This paper was presented as part of the 14th IEEE Global Internet Symposium (GI) 2011 at IEEE INFOCOM 2011

Stabilizing BGP Routing without Harming Convergence

Xiaoqiang Wang
School of Computer
National University of Defense Technology
Changsha, China
wangxiaoqiang@nudt.edu.cn

Olivier Bonaventure
ICTEAM
Université catholique de Louvain
Louvain-la-Neuve, Belgium
Olivier.Bonaventure@uclouvain.be

Peidong Zhu
School of Computer
National University of Defense Technology
Changsha, China
pdzhu@nudt.edu.cn

***Abstract*—RFD and MRAI are the only two built-in mechanisms in BGP router against unstable routes, they can however negatively impact the convergence. In this paper, we propose a churn aggregation approach *CAGG* to stabilize BGP routing without harming convergence. CAGG is based on the observation that AS_PATH change is the dominant cause for BGP updates and only a small number of AS_PATHs are explored by each highly active prefix. A CAGG equipped router converts the multiple AS_PATHs explored by a highly active prefix into an aggregated path, and propagates the aggregated path instead to reduce the number of resulted BGP updates from AS_PATH changes. Our experiments with real BGP data show that CAGG can reduce as much as 50% of BGP updates, and 60% of BGP path exploration duration in its best case, while on average 28.1% and 32% respectively across 36 RouteViews monitors. Furthermore, CAGG is shown to perform better than both RFD and PED[1] in reducing BGP updates, path exploration duration and accelerating BGP convergence, at the cost of buffering around 5,000 AS_PATHs.**

***Index Terms*—BGP,Interdomain Routing, RFD, MRAI, PED, BGP Churn,Convergence.**

I. Introduction

BGP is the defacto standard of inter-domain routing protocol that glues the Internet together. As a consequence of continuous evolution of Internet, BGP is now facing more and more scalability problem, especially for the increasing entries in BGP route table and the rate of BGP updates (*BGP Churn*)[2]. The rapidly increasing BGP churn and slow convergence inherent to BGP path vector nature are the most two important issues in BGP dynamics. Actually the slow convergence, known as path exploration, also increases the BGP churn, so the former issue can benefit from solutions dedicated to accelerating path exploration as well.

The researches dedicated to either reducing churn or accelerating path exploration fall into two categories: *Active* and *Reactive* solutions. In active solutions, such as RCO[3], BGP-RCN[4] and EPIC[5], updates resulted from topology change are tagged with the cause so that the a receiver router can converge to a backup route independent of this change upon receiving the first update. On the contrary, there are more reactive solutions such as Route Flap Damping(RFD)[6], Minimum Route Advertisement Interval(MRAI)[7], Withdrawal Rate Limiting(WRATE)[7], Ghost Flush[8], Sender-side Loop Detection(SSLD)[9], BGP Consistency Assertions[10], Differentiated Update Processing(DUP)[11], Path Diversity Aware Routing(PDAR)[12] and recently proposed Path Exploration Damping(PED)[1], RFD+RG[13]. However, only two approaches, RFD and MRAI, have been implemented by router vendors.

Both RFD and MRAI are becoming less attractive than a few years ago in the early Internet for the rising concern about routing convergence as the emergence of many real time Internet-based applications, such as Skype, network bank, video conference and so on. The weakness of RFD lies in the fact that it is very hard to differentiate path exploration from persistent route flaps with only prefix penalty. Several modifications have been proposed for RFD to lower the false positive ratio in labeling a well-behaved prefix as unstable one, for example, [14] proposes to rise the damping threshold while [15] uses selective route flap damping. However, they can not solve the problem RFD is facing in essence. The classical RFD comes to its end when we can not afford the aftereffect of its mistake that [16] recommends to turn RFD off. MRAI allows a router to privately explore its alternative choices for best route without exposing its neighbors the intermediate step, thus reduces the number of updates during path exploration[17], at the cost of delaying convergence. The default MRAI configuration is not recommended any more in[18]. Indeed, ISPs have seldom countermeasures against those flapping prefixes when more and more networks choose to turn off RFD and MRAI.

