# Improved BGP Convergence via Ghost Flushing

Anat Bremler-Barr Yehuda Afek Shemer Schwarz
Tel-Aviv University
{natali,afek}@math.tau.ac.il,shemers@hotmail.com

***Abstract*— In [1], [2] it was noticed that sometimes it takes BGP a substantial amount of time and messages to converge and stabilize following the failure of some node in the Internet. In this paper we suggest a minor modification to BGP that eliminates the problem pointed out and substantially reduces the convergence time and communication complexity of BGP. Roughly speaking, our modification ensures that bad news (the failure of a node/edge) propagate fast, while good news (the establishment of a new path to a destination) propagate somewhat slower. This is achieved in BGP by allowing withdrawal messages to propagate with no delay as fast as the network forwards them, while announcements propagate as they do in BGP with a delay at each node of one $minRouteAdver$ (except for the first wave of announcements).**

## I. Introduction

### A. The BGP Convergence Problem

A strong relationship between the topological structure of the internet and the time it takes BGP to converge following the detachment of a subnet has been shown in two recent papers [1], [2]. In [1] the authors go as far as making the following recommendation:

> "Our results show that customers sensitive to fail-over (should read fail-down, BAS) latency should multi-home to larger providers, and that smaller providers should limit their number of transit and backup transit interconnections."

In [2] the authors claim they "can certainly improve BGP convergence through the addition of synchronization, diffusing updates, and additional state information, but all of these changes to BGP come at the expense of a more complex protocol and increased router overhead." In this work we further analyze the problem, and suggest a minor modification to the code of BGP that significantly improves the convergence latency, without any modification to BGP's messages format or any other part of BGP. Thus eliminating the negative effects that the desired extra transit and backup transit connections have on today's BGP routing protocol (as per the quote above).

The reduction in the convergence latency of BGP plays a major role in providing QoS and highly available services on the Internet. As shown in [1], [2] current BGP behavior results in fail-down latency of 3 to 15 minutes. Where fail-down is the failure and detachment of a destination from the network (i.e., failure without an alternative path to the detached router/network), while fail-over is when the failure introduces a new longer (backup) path. In either case our modifications reduce the convergence latency to about 10 to 15 seconds (on the same scenarios that were analyzed in [1], [2]).

### B. Related Work

An important parameter in the convergence time of BGP is $minRouterAdver$ timer. Basically it is the amount of time BGP enforces between the sending of consecutive announcements from a router to its neighbors (currently set to 30 seconds). In [2] it is proved that the fail-down convergence time is $n \cdot minRouteAdver$, where $n$ is the longest simple path to a destination ($n$ the number of nodes in the network in the worse case). However, in [2] it is also shown that without $minRouteAdver$ timer each router may explore all possible simple routes to a destination and hence may send $n!$ messages. Moreover, [3] shows that in this case we also may get unacceptably long convergence time. Simulation done by [3] shows that for each specific network topology there is an optimal value of $minRouteAdver$ that minimizes the converge time. However, since this value varies from network to network the technique cannot be a general mechanism to improve BGP. In this paper, we show that by slightly modifying BGP rules, of when to send withdrawals and announcements, we benefit from the $minRouteAdver$ timer and improve the convergence time in a general network, and even decrease its message complexity.

While researchers started to analyze and study the BGP convergence problem only recently [1], [2], [3], [4], [5], [6], [7], [8], the basic problem observed is not new. It is a variant of the known "counting to infinity" problem, that occurs in distance vector routing protocols [9], [10] in a different disguise (of course, in BGP the counting is limited since BGP is using the ASpath to avoid loops). There are known techniques to overcome this problem that add state information to the BGP messages [11],

[12]. Introducing these changes into BGP would make it a more expensive and complex protocol.

There are other solutions that do not require state information [13], [14], such as, Reverse Poisoning (used in RIP [13]), Route Poisoning with Hold down timers and trigger update (that are used in Cisco's IGRP [14]). The RIP technique, *reverse poisoning*, is not relevant to BGP since it is designed to break length two loops and the ASpath easily achieves this and much more.

The IGRP technique *Route poisoning with hold-down-timers*, on the other hand, could be employed in BGP but would have devastating effects on its performances. Mostly because it is a non-scalable solution which is good for limited size networks. Essentially it first cleans the entire network from the old routes and after waiting long enough time to guarantee that the old route does not exist any more in the network, it starts computing a new route. Waiting for such a long time in BGP is prohibitive.

A new solution to reduce the convergence time complexity was recently introduced in [8]. It uses the information provided in the ASpath to define route consistency assertion and uses these assertion to identify infeasible routes. However, this technique requires extra computation resources from the router to compute the consistency check, and to send extra information in the BGP messages. This technique may run into difficulties in some pathological cases, when an AS partitions - and some router in the $AS$ becomes disconnected from other routers in the same AS.

### C. Ghost flushing Solution

In this paper we take a somewhat different view on the problem of convergence latency in BGP. Abstractly the problem is that one lie makes many and in computer networks it continues recursively. That is, following the failure of a destination or some links to a destination, there are pieces of incorrect information (lies) floating in the network for a relatively long period of time. These pieces of information are reminiscent of the paths to a destination that was detached from the network, hence called *ghost information*. To make things worse, some routers rely on the false information to generate more false information. In this way, convoys of false information travel in the network until they disappear. Throughout this period of time there are routers in the network with the wrong information on the route to the destination. Notice that the ghost information disturbs the convergence both in the case of fail-over and fail-down.

