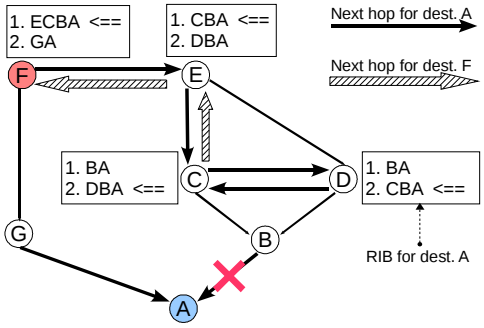

Figure 1: A sample AS-level topology with a transient forwarding loop.

upon it by moving every AS identifier belonging to the loop (which it knows from $p.pathTrace$) to $p.blackList$ and invoking the control plane, where the RIB is searched for the presence of an alternate path that does not traverse a blacklisted domain.

The second core component of our design is an alternate mode of packet delivery, which we term *recovery forwarding*. This mode helps ensure connectivity in situations where a router is unable to forward a packet because it does not possess a valid non-blacklisted path.

Forwarding in recovery mode is facilitated by a set of *recovery destinations*. When a transit router chooses to initiate recovery forwarding for a packet p, it adds the local AS identifier to $p.blackList$, copies p's destination address to an alternate location in the header, and redirects the packet to the address of some recovery destination, chosen at random from a well-known static set of potential destinations. In our current design and simulations, we assign the recovery destination role to a group of 10 well-connected Tier-1 ISPs[1].

The basic intuition that motivates this scheme is that the chosen recovery destination AS_r (or some intermediate router along the path to AS_r) is likely to possess a valid non-blacklisted route to the packet's original destination. As the packet travels toward AS_r in recovery mode, each router on the path first attempts to forward it to the original destination AS_d. If a usable non-blacklisted path is known, the router takes the packet off the recovery path and resumes normal-mode forwarding. Otherwise, the packet is sent to the next hop for destination AS_r. If, after reaching the recovery destination, the packet cannot be taken off the recovery path because AS_r does not possess a usable route to AS_d, the packet is dropped. Alternatively, in an effort to ensure eventual delivery, AS_r can re-initiate recovery forwarding via another destination. In the latter scenario, the process repeats until (a) the packet is taken off the recovery path by some destination that knows of a

[1]Internet service providers can offer recovery forwarding as a paid service for customers that wish to safeguard themselves from BGP-related connectivity failures.

working route to AS_d, (b) the packet is dropped because no recovery destination has such a route, or (c) the packet is dropped because its TTL expires.

We illustrate our scheme using the AS topology in Figure 1. Suppose that initially, domains *C* and *D* both use *B* as the next hop for destination *A*. In this example, failure of the inter-AS link $\langle A-B\rangle$ would cause *B* to send a withdrawal notification to its neighbors. Upon receiving the withdrawal, *C* and *D* would immediately switch to alternate paths $\langle D \rightarrow B \rightarrow A\rangle$ and $\langle C \rightarrow B \rightarrow A\rangle$, respectively. With conventional BGP, domain *C* has no way of determining that the newly-adopted path is invalid until it receives a withdrawal from *D* and, analogously, *D* considers $\langle C \rightarrow B \rightarrow A\rangle$ to be a valid route and adopts *C* as its next hop, thus causing a transient loop to emerge.

Suppose that domain *C* wishes to send a packet to an address in domain *A* and with ACF, packet forwarding proceeds as follows: Initially, *C* adds its local AS identifier to *p.pathTrace* and forwards the packet to its next hop - domain *D*. Upon receiving the packet, *D* appends its identifier to *p.pathTrace* and sends the packet back to *C*, which inspects *p.pathTrace* and detects a loop. It truncates *p.pathTrace* and, for each non-local AS identifier belonging to the loop (in this example only *D*), adds a corresponding entry to *p.blackList*. Next, *C* reattempts to forward the packet, this time avoiding the blacklisted forwarding table entry and discarding the corresponding route. In the example shown, *C* has no alternative working routes for destination *A*, so it adds itself to *p.blackList* and invokes recovery forwarding, choosing domain *F* as the recovery destination. *C* forwards the packet in recovery mode to *E* (its next hop for *F*) and the packet arrives with $p.pathTrace = \langle C\rangle$, $p.blackList = \langle C,D\rangle$. Upon receiving the packet, *E* first attempts to forward *p* to its original destination (*A*), but discovers that both its current next hop (*C*) and the alternate path through *D* are blacklisted in the packet's header and discards the respective routes. Lacking other alternate paths, *E* adds itself to *p.blackList* and forwards the packet further along the recovery path to its peer *F*. Analogously, *F* determines from the blacklist that its next hop *E* does not posses a valid path and purges the respective route from its RIB. However, *F* knows of an alternate working route $\langle G \rightarrow A\rangle$ and adopts it, causing *p* and all subsequent packets destined to *A* to be forwarded via *G*. Eventually, BGP path withdrawals will propagate through the topology and reach *F*, causing it to expose the route $\langle F \rightarrow G \rightarrow A\rangle$. During the transient period of inconsistency, however, the *pathTrace* and *blackList* state being propagated on the data path enables us to discover a valid alternate route and preserve end-to-end packet delivery.