To reduce BGP churn, we propose a novel approach, *Churn Aggregation(CAGG)*, against those highly active prefixes without harming BGP convergence, considering that a significant fraction of BGP churn is associated to a really small fraction of highly active prefixes[19]. For each flapping prefix, a CAGG equipped router converts the multiple AS_PATHs used by this prefix into an aggregated path to reduce the number of BGP updates due to changes of AS_PATH attribute. Our contributions in this paper are twofold: 1) we find the so-called *Path Locality* that only a small number of AS_PATHs are explored by a highly active prefix(or flapping prefix); 2) based on that, CAGG converts the several transient paths into a normalized one, to reduce the updates caused by AS_PATH change. CAGG is shown to perform better than RFD[6] and a MRAI similar method PED[1] in both reducing BGP updates and accelerating BGP convergence.

The remainder of this paper is structured as follows. Section 2 explains our motivation, and CAGG is introduced in Section 3. Section 4 compares the CAGG approach with RFD and PED in performance and convergence, and then evaluates the memory cost. Finally we conclude this paper in Section 5.

II. Motivation

To understand the components of BGP churn, we analyzed the BGP messages sent by 36 monitors that peer with the RouteViews collector during December 2009. We first filtered all session resets from the dataset. Figure 1 sketches the components of routing changes observed from these monitors by comparing each BGP message with the previous one for the same prefix. As we can see, the reason for sending a BGP Update is usually either a change in AS_PATH or a change in BGP community. These two types of changes are responsible for 98% of the observed BGP updates. A change in AS_PATH reflects either a change of route or sometimes a traffic engineering action with a change in AS_PATH prepending.

BGP communities are different. These BGP attributes are used for various purposes, ranging from tagging routes to indicating policy actions. Most of these communities are used only within a single ISP and propagating them over eBGP sessions is useless. In our analysis, seldom prefixes are involved in community changes in 14 monitors. This indicates that the corresponding operator has chosen to filter the BGP communities. However, nearly all the observed prefixes are associated to at least one community change in the other 22 monitors. All operators should deploy BGP filters to filter the BGP communities over eBGP sessions to reduce the BGP churn. In the remainder of this paper, we assume that operators have deployed outbound filters to avoid announcing BGP Updates when only BGP communities have changed.

Changes in AS_PATH can be caused by policy changes, but are often due to route flapping. The principle of route flapping can be depicted on the basis of the topology in Figure 3, in which each node represents an AS and only prefix d is originated from AS 1. The link or node in abstracted routing topology responsible for an observed flapping prefix is called *flapping originator*, or *originator* for short. The route besides AS7 marked with an asterisk is the best route for prefix d selected to forward packets. Route flapping can be caused by several pathological reasons, such as hardware/software errors, connection errors, policy inconsistency, etc. Observed from the originator itself, route flapping can be either repeatedly announcing, withdrawing then followed by re-announcing, or repeatedly oscillating among several routes. The latter behavior is usually caused by policy or configuration errors, such as MED-induced divergence[20]. The behaviors of route flapping vary as the relative position between the originator and the observer as well. For example in Figure 3 the link between AS 1 and AS 2, denoted by (1, 2), is assumed to periodically fail and then restore, and we suppose that AS 7 favors the route learned from AS 4 over AS 5, then in turn over AS 6. Standing on AS 4, alternate announcement $d, 21$ and withdrawal are observed, and AS 8 however sees several announcements eventually converged to either $d, 7631$ or $d, 7421$.

When a route flaps, RouteViews monitors should often announce alternating AS_PATHs. To verify this, we analysed the BGP updates from 6 peering ASes of RouteView collector, including 2 Tier1 ASes (AS3356 and AS1239), 2 Tier2 ASes(AS1221 and AS13030), and 2 stub ASes(AS14608 and AS3130). We chose this subset for geographical and topological diversity. In each of the 6 Ases, the prefixes were first sorted according the number of observed updates associated to them during December, 2009. Then we stored the different AS_PATHs that were advertised for each of the top 10,000 prefixes that contributed to most of the BGP Updates. For each prefix, we define a likelihood P as the probability that a received BGP Update contains an AS_PATH among the 3 most frequent AS_PATHs for this prefix over December 2009. Figure 2 plots the CCDF of this likelihood for each of those 6 ASes. Each data point $(x\%, y\%)$ in the CCDF indicates that $y\%$ of the top 10K prefixes have their probability larger than $x\%$. As we can see, more than 70% of the top 10,000 prefixes in all those 6 ASes have the likelihood higher than 60%. And 15% of these prefixes are found to have explored fewer than 3 paths during one entire month in AS3130 while this fraction in AS14608 is 5% and 10%-15% in the other 4 ASes. We want to emphasise that the fraction of AS_PATH occurrences covered by the top 3 paths per prefix in a shorter time window may be higher, since the flapping routes per prefix may vary as time goes on. This *Path locality* means that there are only a few AS_PATHs observed from an AS to reach a highly active prefix.