Two basic but simple modifications of BGP are suggested in this paper *ghost flushing rule*, and *ghost buster rule*, (a third suggestion, *reset rule* is described in the full paper). The simplest and the one that makes the most difference is the *ghost flushing rule* presented in Section V, in which extra withdrawal messages are injected in order to flush the ghost information from the network. Essentially, under the ghost flushing rule a router sends a withdrawal of a prefix to its neighbors as soon as it learns (with no delay) that the last AS path it has announced for that prefix has been changed and became longer or not valid. The withdrawals generated by the flushing rule inform the neighbor router that the previously announced ASpath is no longer valid. This solution reduces the convergence latency to $d \cdot h$ (from $n \cdot 30$ in current BGP) , where $d$ is the length of the longest ASpath that a router in the network has to the destination, before the failure of that destination ($< 20$)and $h$ is the average delay between two neighboring BGP routers. Effectively reducing the fail-down convergence latency from several minutes to about $10 - 30$ seconds on the same scenarios that were analyzed in [1], [2].

While the minor modification suggested here considerably improves the fail-down convergence latency, it usually also improves the fail-over convergence complexity. This is because in many cases the ghost information also disturbs and confuses the routers with outdated information.

Given BGP with the ghost flushing rule we make an additional observation on the resulting algorithm and suggest another rule called the ghost busting rule presented in Section VI. The observation is, that if the ratio between the time it takes an announcement message to traverse one hop, to the time it takes a withdrawal message, is $k = \frac{\delta + h}{h}$, where $\delta$ is the time by which the announcement is delayed at each node, then the convergence time of the protocol is $\frac{khd}{k-1}$ where $d$ is the diameter of the network. The difference between the ghost flushing rule and the ghost busting rule is that in the former announcement messages may initially not be delayed by the $minRouteAdver$ (as is in existing BGP implementations) and in the later we ensure that any announcement whether after a long quiescent period or not, is delayed by a $minRouteAdver$ delay before being forwarded. I.e., the busting rule guarantees that announcements are always delayed.

Table I summarizes the convergence time complexity and message complexity following a fail-down event (i.e., the upper bound on the total number of messages sent due to the event) for BGP with $minRouteAdver = 0$ with $minRouteAdver = 30$ and with the two modifications suggested in this paper, the Ghost flushing rule, and the Ghost buster rule.

The results presented are supported by simulation in

which the convergence time of the original BGP and of the modified BGP are measured and compared in several different settings. These results which are presented in Section VII support our analysis that the modifications suggested in this paper considerably improve BGP's convergence complexity.

| | fail-down converg. time | fail-down messages |
|---|---|---|
| BGP w/ minRouteAdver =0) | $h \cdot n$ | $n!E$ |
| BGP w/ minRouteAdver = 30 | $30 \cdot n$ | $nE$ |
| Ghost flushing rule | $h \cdot n$ | $\frac{2Ehn}{30}$ |
| with Ghost buster rule | $\frac{kdh}{k-1}$ | $\frac{2Ekdh}{30(k-1)}$ |

TABLE I

CONVERGENCE TIME AND MESSAGE COMPLEXITIES IN THE FAIL-DOWN CASE OF THE DIFFERENT METHODS. WHERE $h$ IS THE AVERAGE DELAY BETWEEN TWO NEIGHBORING BGP ROUTERS, $n$ IS THE LONGEST SIMPLE PATH OF AS'S WHICH IS BOUNDED BY THE NUMBER OF AS'S IN THE NETWORK, $d$ IS THE LONGEST AS PATH THAT A ROUTER HAS TO THE AFFECTED DESTINATION, $E$ IS THE NUMBER OF BGP SESSIONS BETWEEN THE ROUTERS, AND $k$ IS THE RATIO BETWEEN THE TIME IT TAKES AN ANNOUNCEMENT MESSAGE TO TRAVERSE ONE HOP TO THE TIME IT TAKES A WITHDRAWAL MESSAGE ($k = \frac{\delta+h}{h}$, WHERE $\delta$ IS THE TIME BY WHICH THE ANNOUNCEMENT IS DELAYED AT EACH NODE, I.E., $\delta \simeq minRouteAdver$).

## II. BGP short overview

BGP is a distance and path vector routing protocol. Meaning that with each destination (prefix) in the routing table an $ASpath$ is associated, and the corresponding $ASpath$ is sent with each update sent on this destination to the neighboring peers. The $ASpath$ is the sequence of ASes along the preferred path from the router to the destination. For each destination a router records the last announcement (with the $ASpath$) it has received from each of its peers (neighboring BGP routers). Then, for each destination the router chooses one of the peers as the next-hop on the preferred path to that destination. Usually the router picks the peer that announced the shortest $ASpath$, however BGP is much more sophisticated and enables a more complex path selection according to policy.

The main motivation and usage of the $ASpath$ is in avoiding cycles in the routing protocol. This is achieved by each router simply invalidating any route that includes the router's own AS number in the $ASpath$. Some mechanisms to avoid route oscillations (which is not the problem addressed here) were introduced in [4], [5].

BGP is an event driven (incremental) protocol where a router sends an update to its peers only when its preferred $ASpath$ to a destination has changed. There are two types of messages exchanged between peering BGP routers: *announcements* and *withdrawals*. A router sends an announcement when its preferred $ASpath$ to a destination has been changed or when it has a route to a new destination. Withdrawal messages are sent when a router learns that a subnetwork (i.e., destination) is no more reachable through any of its interfaces. To avoid avalanches of messages and to limit the rate at which routers have to process updates, it is required in BGP that after sending an announcement for a destination a router waits a minimum amount of time before sending an announcement again for the same destination, or to any destination (it is recommended by the IETF to set this delay, called $minRouteAdver$ to 30 seconds [15]). However the delivery of a withdrawal message is never delayed because BGP tries to avoid "black holes", in which messages are sent to a destination which is no longer reachable. See Figure 2 for a high level pseudo code of BGP.