Before we proceed to a detailed description of the design, we make two high-level observations about our approach. First, since ACF utilizes two distinct modes of forwarding (i.e., *normal* and *recovery* modes), it can cause some packets to traverse multiple distinct paths to the destination during periods of convergence. For example, AS_s may initially attempt to send a packet to AS_d via a path P_1, but one of the intermediate hops may decide to re-route it via a recovery destination, which, in turn, can choose to forward the packet via P_2 - an alternate path to AS_d that is link-disjoint from P_1. Unlike earlier work on failover BGP paths [10], our mechanism does not require routers to construct an explicit set of failover routes and to maintain multiple forwarding table entries. In ACF, the two modes make use of the same forwarding table and we try to *discover* an alternate route dynamically by extending the forwarding plane.

Second, we do not assume that the set of paths to recovery destinations is stable and that every AS possesses a working loop-free route to some recovery destination at all times. Indeed, certain failure scenarios (e.g, a core link failure) can result in disruption of paths to multiple endpoints, including those that serve as recovery destinations, and clearly, our design must succeed in retaining end-to-end connectivity in the face of such failures. Thankfully, there is a simple and effective solution that enables us to handle such cases - we protect recovery-path forwarding against routing anomalies using precisely the same mechanism that we use to safeguard packet delivery on the normal forwarding path, i.e., using the *pathTrace* and *blackList* fields in the packet header.

3 The Design of ACF

3.1 Packet header state

ACF adds the following fields to the packet header:

recoveryMode: A single-bit flag indicating the current forwarding mode (*normal* or *recovery*).

finalDestAddr: In recovery mode, this field carries the packet's actual destination address (i.e., its destination prior to redirection).

pathTrace: An ordered list of AS-level hops traversed by the packet in the current forwarding mode.

blackList: A set of AS identifiers that are known to possess deficient routing state for the packet's original destination.

blackListRecov: A set of AS identifiers that are known to possess deficient routing state for the packet's designated recovery destination.

In our current design, *pathTrace* is represented as a linear list of 16-bit AS numbers. The length of this field is either fixed or selected from a small set of predefined values. In Section 4, we expand upon this point and evaluate the corresponding space requirements in the packet

```
// Check for loops
if for some i: p.pathTrace[i] = localASNum then
    for j ← i+1 to p.pathTrace.length do
        Add p.pathTrace[j] to p.blackList
    p.pathTrace ← Prefix(p.pathTrace, i)

// Validate the next hop
nextHop ← GetNextHop(p.destAddr)
if nextHop ∈ p.blackList then
    FindAlternateRoute(p.destAddr, p.blackList)
    nextHop ← GetNextHop(p.destAddr)
if nextHop = NONE then
    // Switch to recovery mode
    Add localASNum to p.blackList
    p.recoveryMode ← TRUE
    p.pathTrace ← EMPTY
    p.finalDestAddr ← p.destAddr
    p.destAddr ← SelectRecovDest(p.blackList)
    nextHop ← GetNextHop(p.destAddr)
Append localASNum to p.pathTrace
ForwardPacket(p, nextHop)
```

Algorithm 1: Pseudocode for normal-mode ACF.

header. On the other hand, *blackList* and *blackListRecov* can be represented using a space-efficient Bloom filter encoding (note that AS identifiers are never removed from blacklists).