III. CAGG Model

The measurements above show that AS_PATH changes are an important contributor to BGP churn. Furthermore, a small fraction of highly active prefixes are responsible for a large fraction of the BGP churn[19] and the highly active prefixes explore only a small number of AS_PATHs. These findings inspire our proposed Churn Aggregation technique. Our idea is to convert the multiple AS_PATHs used by a highly active prefix into an aggregated path to reduce the number of BGP updates due to changes of AS_PATH attribute.

From a router's point of view, for a downstream neighbor N regarding prefix d, BGP routing information propagated to this neighbor is denoted as $R_{N,d} = (\langle r_1, t_1\rangle, \langle r_2, t_2\rangle ... \langle r_n, t_n\rangle)(n \geq 1)$ while each of the routes r_i is disseminated at time t_i. The objective of Churn Aggregation (CAGG) is to compute a new routing $R'_{N,d}$, in which the number of messages is reduced. For simplicity, we name BGP equipped with CAGG **aBGP(aggregation BGP)**. aBGP operates like a normal BGP router, except that it uses an outbound filter on eBGP sessions.

To understand intuitively the operation of CAGG, let us consider the topology depicted in Figure 3. Supposing that link $(1, 2)$ is periodically flapping, and path 247 is faster than 257 in propagating routing changes. Observed from AS

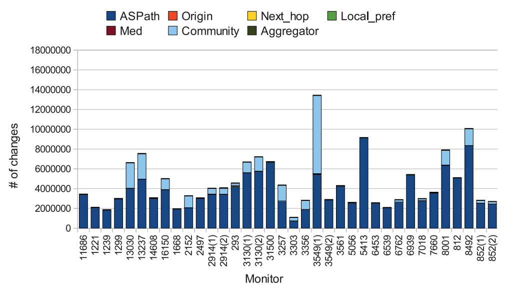


Fig. 1. Components of routing changes

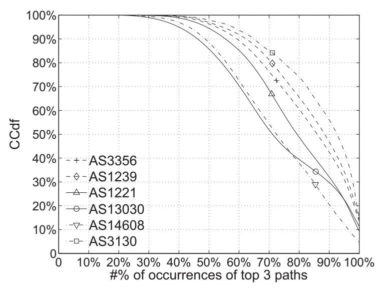


Fig. 2. CCDF of the top 3 paths' occurrences per prefix in top 10K prefixes

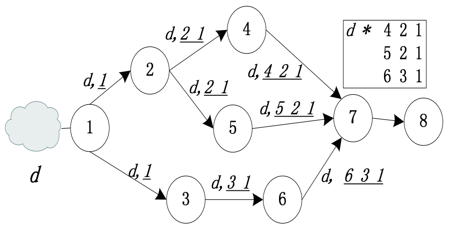


Fig. 3. Sample topology of BGP routing

8, routes towards d oscillate between the two sequences, $7421 \rightarrow 7521 \rightarrow 7631$ and $7631 \rightarrow 7421$. Now the CAGG deployed in AS 7 maps all the three paths above to an aggregated path $7\{2,3,4,5,6\}1$, and propagated to AS 8 the aggregated path instead. Then the two sequences are converted to $7\{2,3,4,5,6\}1 \rightarrow 7\{2,3,4,5,6\}1 \rightarrow 7\{2,3,4,5,6\}1$ and $7\{2,3,4,5,6\}1 \rightarrow 7\{2,3,4,5,6\}1$ respectively, and only the first route is necessary while the others are ignored.

The CAGG filter operates as follows :

- *Decomposing*. Each $r_i \in R_{N,d}$ is further modeled as a two values tuple $\langle p_i, q_i \rangle$ where p_i is its AS_PATH attribute and q_i the other attributes[1]. Correspondingly, the vectors of p_i and q_i are denoted as $P_{N,d}$ and $Q_{N,d}$, and $R_{N,d} = (P_{N,d}, Q_{N,d}, T_{N,d})$ while $T_{N,d}$ is the time vector of $(t_1, t_2 ... t_n)$.
- *Mapping*. CAGG defines a function f with domain $P_{N,d}$ and range $P'_{N,d}$ that computes for each $i (1 \leq i \leq n)$ and $p_i \in P_{N,d}, p'_i \in P'_{N,d}$,
$$p'_i = \begin{cases} f(p_i) & p_i \neq \phi \\ p_i & p_i = \phi \end{cases}$$
This step does not modify BGP Withdrawal messages.
- *Reforming*. A new routing $R'_{N,d}$ is constructed such that $R'_{N,d} = (P'_{N,d}, Q_{N,d}, T_{N,d})$.
- *Propagating*. Not all the messages in $R'_{N,d}$ are propagated to N since some of the neighboring entries become identical now, and a BGP router only advertises changes. The CAGG filter advertises only the entries that are different from the previous one. For each $i = n \rightarrow 2$, the tuple $\langle r'_i, t_i \rangle$ is removed if $r'_i = r'_{i-1} (p'_i = p'_{i-1} \& q_i = q_{i-1})$, and the rest are propagated to N according to the timestamp vector $T_{N,d}$.