In this paper we consider four types of events that may occur in a BGP at a router. The events may be either due to a change in the Internet topology (failure or recovery of either a router or a link) or due to a change in a routing policy. The way BGP with our modification handles the events is independent of whether the event is due to a topological change or routing policy change:

- $E_{up}$ - A previously unavailable destination is announced as available at a router.
- $E_{down}$ or $fail - down$ - A previously available destination is announced as unavailable at a router.
- $E_{shorter}$ - A preferred $ASpath$ to a destination implicitly replaces a less preferred $ASpath$ (e.g., the path is becoming shorter).
- $E_{longer}$ or $fail - over$ - The $ASpath$ to a destination is replaced by a worse (longer) one. This happens for example, if the preferred route fails.

## III. BGP Model

We define the network graph as a bi-directonal Graph $G(V, E)$, where the set $V$ of $n$ nodes corresponds to the different AS's, and the set $E$ represents BGP sessions between AS's. Following [1], [4] and for the sake of simplicity, we associate one router with each $AS$ and one router with each destination. However, as it was shown in [8], one router cannot be associated with all the routers in the $AS$, since due to traffic engineering characteristics of BGP, different routers in the same $AS$ may announce different routes to the same destination. Similar to [8]

one can overcome this problem by introducing virtual AS's. A new virtual AS is introduced whenever there is a maximal proper subset (i.e., subset not equal) of routers of a previously defined AS such that all the routers in that subset route along the same path to a destination. The correctness of our modifications and their analysis relies on the property of a valid $ASpath$ in convergence time, as was defined in the model of SPVP (Simple Path Vector Protocol) [8] which is given below (Definition 1). Notice that our algorithms and analysis do not depend on (i.e., may be more general) the full model that was introduced in [7] (e.g., our work does not rely on the consistency between different AS paths present in a routing table in the same router). Moreover, while [8] captures the real-time model of the BGP, it ignores the impact of the $minRouteAdver$ timer which is mandatory for our work.

Unless it says otherwise all the discussions in this paper are with respect to destination $dst$. Let $ASpath^r$ be the last $ASpath$ to $dst$ announced by router $r$.

*Definition 1:* $ASpath^r = \{AS_r = AS_{r_0}, AS_{r_1}, .., AS_{r_m} = AS_{dst}\}$ is a **valid path** in convergence time, if there exist a simple path $\{AS_r = AS_{r_0}, AS_{r_1}, .., AS_{r_m} = AS_{dst}\}$ in the network graph. Then for every $i$, $0 \leq i < m$ $ASpath^{r_i} = AS_{r_i}, AS_{r_{i+1}}, .., AS_{r_m}$.

Notice, that we make no assumption on the way a router chooses its preferred $ASpath$. While, the default in BGP is to choose the shortest $ASpath$ a router may override this default and choose its preferred $ASpath$ according to any complex policy defined in the router or some matric supported by BGP. We define $d$ the **network diameter** as the longest simple preferred $ASpath$ before the fail-down event.

## IV. Ghost information

The main issue of this paper is how to deal with outdated pieces of information floating in the network. More specifically how to distinguish between correct and incorrect pieces of information, which we call *ghost information*.

Let us follow the example given in [1], [2] described slightly differently, exposing what we believe is the essence of the problem.

For ease of explanation only, we give the scenario in a synchronous network, where at each round the router receives the messages sent in the previous round, calculates its new state and sends new messages to its peers if required. The duration of each round is $h$ seconds which we assume, for simplicity, as 1 second. For the ease of the explanation assume for this example that if a router has to update its $ASpath$ to one of a few equal length $ASpath$s then it chooses the $ASpath$ that begins with the smallest $AS$ number.

Consider the topology shown in Figure 1(a) in which $AS$ 0 is connected to all the four AS's which are connected in a clique. Let all the $AS's$ in the clique route to $dst$ through $AS$ 0. Consider the case where $AS$ 0 withdraws its route to $dst$ since network $dst$ becomes unreachable, and hence $dst$ should be withdrawn from all the router's routing tables. However, at the time $dst$ becomes unreachable the false information is still in the network. Moreover, as was explained above, nodes start to use and rely on the false information and thus the ghost information start to travel around the network building longer and longer $ASpath$s until they include a cycle.

Formally we define:

*Definition 2:* $ASpath^r$ is a *ghost information* if it is not a valid AS path held or stored by one of the routers during the convergence period.

The *ghost information* may be also in the $ASpath$ that a router stores as the last received $ASpath$ from its neighbor. Formally, we denote by $ASpath_p^r$ the last $ASpath$ to $dst$ that $r$ received from router $p$. This may be different from the actual $ASpath$ of $p$ ($ASpath^p$), since $p$ may send another announcement with a new $Aspath$ which $r$ has not yet received. Formally we define,

*Definition 3:* $ASpath_p^r$ is a *ghost information* if this $ASpath$ is not a valid path in router $p$ at convergence time, or there is some message M in transmit between $r$ to $p$ with an $ASpath$ different than $ASpath_p^r$.