3.2 Forwarding algorithm

When a packet p arrives at a router, its *recoveryMode* flag is inspected to determine the appropriate forwarding mode. We illustrate normal-mode ACF using high-level pseudocode in Algorithm 1. First, the router checks the *pathTrace* field for the presence of its local AS number. If a loop is detected, all AS components of the loop are added to $p.blackList$ and the path trace is truncated to exclude the loop. Next, the forwarding table is consulted to obtain the next hop for p's destination and the content of $p.blackList$ is inspected. If the next-hop AS is present in $p.blackList$, the current route is discarded and the control plane (*FindAlternateRoute*) is invoked to find and install an alternate non-blacklisted route.

In *FindAlternateRoute* the standard BGP route selection process is invoked to identify a new preferred route that will be used for forwarding p and all subsequent packets destined to the same prefix and, crucially, all blacklisted routes are excluded from consideration during this process. We investigated and evaluated two alternative methods for deciding whether to exclude a particular candidate route $R = \langle AS_1^R, AS_2^R, ...AS_k^R \rangle$ for a given packet p:

1. Examine only the next hop and exclude R iff $AS_1^R \in p.blackList$.

2. Examine the entire AS-PATH attribute and exclude R iff $\exists i$ such that $AS_i^R \in p.blackList$.

Consider a scenario, in which AS_s knows of two distinct routes to AS_d, namely $\langle AS_1 \rightarrow AS_2 \rightarrow AS_d \rangle$ and $\langle AS_3 \rightarrow AS_2 \rightarrow AS_d \rangle$. Initially, it tries to forward the packet via AS_1, but the packet returns with $\langle AS_s, AS_1, AS_2, AS_4 \rangle$ in its *pathTrace*, causing AS_s to blacklist AS_1, AS_2, and AS_4. Using method (1), AS_s would next attempt to forward via AS_3, but this would result in wasted effort if AS_3 does not know of any alternate paths to AS_d that do not go through AS_2. Conversely, method (2) would require AS_s to discard its path through AS_3 and invoke recovery forwarding due to absence of other alternatives. In this situation, skipping AS_3 can result in a lost opportunity to forward the packet via an efficient alternate route if AS_3 does indeed possess such a route.

We examined both alternatives and found that the second method is substantially more effective in reducing transient packet loss for the set of failure cases we simulated. It allows problematic paths to be detected and discarded more quickly and reduces the number of hops it takes for a packet to home in on a valid alternate route. Note that as a further optimization, we could also evaluate the criterion of method (2) on the data path (currently, we check only the next hop), but this improvement would come at the expense of additional processing overhead and forwarding state, which our approach explicitly seeks to avoid. Hence, our current design adopts a compromise by validating only the next hop on the data plane and performing full AS-PATH inspection only upon evidence of anomalous behavior.

If *FindAlternateRoute* fails to identify and install another working route, recovery forwarding is invoked. The router adds its local AS number to $p.blackList$, clears $p.pathTrace$ and $p.blackListRecov$, chooses a non-blacklisted recovery destination, and looks up the corresponding next hop.

Forwarding in recovery mode proceeds analogously and we omit the pseudocode due to space constraints. In this mode, a router first looks up and validates the next hop for $p.finalDestAddr$ and if a non-blacklisted path is found, normal-mode forwarding is resumed by clearing $p.pathTrace$, and setting $p.destAddr \leftarrow p.finalDestAddr$. Otherwise the local AS number is inserted into $p.blackList$ and recovery-mode forwarding is continued. The router inspects $p.pathTrace$ and, if a loop is detected, truncates it and augments $p.blackListRecov$. If necessary, *FindAlternateRoute* is invoked to find an alternate non-blacklisted path and if no such paths exist, the router initiates recovery forwarding via another destination.

4 Preliminary Evaluation

The preliminary evaluation we present in this section focuses on addressing three key questions: (1) How effective is ACF at sustaining end-to-end connectivity during

convergence? (2) In the absence of precomputed backup routes, how long does it take to recover a packet from an anomalous path and identify an alternate working route? (3) How significant is the packet header overhead incurred by our scheme?