Observed over a long time, the lifetime of route r_i in $R_{N,d}$ propagated to N for prefix d ranges from t_i to t_{i+1}, so does r'_i in $R'_{N,d}$. Huston *et al.* decomposed BGP convergence into two distinct parts of pursuing *reachability* and *optimality* respectively[1]. For a destination prefix, reachability is guaranteed as long as there is at least one route available. And optimality is not achieved until every router learns the best route, which is usually indicated by a long enough silent period, during which no new routes are received for that prefix. For any time range $[t_i, t_{i+1}]$, the reachability of *aBGP* is the same to *BGP* since the type of r'_i and r_i are exactly the same, either Announcement or Withdrawal. Then *aBGP* and BGP are sharing the same time vector that optimality is not delayed in *aBGP* as well. In fact, routing convergence is accelerated in *aBGP* and formal proofs are in[21].

[1]BGP Withdrawal is denoted by the empty path ϕ and empty attributes ϕ as well.

Algorithm 1 Map BGP route r_i to aBGP route r'_i

```
Input: $r_{i-1}, r_i, r'_{i-1}, H_{N,d}$
Output: $r'_i$
1:  if $r_i = r_{i-1}$ then
2:      $r'_i \leftarrow r'_{i-1}$ {$r_i$ is a duplicated update}
3:  else
4:      $r'_i \leftarrow r_i$ {$r'_i$ is initialize to be $r_i$}
5:      if $r_i \neq$ WITHDRAWAL then
6:          if $r_i.AS\_PATH \prec r'_{i-1}.AS\_PATH$ then
7:              $r'_i.AS\_PATH \leftarrow r'_{i-1}.AS\_PATH$
8:          else
9:              $r'_i.AS\_PATH \leftarrow agg(r_i.AS\_PATH, H_{N,d})$
10:         if $r'_i.AS\_PATH \neq r_i.AS\_PATH$ then
11:             $r'_i.AGGREGATOR \leftarrow identity$
12: return $r'_i$
```

A. Algorithm

An AS_PATH a is said to be **represented by** $b (a \prec b)$ if and only if their aggregated path c is equal to b. Without loss of generality, we base the description of our algorithm on a router with identifier $identity$ (AS number + Router ID). For a route r_i regarding prefix d to be disseminated to one BGP neighbor N at time t_i, and a history set $H_{N,d}$ used to record the AS_PATHs recently propagated to N for prefix d, our goal is to compute its *aBGP route* r'_i. In addition to the algorithm presented in Algorithm 1, CAGG tracks per prefix and per neighbor the latest route it had sent, in order to filter our the identical route to the previous one.

The function agg aggregates $r_i.AS\_PATH$ and paths in history set $H_{N,d}$ together. The return value can be an aggregated path involved with $r_i.AS\_PATH$, or $r_i.AS\_PATH$ itself when the aggregation fails. Not all the paths in $H_{N,d}$ are selected to be aggregated together with $r.AS\_PATH$, and the detailed selections depend on the policy used in CAGG. For instance, we can always select the shortest $k(k \geq 1)$ paths in AS_PATH length, the top k paths most frequently used and so on. Another feature of function agg worthy to be mentioned is that the output path's AS_PATH length is assured to be shorter than any of the member paths involved in this aggregation. In fact, this is common for an aggregation, for example the aggregated path resulting from the three paths in Figure3 is $7\{2,3,4,5,6\}1$ with the length 3, while all the member paths have length 4. However, it may not hold in some extreme cases. For instance, the aggregated path of 765 and 7654 is 7654, which is longer than member path 765.