Coming back to the example, $AS$ 4 after receiving a withdrawal from AS 0 chooses $ASpath = 10$ according to the *ghost information* , the last announcement ($ASpath = 10$) it received from $AS$ 1, and would send an announcement saying that any router that was used to reach $dst$ through it should use the path 410. Notice how the ghost information, the existence of a path through $AS$ 1 to $dst$, traverses in the network. In time $t = 2$, it is responsible for the ghost $ASpath$ 10 at $AS$ 2, $AS$ 3 and $AS$ 4. This ghost information then traverses and becomes at time $t = 3$ the ghost $ASpath$ 210 at $AS$ 3 and $AS$ 4 and the ghost $ASpath$ 310 at $AS$ 2.

In the next round ((c) $t = 2$), AS 4 learns that if it chooses to route through $AS$ 1 the new path should be 120, since $AS$ 1 changes its path to 20. However, AS 4 cannot update its peer about the change in its $ASpath$ until at least 30 seconds have passed from the previous announcement sent. *Here, we see the negative effect of $minRouteAdver$ that delays the elimination of false (ghost) information*. In this round also AS 1 learns that all the other $ASes$ are routing though it - and hence it would change the $ASpath$ to $\{\}$ and send

immediate withdrawal to all of its peers. As a result of this, in $t = 3$ AS 4 changes its $ASpath$ to $= 210$. *Notice that because of the* $minRouteAdver$ *rule, AS* 2 *does not learn that now all the ASes route through it until* $t = 31$*, where all the ASes would send the next announcement.* This delayed convergence yields a 62 seconds (rounds) convergence. As noted in [1] in most cases the architecture is more complex and the average convergence latency is 3 minutes and more.

In [2] a detailed scenario is given, showing that the convergence time is $(n-2) \cdot minRouteAdver$ seconds with message complexity $nE$, where $n$ is the number of nodes (AS's), and $E$ is the number of links.

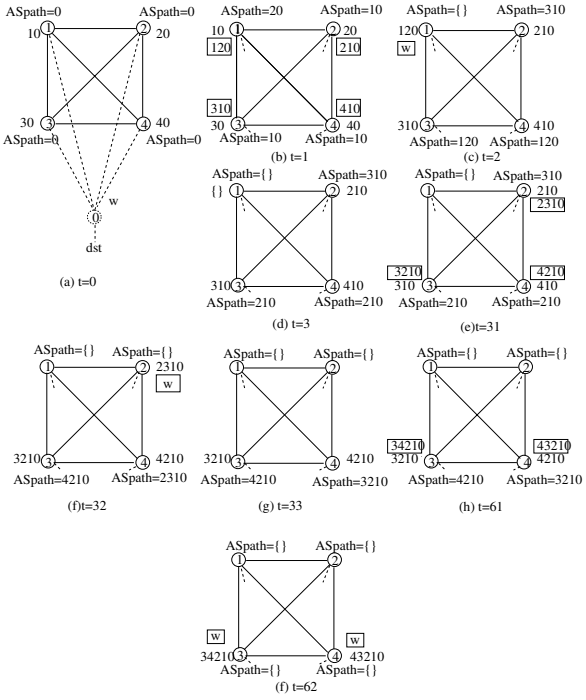


Fig. 1. Illustration of the ghost information problem in a clique. Next to each node we write its current $ASpath$ (with $ASpath = ..$ above or below the node). This is the path to the destination $dst$ that appears in this node routing table in the corresponding snapshot. Next to the node, we depict the $ASpath$ that the node has sent in the last announcement, preceding this round, which is the $ASpath$ its peers believe this router has. The $ASpath$ of the current announcement message (if exists) in a frame. The letter $w$ stands for a withdrawal message.

Observing the above scenario, one may conclude that the large (30 seconds) $minRouteAdver$ is the source of the problem and that by drastically reducing it the problem would be solved. However, as shown in [1], without this delay the message complexity of the convergence process jumps to $O(n!E)$ and as mentioned before the load on the routers would drastically increase. Since without the $minRouteAdver$ timer that delays the propagation of announcements every node may explore any possible combination of $ASpath$ value until converging to the stable shortest paths'.

## V. Ghost Flushing rule

In order to quickly flush the ghost information from the network one should update, as fast as possible its neighbor whenever the previous $ASpath$ it was advertising is not valid any more. This is done by the following modification:

```
When the distance to destination dst
 is updated to a worse ASpath
 AND
    a minRouteAdver time did not
    elapse since the last announcement
then
    send withdrawal(dst) to
    all neighboring BGP peers
```

Notice that if more than $minRouteAdver$ has passed since the last announcement sent then it will send an $ASpath$ rather than a withdrawal.

In figure 2 lines 16a-16d are the only modification (addition) to the traditional BGP pseudo-code. The above modification uses withdrawal as a mechanism to flush the ghost information. Unlike in the traditional BGP, where the withdrawal messages are used to indicate that there is absolutely no path to the destination, here the withdrawal messages are used to indicate that the previously sent $ASpath$ is not valid anymore.

Moreover, here the router that sends the withdrawal might have a route to the destination, and it routes packets to destination according to its new $ASpath$. We need to use the withdrawal message only in the cases where the $minRouteAdver$ rule prevents us from sending the new $ASpath$. I.e., the withdrawal message plays the role of telling the neighbor router that the last $ASpath$ announced to him, is irrelevant (not a valid path).