Methodology: To answer these questions, we implemented an event-driven parallel simulator that enables us to study the dynamics of BGP convergence in realistic Internet-scale AS topologies and simulate packet forwarding at an arbitrary point in time during the convergence process. Our initial experiments examine the effects of inter-AS link failures on end-to-end reachability and focus on failures of access links that connect to a multi-homed edge domain. We use the CAIDA AS-level topology from May 12, 2008 [2] annotated with inferred inter-AS relationships. The topology contains 27969 distinct ASes and 56841 inter-AS links. Following standard convention, our simulator implements "valley-free" route propagation policies [7] and customer routes are always preferred over peer and provider routes.

The topology includes 12937 multihomed edge ASes and a set of 29426 adjacent provider links. We conduct a failure experiment for each provider link $\langle AS_p - AS_d \rangle$ in this set. We begin by simulating normal BGP convergence that results in adoption of consistent policy-compliant paths toward the destination AS_d. Next, we fail its link to AS_p, simulate packet forwarding from each AS to AS_d during the period of reconvergence, and identify the set of ASes that experience temporary loss of connectivity to AS_d during this period. With traditional forwarding, a source domain is considered disconnected if an intermediate router on its path to AS_d drops a packet because it does not possess a route or if the packet's TTL (initially set to 32 hops) expires, indicating a forwarding loop. With ACF, a domain is disconnected if its packet is dropped at the recovery destination and upon TTL expiration.

Transient disconnection after link failures: As expected, we found that BGP with conventional forwarding exhibits a substantial amount of transient disconnectivity. 51% of failures cause some of the ASes to experience connectivity loss and 17% of failures cause at least half of all ASes in the topology to lose connectivity. Figure 2 plots the fraction of disconnected domains for the cumulative fraction of failure cases and demonstrates the effectiveness of ACF. In 84% of failure cases that produce some disconnectivity with conventional forwarding, ACF fully eliminates unwarranted packet loss and further, in 96% of such cases no more than 1% of all ASes experience disconnection. The figure also illustrates that recovery forwarding plays a pivotal role in protecting packet delivery and ensuring connectivity in the face of anomalies. In a small number of cases (0.2% of failure cases) our scheme offers little or no measurable improvement, leaving over

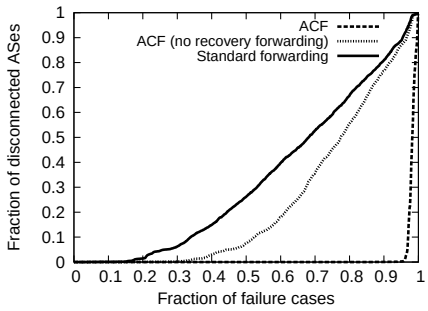

Figure 2: Prevalence of transient disconnection after a single provider link failure. The x-axis represents the fraction of all failure cases that cause some disconnectivity with traditional forwarding.

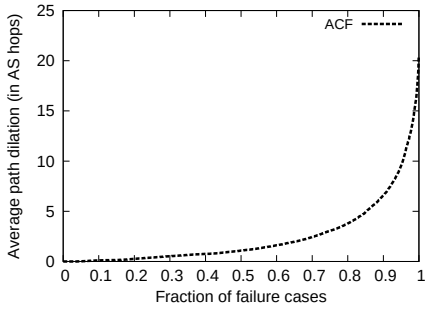

Figure 3: Average path dilation with ACF.

90% of the topology disconnected and further inspection revealed that in most of these cases, packets fail to discover a working route within 32 hops.

Path efficiency: By not maintaining a precomputed set of efficient alternate routes and instead letting packets discover them dynamically, our scheme can increase the number of hops a packet traverses during periods of instability. This overhead can be attributed to the fact that packets can encounter loops and that finding a working path can require detouring to a recovery destination. We measured this overhead in the above experiment and Figure 3 plots the path dilation (averaged over all ASes) for the cumulative fraction of failure cases. This quantity is computed by subtracting the length of the final route (in AS hops) adopted after reconvergence from the length of the longest path a packet would have to traverse under ACF before reaching its destination. In 65% of failures that cause loss under traditional forwarding, ACF recovers packets using no more than two extra AS hops and only 9% of failures incur the cost of 7 hops or more.

Packet header overhead: Table 1 shows the maximum number of entries in the *pathTrace* and *blackList* header fields for a representative sample of failure cases correspoding to 0%, 0.09%, 0.9%, 9%, and 90% transient dis-

% disconnected	0%	0.09%	0.9%	9%	90%
pathTrace len.	11	16	16	20	13
blackList len.	4	11	9	11	16

Table 1: Maximum number of *pathTrace* and *blackList* entries in a representative sample of failures cases.

connection with ACF[2]. In the worst case, *pathTrace* consumes 40 bytes assuming that each entry is a 16-bit AS number. Up to 16 entries are added to *blackList* and a Bloom filter representation with 1% lookup error rate would require 10 bytes.