Comparing r'_i with r_i, the NEXT_HOP, MED, COMMUNITY, ATOMIC_AGGREGATE and even LOCAL_PREF attributes do never change in all cases, however, AGGREGATOR attribute is subject to a change if the r'_i is involved in aggregation. AGGREGATOR attribute is currently just informational and not used in routing selection, thus the impacts of its change are ignored. CAGG doesn't break the interaction pattern between routing and forwarding as in BGP that a router A will install the NEXT_HOP attribute attached to the received BGP route from B in its local RIB for data forwarding if that route wins in A's BGP decision process. The NEXT_HOP attribute of r'_i remains the same to r_i, so does the forwarding path.

Churn Aggregation approach differs from traditional routing aggregation(TAGG)[7] primarily in three aspects: (1) TAGG aggregates a group of routes destined for different prefixes, which are usually covered by a super prefix. CAGG however aggregates AS_PATHs regarding the same prefix; (2) TAGG essentially consists of two separated aggregations, NLRI and path attributes, including ORIGIN, NEXT_HOP, AS_PATH, ATOMIC_AGGREGATE and AGGREGATOR[2]. On the contrary, only AS_PATH aggregation is involved in CAGG; (3) the routes involved in TAGG are assured to be simultaneously available, but this is not the case in CAGG. For example, supposing that the latest path propagated from AS 7 in Figure 3 to AS 8 is the aggregated path $7\{2,3,4,5,6\}1$, and the failure of link $(1,2)$ invalidates paths 7421 and 7521. However, the propagated path remains unchanged since the only available path 7631 can still be represented by the aggregated path.

B. Safety, Performance and Convergence Issues

Given a sequence of *aBGP* routes, there are two major differences from BGP: AS_PATH, and the moments to disseminate some of them. Proving that *aBGP* is loop free and the priority of aBGP routes are no less than those in *BGP* are straightforward for two reasons: (1) all the ASes in $r.AS\_PATH$ are assured to be in the *aBGP* path $r'.AS\_PATH$; (2) the length of aggregated path is no longer than each of the paths involved in the aggregation.

In case if $r'.AS\_PATH \neq r.AS\_PATH$, a CAGG enabled AS M aggregates several history paths with $r.AS\_PATH$ together, and propagates only aggregated path $r'.AS\_PATH$ instead. Those ASes involved in any history path h_i only are exclusive to use $r'.AS\_PATH$ since their AS numbers have been included in the AS list of this route even when h_i has actually been withdrawn. In response to this risk, our solution is to *enable SSLD in CAGG*. SSLD(Sender Side Loop Detection)[9] is a technology used to avoid propagating routes to a peer which would detect a loop and discard it later, in order to partially reduce communication cost. In detail, SSLD checks if AS_PATH list in route to be propagated already has receiver AS inside, and discard this route in advance if so. For each direct neighbor N of a CAGG enabled router, the key idea is to perform the loop check on the sender side other than receiver side so that N's AS number does not appear in any paths exported to N, neither does the history cache $H_{N,x}$ where x is an arbitrary prefix exported to N from this router. Formal proofs are presented in[21].

C. Implementation

CAGG maintains per peer N and per prefix d a penalty value $P_{N,d}$ to measure the recent activities related to this prefix, just as that in classical RFD. $P_{N,d}$ is accumulated by a constant increment dependent of the type of the routing changes upon detecting a route for d to be propagated to peer N. CAGG also maintains a path frequency value $F_{N,d,p}$ for each path p in $H_{N,d}$, to record p's frequency of that selected as best path for prefix d. Both $P_{N,d}$ and $F_{N,d,p}$ decay exponentially according to the equation that $P_{N,d}(t') = P_{N,d}(t) \times e^{-\lambda_{N,d}(t'-t)}$ and $F_{N,d,p}(t') = F_{N,d,p}(t) \times e^{-\lambda_{N,d}(t'-t)}$ respectively. $\lambda_{N,d}$ is defined on per peer and per prefix basis, but we use a globally unique λ for simplicity in this paper defined together with Halftime H such that $e^{-\lambda \times H} = \frac{1}{2}$.

CAGG includes two types of garbage collection, *AS_PATH-level* and *Prefix-level* clean, to minimize the memory consumption. Once detected a route r for prefix d to be propagated to neighbor N, the $P_{N,d}$, $H_{N,d}$ and $F_{N,d,r.AS\_PATH}$ are first updated. Then *AS_PATH-level* clean removes from $H_{N,d}$ the paths whose prefix penalties have fallen beyond T_{sweep}. If $P_{N,d}$ exceeds the *Cutoff Threshold*, as in RFD, Algorithm 1 is triggered to compute the *aBGP* route r'; or else the *aBGP* route r' is directly set to be r. *Prefix-level clean* is scheduled as a background thread to clean the history of those prefixes whose *Prefix Penalties* have fallen below $T_{inactive}$. In our experiments, this clean process is invoked every 6 hours. More implementation details are presented in [21].