Notice, that the router cannot announce the new $ASpath$ to its neighbor, since this solution led to $O(n!E)$ message complexity. We call this message a *flush withdrawal*, to indicate its special functionality, however it is implemented by a regular withdrawal message. Essentially, this would result in a wave of withdrawal messages flushing a ghost $ASpath$. Any node that is the source of the ghost information sends in this process a withdrawal message and the nodes depending on that information also erase the erroneous information and forward the wave of withdrawal messages. Thus, any $ASpath$ in the network that depends on a ghost information is thus flushed as fast as possible. The flushing progresses quickly along the paths because the $minRoutAdvr$ does not apply to the withdrawal messages. Let $h$ be

a bound on the time it takes a BGP message (announcement or withdrawal) to traverse between neighboring BGP routers, including the processing time, then the time complexity is reduced to $nh$ (Lemma 5.1) from $minRouteAdver \cdot n$ and the total message complexity is reduced to $\frac{2nhE}{minRouteAdver}$ from $nE$ (Lemma 5.2). Notice that $h \ll minRouteAdver$, while $minRouteAdver$ is 30 seconds $h$ is $1-2$ seconds. Hence we reduce the message complexity and the time complexity by at least a factor of 15.

Notice, that it is enough that the withdrawal message is sent only if the $ASpath$ is changed to a less preferred one, since ghost information may harm convergence, only if the ghost information is preferred over the $ASpath$ into which the process converges.

We prove this results in the following lemmas. All the lemmas and definitions are with respect to a destination $dst$ not specifically mentioned:

*Lemma 5.1:* The time it takes BGP with the flushing rule to converge following an $E_{down}$ is $n \cdot h$.

Proof: Let node $dst$ become unreachable to all its neighbors due to a failure. This lemma follows from claim 1 below according to the following argument: We prove by induction that $kh$ seconds after $E_{down}$ the $ASpath$ to destination $dst$ of any node in the network is longer than $k$. Hence after $n$ units of time all the nodes would withdraw their route (because the maximum valid $ASpath$ in BGP is of length $n$, due to BGP loop detection mechanism). ∎

*Definition 4:* We define $|ASpath|$ as the length of the $ASpath$.

*Claim 1:* At time $kh$ every message or node has an $|ASpath| > k$.

Proof by induction. Let the node $dst$ become unreachable due to the failure. The basis of the induction: consider the nodes that are at distance 1 from the $dst$. After the failure, at time $h$ - they learn that it is impossible to reach $dst$ directly, hence the new $|ASpath| > 1$.

Inductive step: consider the $ASpath$ of the nodes at time $(k+1)h$. From the induction hypothesis at time $kh$ the $ASpath$ of any nodes in the network is longer than $k$. Hence also at time $(k+1)h$ the $ASpath$ of nodes in the network is longer than $k$, since the $ASpath$ in the network can only grow in length (if the destination does not become connected again during that time). We now prove that there is no node $v$ with $|ASpath| = k+1$ at time $(k+1)h$. Assume to the contrary: there exists node $v$ with $|ASpath| = k+1$. Let us look at time $t$ where node $v$ changes to this $ASpath$, since it received an announcement $m$ from some peer $p$ where $|ASpath(m)| = k$. Since we prove that from time $kh$ there is no node with $ASpath$ shorter or equal to $k$, there must be a time before $kh$ where the $ASpath$ of $p$ was change to less preferred $|ASpath| > k$. By our modification at this time $p$ sends a withdrawal to its peers or a new $|ASPath| > k+1$ if it didn't send an announcement in the last $minRouteAdver$. The withdrawal message is received at $v$ before time $(k+1)h$ and hence we arrive at a contradiction to the assumption that $v$ has an $ASpath$ with length equal to $k+1$. ∎

*Lemma 5.2:* At each $minRouteAdver$ time interval following a $E_{down}$ in BGP with the flushing rule a router may send at most two messages .

Sketch of proof: According to the algorithm at least $minRouteAdver$ must elapsed between one announcement to the next. Here we claim that at most one withdrawal message is sent by any node between two consecutive announcement messages. This is because in Line $16d$ in Figure 2 $LastAnnounceASpath$ is set to an empty set, and this value prohibits any other withdrawal messages ( Line $16a$ in Figure 2) because no message can generate a path longer than the empty path. Hence in $minRouteAdver$ after the first announcement, there is a maximum of two messages - one announcement and one withdrawal. ∎

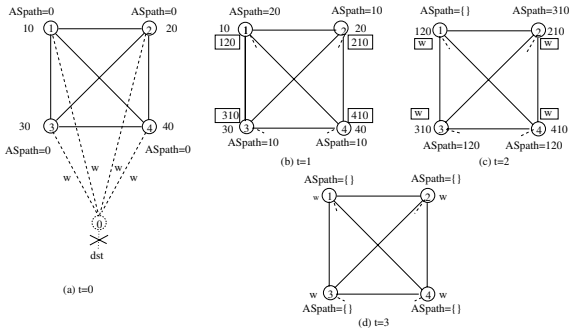


Fig. 3. Depicting the same situation as in Figure 1. Clearly the scenario is much shorter in time since the flushing technique is used.

*Lemma 5.3:* The convergence message complexity of BGP with the flushing rule following $E_{down}$ is $O(\frac{2hnE}{minRouteAdver})$

Straightforward from lemma 5.2 and 5.1. ∎

Figure 3 describes a detailed scenario of the same situation as in Figure 1. Clearly the scenario is much shorter in time since the flushing technique is used. Notice that the time is reduced from $t = 62$ to $t = 4$ seconds.