In summary, our initial evaluation suggest ACF to be a promising approach that significantly reduces transient packet loss and incurs reasonable bandwidth and latency overheads. However, the results presented here are only a first step toward understanding its full behavior in a complex Internet-scale environment and future work will include evaluating ACF under a broader range of scenarios that include failures of transit links, multiple concurrent failures, link recovery, and BGP policy changes.

5 Discussion and Future Work

In this section, we briefly discuss several concerns pertaining to ACF and outline directions for further study.

Feasibility of deployment: ACF introduces several changes to the core mechanisms of IP forwarding and can thus be seen as facing a substantial barrier to adoption. More concretely, ACF requires adding several fields to the packet header, as well as introducing additional logic on the forwarding path. While clearly non-trivial, we believe that packet format issues can be addressed via the use of IP options and/or shim headers. Investigating these issues in detail and proposing a viable path toward deployment are two essential topics of future work.

Packet processing overhead: Our scheme adds complexity and computational overhead to the forwarding plane. We note that $FindAlternateRoute$ - the most significant source of overhead in ACF - is invoked only during periods of instability and only for the purpose of replacing a broken route whose continued usage would otherwise result in packet loss. In the common case, the overhead reduces to checking *blackList* and *pathTrace* for the presence of the local AS number - operations that incur the cost of a single Bloom filter lookup and a linear scan, respectively. Both operations admit efficient implementation in hardware and parallelization. Finally, if the cost of a vector scan at each hop is deemed unacceptable, loop detection and recovery can be deferred until TTL expiration and handled at the control plane.

[2] *recovBlackList* is not shown because recovery destination paths remain stable in this experiment.

ACF and routing policies: Due to recovery forwarding, packets in ACF can be forwarded along a path which violates ISP export policies when viewed from an end-to-end perspective. At the same time, each individual forwarding decision in ACF respects policies by considering only the set of exported routes available in the RIB. In particular, only policy-compliant paths are used in recovery mode to guide a packet toward a recovery destination. ACF envisions the emergence of a new inter-ISP relationship landscape, where a group of highly-connected Tier-1 networks would provide the recovery destination service to multihomed customers that wish to safeguard themselves from the adverse effects of routing convergence. Viewed in this manner, our scheme can be said to provide policy-compliant forwarding via an intermediate destination.

6 Related Work

The undesirable side-effects of BGP convergence have been studied extensively through measurements and simulations [11, 12, 14, 16, 18]. Prior work on addressing this problem includes a family of protocol extensions and heuristics for accelerating BGP convergence and some examples include ghost flushing [3], root cause notifications [15], consistency assertions [17], and limiting BGP path exploration [5].

Another set of techniques, to which our scheme belongs, focuses on protecting end-to-end packet delivery from the adverse effects of convergence and recent work in this area includes Resilient BGP [10] and Consensus Routing [8]. Analogously to both schemes, ACF implements two logically distinct modes of forwarding, which are differentiated using an extra bit in the packet header. R-BGP advocates the use of precomputed failover paths and requires routers to maintain multiple forwarding table entries for some destinations. To achieve loop-freeness, R-BGP introduces an assumption regarding route selection preferences and augments BGP update messages with root cause information. In contrast, our scheme works with general preference policies and requires no changes to the routing protocol, but does not offer provable guarantees of reachability. Consensus Routing ensures loop-freedom by enforcing a globally-consistent view of forwarding state, achieved by strategically delaying the adoption of BGP updates. Several transient forwarding modes are used to ensure high availability and our approach borrows the idea of detouring via a highly-connected domain. Consensus Routing also modifies the forwarding path and the per-hop packet encapsulation used in the backtracking transient mode is conceptually analogous to ACF's *pathTrace*. Our main insight is that carrying the list of prior hops on the data path also provides the ability to detects loops and thus, global consistency is extraneous if packets can be recovered from

loops and redirected via a reasonably efficient path. Consensus Routing delays the adoption of new routes by up to several minutes which, in certain scenarios, can have an adverse effect on end-to-end reachability.