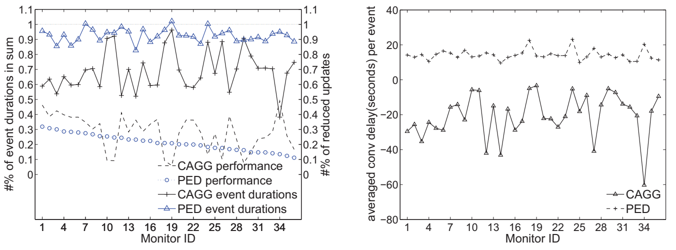
IV. Evaluation

Two algorithms, RFD and PED, are taken as reference in the latitudinal comparison with our CAGG approach. This comparison consists of three parts: CAGG is compared with 1) PED and 2) RFD respectively in both performance and convergence; 3) the evaluation of the memory cost across 36 monitors.

A. Experimental Setting

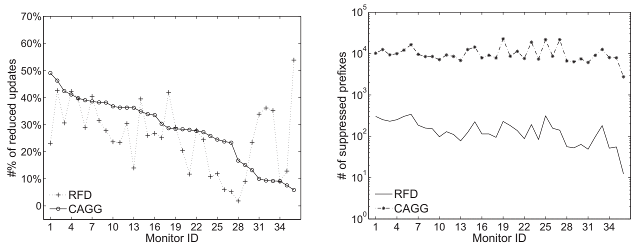
In our experimental setting, RFD is configured with Cisco default parameters, and PED with PEDI[1] 35 seconds, which

[2]Routes that have different MED attributes are not allowed to aggregate together[7].



(a) Event durations and performance (b) Relative convergence delay

Fig. 4. Compare CAGG with PED. The monitors are sorted by PED performance in reducing BGP updates in descendant order.



(a) Reduced updates (b) # of suppressed prefixes

Fig. 5. Compare CAGG with RFD. The monitors are sorted by CAGG performance in descendant order.

has been shown by Geoff Huston et al. to perform better than 30 seconds MRAI, a typical setting for current eBGP session. We use the BGP data from route-views2.oregon-ix.net ranging from 00:00, 1st, Dec, 2009 to 23:45, 31st, Dec, 2009. This collector is supposed to be peering with 46 monitors, however, we find only 42 active monitors in 38 ASes, and 36 monitors are selected in our experiments since the other 6 are not always available in the entire month. As for CAGG, the Halftime(H) we use is 9,000 seconds, which balances the performance with memory cost[21].

B. Comparison between PED and CAGG

PED is supposed to depress the transient updates during path exploration, no matter the involved prefix is stable or unstable in history, CAGG however pays attention to only recently active prefixes. As shown in Figure 4(a), CAGG outperforms PED in reducing BGP updates in 80.5% of the monitors (29 out of 36), and the averaged ratio of reduced BGP updates by CAGG across the 36 monitors is 28.1% while 21.2% by PED. As for the total event durations, two conclusions can be made from Figure 4(a): (1) both PED and CAGG are capable of shortening path exploration durations, however, PED may sometimes prolongs the total durations, as in Monitor 7, 19 and 24; (2) there is no correlation for PED between the number of reduced BGP updates and its total event duration. On the contrary, a clear positive correlation can be found in CAGG that more updates are reduced, shorter event duration we can achieve. The results are really encouraging that as much as 60% path exploration duration(with Monitor ID 34) can be eliminated, while the average reduction is 32% by CAGG and 7.3% by PED.

The averaged relative convergence delay per event of the 36 monitors is shown in Figure 4(b), where CAGG significantly outperforms PED that CAGG always accelerates BGP convergence while PED always delays the convergence. For each event u, and its corresponding CAGG output u_c and PED output u_p, the relative convergence delay of CAGG is defined to be the time distance between the last update in u and u_c that $conv(u_c) = L(u_c) - L(u)$ where $L(x)$ denotes the time to disseminate the last update in event x. Similarly, $conv(u_p) = L(u_p) - L(u)$. In opposite to a positive $conv$, which delays convergence, a negative $conv$ will accelerate the convergence process. As we can see in Figure 4(b), PED delays BGP convergence 14.2 seconds per event on average across the 36 monitors while the convergence in CAGG is 20.7 seconds faster than in BGP per event. Furthermore, the relative convergence delay curve of CAGG coincides well with the event duration curve in Figure 4(a).