## A. Fail-over

So far we discussed the effect of the ghost flushing rule on the convergence complexity following the disconnec-

Upon receiving message ($type$,$PeerASpath$,$dst$) from peer $p$ in router $r$ in $AS_r$

1 If ($type == Withdrawal$)
2 $\quad ASpath_{dst_p} = \{\}$
3 If ($type == Announcement$)
4 $\quad$ If $AS_r \in PeerASpath_{dst}$ {The loop detection/prevention mechanism }
5 $\quad\quad ASpath_{dst_p} = \{\}$
6 $\quad\quad$ Else
7 $\quad\quad ASpath_{dst_p} = PeerASpath$ {The $ASpath$ to $dst$ associated with peer $p$}
8 $NewASpath_{dst}$ = Compute the Preferred $ASpath_p$ from
$\quad\quad$ the announcements associated with all the peers
9 If ($NewASpath_{dst} \neq ASpath_{dst}$)
10 $\quad ASpath_{dst} = NewASpath_{dst}$
11 $\quad$ If ($NewASpath_{dst} == \{\}$)
12 $\quad\quad$ Send message ($withdrawal$,{},$dst$) to each peer
13 $\quad\quad LastAnnouncedASpath_{dst} = \{\}$
14 $\quad\quad NextHop_{dst} = NULL$ {$NextHop_{dst}$ to be used for routing packets to $dst$}
15 $\quad$ Else
16 $\quad\quad NextHop_{dst} = p*$,
$\quad\quad$ { where $p*$ is the peer through which $NewASpath_{dst}$ was announced }

16a $\quad\quad$ If ($NewASpath_{dst}$ less preferred than $LastAnnouncedASpath_{dst}$)
$\quad\quad$ {An empty path ({}) is considered longer than any other path}
16b $\quad\quad\quad$ If ($currentTimeStamp - LastAnnouncedTime_{dst} < minRouteAdver$)
16c $\quad\quad\quad\quad$ Send message ($withdrawal$,{},$dst$) to each peer
16d $\quad\quad\quad\quad LastAnnouncedASpath_{dst} = \{\}$

17 $\quad\quad$ If ($currentTimeStamp - LastAnnouncedTime_{dst} \geq minRouteAdver$)
18 $\quad\quad\quad$ SendAnnouncement(dst)
19 $\quad\quad$ Else
20 $\quad\quad\quad$ SendAnnouncement(dst) at time $LastAnnouncedTime_{dst}$+$minRouteAdver$

21 SendAnnouncemnt(dst)
22 If ($LastAnnouncedASpath_{dst} \neq ASpath_{dst}$)
23 $\quad$ send message ($announcement$,$ASpath_{dst}$,$dst$) to each peer
24 $\quad LastAnnouncedASpath_{dst} = ASpath_{dst}$
25 $\quad LastAnnouncedTime_{dst} = currentTimeStamp$

Fig. 2. Original and modified/new (in internal frame) BGP pseudo code Code

tion of a destination. Here we discuss the impact of the flushing rule in the case that a path has failed and the destination is still reachable but through a longer path. In this case, (fail-over) there is a valid substitute path to the destination ($E_{longer}$).

One may wonder if the ghost flushing rule may not harm the convergence complexity in case of $E_{longer}$. Since due to the ghost flushing rule one may announce withdrawal even if there is an alternate path. Notice, however, that in the ghost flushing rule, a withdrawal is announced only in the case in which due to BGP $minRouteAdver$ timer, the router cannot announce the new $ASpath$. Hence, in all the cases where the announcement is the first announcement after at least $minRouteAdver$ time from the last announcement, the ghost flushing rule acts the same as in BGP. Moreover, the withdrawal is sent only in the cases where the last $ASpath$ that was sent is not relevant any more. BGP in this case does not inform the neighbor that the $ASpath$ is no longer relevant, this has two drawbacks: (1) the ghost cycling in the network, and negatively effecting the convergence. (2) The routing decision is done according to the wrong $ASpath$. Hence the packet would route to the wrong direction, and in many cases would cycle until the TTL expires. The ghost flushing rule in this case

has two positive effects: (1) Flush the ghost information quickly. From claim 1 after $k$ time unites there are no ghost $ASpathes$ of length shorter or equal to $k$. (2) Eliminating the fact that packets route to the wrong direction, since old $ASpath$ information is flushed by the flushing withdrawal message.

The following scenario, (Figure 4) demonstrates the two effects of the flushing rule in case of $E_{longer}$. The network contains a clique and a backup long path. The route to node $x$ (node 0) from any node in the clique should change to go over the alternate long path. However as can be seen in states (c) and (d), the dissemination of the backup path is delayed, due to the ghost information in the clique. By using the modified BGP with the flushing rule all the ghost information disappears in 4 seconds (assuming $h = 1$) and then the backup path converges in additional 4 seconds (instead of 121 seconds). Moreover, in the BGP scenario, packets originated from the clique and that are destined to $x$, would cycle in the clique, in the duration of the convergence time, until their $TTL$ expires (since the convergence time is as high as 121 seconds, which is more than the time it takes a packet to traverse $initial - TTL$ hops).

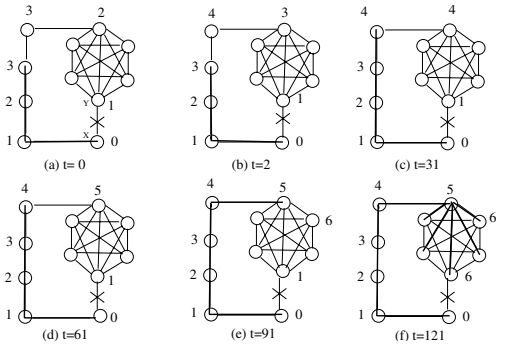


Fig. 4. Illustration of the impact of ghost information in case of a fail-over. Near every node there is the length of its $ASpath$. The ghost information that cycle in the clique, as in the clique scenario, slows down the spread of the backup path.