Failure-Carrying Packets [**?**] is a recent proposal for link-state protocols that protects end-to-end delivery by augmenting packets with information about link failures. ACF adopts an analogous approach, but focuses on interdomain policy routing. We compared ACF with the strawman design for path-vector FCP presented in [?] and found that FCP improves end-to-end path availability for only a fraction of failure cases, demonstrating an improvement comparable to ACF without recovery-mode forwarding. Compared to FCP, our scheme does not require routers to precompute and cache a set of alternate forwarding table entries and incurs a smaller per-packet processing overhead (control-plane path reselection is invoked much less frequently). A detailed performance comparison with FCP is a topic of future work.

7 Acknowledgments

We are grateful to Philip Brighten Godfrey for his insightful comments on earlier versions of this paper and the anonymous reviewers for their valuable and constructive feedback.

References

[1] A border gateway protocol 4 (BGP-4). `http://www.ietf.org/rfc/rfc4271.txt`.

[2] Caida inferred as relationships dataset. `http://www.caida.org/data/active/as-relationships/`.

[3] A. Bremler-Barr, Y. Afek, and S. Schwarz. Improved bgp convergence via ghost flushing. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE*, volume 2, pages 927–937 vol.2, 2003.

[4] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. van der Merwe. Design and implementation of a routing control platform. In *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, pages 15–28, Berkeley, CA, USA, 2005. USENIX Association.

[5] J. Chandrashekar, Z. Duan, Z. L. Zhang, and J. Krasky. Limiting path exploration in bgp. volume 4, pages 2337–2348 vol. 4, 2005.

[6] P. Francois and O. Bonaventure. Avoiding transient loops during igp convergence in ip networks. In IEEE, editor, *Proceedings of IEEE INFOCOM 2005*, March 2005.

[7] L. Gao and J. Rexford. Stable internet routing without global coordination. *IEEE/ACM Trans. Netw.*, 9(6):681–692, 2001.

[8] J. P. John, E. Katz-Bassett, A. Krishnamurthy, and T. Anderson. Consensus routing: The internet as a distributed system. In *5th USENIX Symposium on Networked Systems Design & Implementation*, April 2008.

[9] N. Kushman, S. Kandula, and D. Katabi. Can You Hear Me Now?! It Must be BGP. In *Computer Communication Review*, March 2007.

[10] N. Kushman, S. Kandula, D. Katabi, and B. Maggs. R-bgp: Staying connected in a connected world. In *4th USENIX Symposium on Networked Systems Design & Implementation*, Cambridge, MA, April 2007.

[11] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. Delayed internet routing convergence. In *SIGCOMM '00: Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 175–187, New York, NY, USA, 2000. ACM.

[12] C. Labovitz, A. Ahuja, and F. Jahanian. Experimental study of internet stability and backbone failures. In *FTCS '99: Proceedings of the Twenty-Ninth Annual International Symposium on Fault-Tolerant Computing*, page 278, Washington, DC, USA, 1999. IEEE Computer Society.

[13] J. Luo, J. Xie, R. Hao, and X. Li. An approach to accelerate convergence for path vector protocol. In *Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE*, volume 3, pages 2390–2394 vol.3, 2002.

[14] Z. M. Mao, R. Govindan, G. Varghese, and R. H. Katz. Route flap damping exacerbates internet routing convergence. *SIGCOMM Comput. Commun. Rev.*, 32(4):221–233, 2002.

[15] D. Pei, M. Azuma, D. Massey, and L. Zhang. Bgp-rcn: improving bgp convergence through root cause notification. *Comput. Netw. ISDN Syst.*, 48(2):175–194, 205.

[16] D. Pei, X. Zhao, D. Massey, and L. Zhang. A study of bgp path vector route looping behavior. In *ICDCS '04: Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, pages 720–729, Washington, DC, USA, 2004. IEEE Computer Society.

[17] D. Pei, X. Zhao, L. Wang, D. Massey, A. Mankin, S. Su, and L. Zhang. Improving bgp convergence through consistency assertions. *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 2:902–911 vol.2, 2002.

[18] F. Wang, Z. M. Mao, J. Wang, L. Gao, and R. Bush. A measurement study on the impact of routing events on end-to-end internet path performance. In *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 375–386, New York, NY, USA, 2006. ACM.