C. Comparison between RFD and CAGG

Both RFD and CAGG are history-based damping mechanisms used to stabilize Internet routing, and both of them are supposed to deal with highly active prefixes. The comparison includes three parts: (1) how about their performances in reducing BGP updates, especially for the top 10,000 prefixes; (2) how many prefixes are suppressed by these two mechanisms respectively; and (3) how about the routing reachability and optimality to those suppressed prefixes in RFD and CAGG.

As for the first question, CAGG outperforms RFD in 68% of the monitors (24 out of 36), as in Figure 5(a). Similar results are observed on the top 10,000 prefixes that CAGG performs better in the top 10,000 prefixes than RFD in 55.6% (20 out of 36) of the monitors. Compared with the performance across overall prefixes, CAGG's advantage against RFD is not so significant. That's because RFD arbitrarily depresses all the updates for a prefix whose penalty value has exceeded a given threshold, thus more efficient in reducing updates for those highly active prefixes, however, CAGG still outperforms RFD across overall prefixes.

The averaged number of suppressed prefixes in RFD and CAGG across all the 36 monitors are shown in Figure 5(b). As we can see, CAGG suppresses significantly more prefixes than RFD in all the given monitors. RFD steadily suppresses around 200 prefixes across all the monitors, and the number decreases a little as the performance degrades. Although RFD suppresses fewer prefixes, we can find in the case study[21] that RFD can sometimes suppress as many as 3,200 prefixes, and the caused reachability loss can not be ignored.

D. Memory cost

CAGG buffers the AS_PATHs for those highly active prefixes, thus incurs extra memory overhead. The memory cost of each monitor(number of buffered AS_PATHs) is sampled every 2 hours, and then compared with the AS_PATHs in corresponding RIB. Only the averages are presented in Figure 6. There are 45,368 unique AS_PATHs in RIB on average

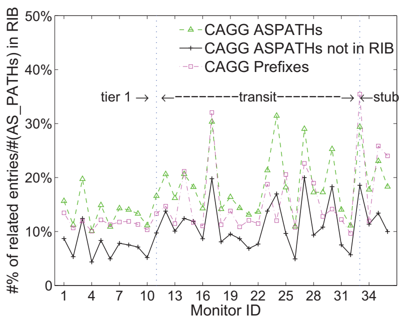


Fig. 6. Memory cost related to CAGG. Inside each tier, the monitors are sorted in a descend order according their performances in reducing updates.

over one month across the 36 monitors, and the averaged AS_PATHs in CAGG history cache accounts for 17.6% of the RIB path size, while the unique paths cached by CAGG after removing those already in RIB falls to 8.9% of the number of AS_PATHs in RIB. What we want to emphasize is that this is the upper bound of memory cost of CAGG: (1) if attribute sharing is extended to the AS_PATHs stored in *Adj-RIB-Ins*[7](They are also learned from BGP neighbors but not selected as best paths, thus invisible in local-RIB), the memory cost will be further reduced; (2) we maintained history paths for all the prefixes observed from each monitor in the evaluation, however, only a subset of those prefixes are exported to BGP peers. Correspondingly, the memory cost is overestimated here. The "CAGG Prefixes" curve in Figure6 indicates the number of prefixes CAGG maintains history paths for. Interestingly, these prefixes are even more than buffered AS_PATHs. It means that there are many groups of prefixes alway sharing the same AS_PATH.

The memory cost on average in high tier ASes are lower than in low tier ASes. The averaged number of CAGG AS_PATHs those not in RIB is 3,373 in the 11 Tier 1 monitors, accounting for 7.41% of AS_PATHs in RIB, while the numbers and fractions are 5,275, 10.5% and 5,420 and 11% respectively in transit and stub ASes.

V. Conclusion

This paper proposes a novel countermeasure against routing instabilities without harming BGP convergence, on the basis of the observation that a highly active prefix explores only a small number of AS_PATHs. CAGG converts the multiple paths frequently explored by a prefix into an aggregated path, to reduce the resulted updates from AS_PATH change. This handling is much safer than arbitrarily suppressing updates related to a highly active prefix in RFD, so more aggressive damping parameter can be used. Our experiments with real BGP data show that CAGG can significantly reduce BGP updates and path exploration duration, 28.1% and 32% respectively on average, and 50% and 60% respectively in the best case. Latitudinal comparison with RFD and PED also proves our efficiency in depressing routing instabilities. Currently, CAGG works over only eBGP sessions thus cannot protect its iBGP peers from excessive updates flooding. It is possible to introduce CAGG to iBGP as well after their interactions are exhaustively studied, and this is part of our future work.