Notice, that in the case of fail-over, we cannot analytically prove that we reduce the convergence time and we need to support this claim by simulation. As was noted in [8] in this case, the convergence time, may be dominated by two factors - the time until all the ghost information vanishes, and the time it takes to the backup path to propagate in the network assuming no ghost information delays it. In all the cases where the ghost information is a dominating factor of the slow convergence time and not the propagation of the backup path (as in example 4) - the flushing rule would help.

Our simulation (see subsection VII-E), shows that in the majority of the cases the flushing rule has better time convergence than the standard BGP also in $E_{longer}$.

## VI. Ghost Buster rule

In the previous section we proved that the flushing rule guarantees convergence within $O(n)$ time units. Here we want to show that in most likely situations the flushing rule would cause convergence within $O(d)$ time units, where $d$ is the network diameter. Essentially the flushing rule reduces the convergence latency because while the announcements propagation in the network is slowed down by the $minRouteAdver$ the withdrawal messages are forwarded as fast as possible by the flushing rule. Thus, the withdrawal messages act as a cleaning process that eats up the ghost information while the ghost information is being blocked by the $minRouteAdver$ delay. We observe that adding a more aggressive rule, the ghost buster rule, on top of the flushing rule gives a convergence time of $O(d)$. In the ghost buster rule not only the withdrawals are propagated as fast as possible, but we make sure that announcements are guaranteed to be delayed.

Notice that in the original BGP algorithm, in most cases, announcements are delayed due to the $minRouteAdver$ timer, especially when the $minRouteAdver$ is implemented per peer and not per destination (i.e., for each announcement sent on the corresponding interface, and not for each announcement that corresponds to the same destination). Moreover, a more aggressive delay of announcements may occur due the route damping mechanism [16], in cases of destinations that change their route frequently. Hence an effect similar to the ghost buster algorithm occurs in the modified BGP with the ghost flushing rule.

Specifically:

```
A router announces the preferred
    new ASpath to its peering,
iff it received the announcement
    about the new ASpath at least
    delta seconds ago,
otherwise
    it suppresses the announcement
    until delta time passes.
```

The key parameter of the ghost busting rule is the ratio between the speed at which withdrawals propagate and the speed at which announcements propagate.

*Definition 5:* We denote by $K$, the rate of the algorithm, which equals to $K = \frac{delta+h}{h}$.

*Lemma 6.1:* Under the busting rule, $t$ the convergence time following an $E_{down}$ event is $hd\frac{K}{K-1}$ seconds.

Sketch of proof:

For the sake of clarity we take $h$ as a bound (and not average time) on one hop delay of any BGP message delivery, including the processing time. However a similar but more complex proof can be shown where $h$ is the average delay on one hop. By the ghost buster rule the length of the maximum ghost $ASpath$ in the network can increase by one only once in $delta + h$ time. From the ghost flushing rule each $h$ time units the ghost $ASpath$ variable with the minimum length disappear (Lemma 5.1)

Since the maximum length of an $ASpath$ is $d$. The equation for $t$ is:

$$d + \frac{t}{delta + h} = \frac{t}{h}$$

by the definition of $K$ we can replace $delta + h = Kh$ and get

$$d + \frac{t}{Kh} = \frac{t}{h}$$

and get

$$t = \frac{dhK}{K - 1}$$

∎

Take note that the proof about the ghost buster rule requires that any new announcement be delayed, even an announcement regarding a more preferable $ASpath$. Otherwise, we cannot assume that the head of the ghost segment would grow by one hop each $delta + h$ time, since it may encounter a node with a ghost $ASpath$ which is less preferable (the ghost $ASpath$ can encounter even an empty $ASpath$). Hence, the ghost buster rule has a negative effect in cases of $E_{up}$ and $E_{shorter}$ since it requires that any new announcement be delayed . However, as mentioned earlier, we argue that BGP with the ghost flushing rule act according to the ghost buster rule anyway, due to the $minRouteAdver$ rule.

## VII. Simulation

In this section we compare the time convergence of the basic BGP protocol and the modified BGP protocol, that uses the Ghost flushing rule (see basic code 2) in real and artificial topologies. The simulation results support our claims that the modified BGP performs similarly to the Ghost buster rule or to the **reset rule** rule and hence the convergence time is reduced from $30n$ to $dh$.

### A. General Description of the Simulation

We implemented BGP as described in code 2. For the ease of implementation we assume that BGP chooses the routes according to the shortest path metric, with no restriction on advertising or accepting messages from peers resulting from policy routing. Tie breaking rule for two equal length routes is based on the ID (AS number) of the peer that advertises that route. Each node represents one AS (autonomous system).

In each simulation interval, each node receives all the messages from its FIFO input buffer, processes them (we assume it takes at least 0.25 seconds), builds new messages and passes these messages from its output buffer to its peers input buffers. We give a random latency to each message ranging between 0.25 and 2 seconds. The initial value of each node's MinRouteAdver timer is set randomly to a value between 0 and 30 seconds.

Using the simulation we checked the effect of the removal of either one route (removing of some destination inside an AS) or an edge on the convergence time of the network. At time t=0 all the nodes in the graph were initialized with routing tables as if BGP has reached a stable state.

### B. Clique

We started with the simple case of a clique network (complete graph), where we measured the convergence time after the failure of a route to destination $dst$ (see Figure 1) as a function of the size of the clique. As can be seen the modified BGP, has a fix convergence time, since its behavior is proportional to $d$).