VI. Acknowledgement

This work is primarily finished during the visit in UCL, Belgium. This work has been partially supported by NSF of China under Contract No.60873214. Many thanks to Pierre Francois, Pei-Chun Cheng and anonymous reviewers for their valuable comments on this work.

References

[1] G. Huston, M. Rossi, and G. Armitage, "A Technique for Reducing BGP Update Announcements through Path Exploration Damping," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 8, 2010.
[2] A. Elmokashfi, A. Kvalbein, and C. Dovrolis, "On the scalability of BGP: the roles of topology growth and update rate-limiting," in *Proceedings of the 2008 ACM CoNEXT Conference*. ACM, 2008, pp. 1–12.
[3] J. Luo, J. Xie, R. Hao, and X. Li, "An approach to accelerate convergence for path vector protocol," *Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE*, pp. 2390–2394, 2002.
[4] D. Pei, M. Azuma, D. Massey, and L. Zhang, "Bgp-rcn: improving bgp convergence through root cause notification," *Comput. Netw. ISDN Syst.*, vol. 48, no. 2, pp. 175–194, 205.
[5] J. Chandrashekar, Z. Duan, Z.-L. Zhang, and J. Krasky, "Limiting path exploration in bgp," in *INFOCOM*. IEEE, 2005, pp. 2337–2348.
[6] C. Villamizar, R. Chandra, and R. Govindan, "Bgp route flap damping," http://www.ietf.org/rfc/rfc2439.txt.
[7] Y. Rekhter, T. Li, and S. Hares, "A border gateway protocol 4 (bgp-4), rfc 1771 & rfc 4271," Internet Official Protocol Standards, 1995,2006.
[8] A. Bremler-Barr, Y. Afek, and S. Schwarz, "Improved BGP convergence via ghost flushing," *IEEE INFOCOM 2003*, vol. 00, pp. 927–937, 2003.
[9] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian, "Delayed Internet routing convergence," *IEEE/ACM transactions on networking*, vol. 9, no. 3, pp. 293–306, 2001.
[10] D. Pei, X. Zhao, L. Wang, D. Massey, A. Mankin, S. F. Wu, and L. Zhang, "Improving bgp convergence through consistency assertions," in *In Proceedings of the IEEE INFOCOM*, 2002.
[11] W. Sun, Z. Mao, and K. Shin, "Differentiated BGP Update Processing for Improved Routing Convergence," *Proceedings of the 2006 IEEE International Conference on Network Protocols*, no. i, pp. 280–289, Nov. 2006.
[12] F. Wang and L. Gao, "Path Diversity Aware Interdomain Routing," *IEEE INFOCOM 2009 - The 28th Conference on Computer Communications*, pp. 307–315, Apr. 2009.
[13] P. chun Cheng, J. H. Park, K. Patel, and L. Zhang, "Flap damping with assured reachability," UCLA CS Technical Report 100024.
[14] C. Pelsser, O. Maennel, K. Patel, and R. Bush, "Route flap damping considered useable," IEPG,Beijing,http://www.iepg.org/2010-11-ietf79/101107.iepg-rfd.pdf.
[15] Z. Mao, R. Govindan, G. Varghese, and R. Katz, "Route flap damping exacerbates Internet routing convergence," in *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM, 2002, p. 233.
[16] R. R. W. Group, "Recommendations on route-flap damping," RIPEDocumentID378,http://www.ripe.net/docs/ripe-378.html.
[17] T. Griffin and B. Premore, "An experimental analysis of BGP convergence time," in *Proceedings of ICNP*. Citeseer, 2001, pp. 53–61.
[18] P. Jakma, "Revised default values for the bgp minimum route advertisement interval," http://tools.ietf.org/html/draft-ietf-idr-mrai-dep-02.
[19] R. V. Oliveira, R. Izhak-ratzin, B. Zhang, and L. Zhang, "Measurement of highly active prefixes in bgp," *IEEE GLOBECOM*, vol. 2005, 2005.
[20] T. Griffin, "Analysis of the MED oscillation problem in BGP," in *IEEE International Conference on Network Protocols (ICNP)*, 2002.
[21] W. Xiaoqiang and O. Bonaventure, "Stabilizing bgp routing without harming convergence," *Technical Report, http://u.sohu.com/download/10/1294122421724068867611 6*, 2010.