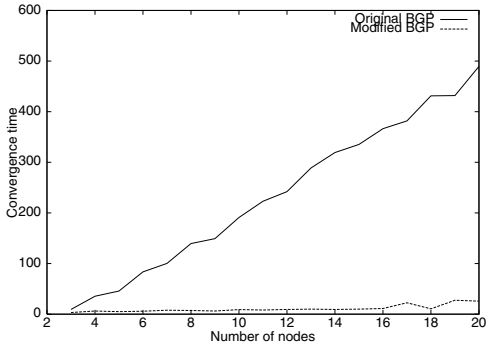


Fig. 5. The convergence time of the original BGP and the modified BGP following the failure of a destination in one of the AS's in a Clique network. As can be seen for a network with 20 ASs the convergence latency of the original algorithm is around 500 seconds, while the new (modified) algorithm converges after 18 seconds.

### C. Real Topology

We tested both algorithms performance on the topology that was studied in [1] (See Figure 6 which is based on the real example taken from the Internet topology). At time $t = 0$ we removed destination $dst$ that is announced by $AS3$.

We repeated the experiments 100 times, and draw a histogram of the convergence times. As can be seen in

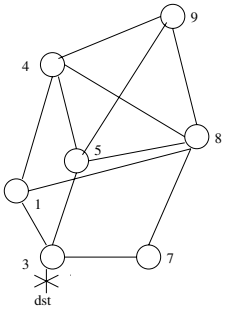


Fig. 6. A topology similar to the one used by [1].

Figures 6 the modified BGP shows dramatic and stable improvements.

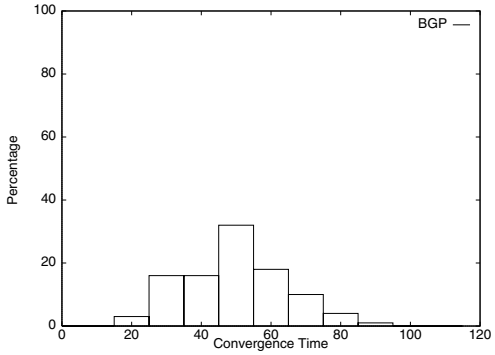


Fig. 7. A histogram of convergence Times using the original BGP on the topology of Figure 7

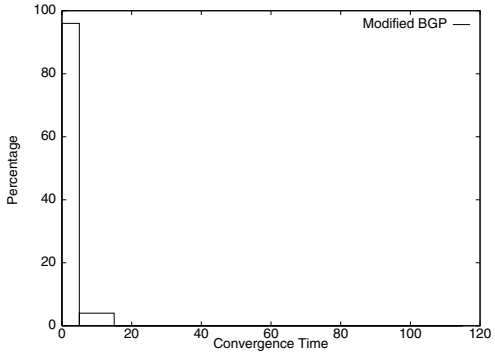


Fig. 8. A histogram of convergence Times using the new (modified) BGP on the topology of Figure 7.

## D. Core Of The Internet

Finally, we compare the convergence times of the two algorithms on the core of the AS network taken from the Internet.

We took the Routing Table of the route server [17] (which is a route server of $41$ BGP routers) from consecutive days $(4.12.00, 5.12.00, 6.12.00)$ and created from it the AS graph based on all the ASpath's of all the routes in the table. For example if there is an ASpath $23, 43, 123$ to destination $dst$, we add to the AS graph the directed edges $(23, 43)$ and $(43, 123)$. After building the AS graph we have recursively removed nodes that their out-degree is zero. At the end we were left out with the core which includes $375$ nodes with $2186$ edges.

We randomly chose ASes from the core, and simulated the failure of a destination (network) inside this AS.

The simulation results (Table II show that the new (modified) BGP dramatically improves the convergence time. This can be explained by the fact that the core of the Internet is similar in its parameters to a clique. Notice, that we took only the core of the internet, since only in the core the ghost information may cycle and harm convergence.

| | Out-degree AS | In-degree AS | BGP | Modified |
|---|---|---|---|---|
| 1 | 45 | 10 | 963 | 22 |
| 2 | 52 | 17 | 898 | 51 |
| 3 | 3 | 4 | 1031 | 36 |
| 4 | 112 | 27 | 1017 | 50 |
| 5 | 61 | 11 | 1034 | 36 |
| 6 | 20 | 24 | 920 | 33 |
| 7 | 1 | 6 | 2 | 2.5 |
| 8 | 18 | 13 | 1111 | 54 |
| 9 | 1 | 19 | 981 | 62 |
| 10 | 1 | 1 | 4 | 5.5 |

TABLE II
CONVERGENCE TIME AT THE CORE OF THE INTERNET.

## E. Link failure

We measure the convergence time in the topology of Figure 4, in the case of a link failure (i.e., $E_{longer}$) as function of $n$. The topology consists of a clique of size $n/2$ and one alternate path between two nodes of the clique of length $n/2$. The results are given in Figure 9 and show that the modified BGP algorithm converges much faster also in this topology $E_{longer}$ (a topology in which the path becomes longer, without loosing any destination).

The above topology of subsection VII-E is unique because, in this case, the ghost information may dramatically harm the convergence of the fail-over event. This is due to the fact that the backup path is dramatically longer than the the original path. Hence, the ghost message may traverse for a long time and disappear only when the ghost $ASpath$ becomes longer than the back-up path. In order to see the impact of the ghost flushing rule in topology in which the ghost $ASpath$ vanishes quickly, we repeat the experiment of link failure on the real topology of Figure 6. Here the back-up path is a similar length as the original path.

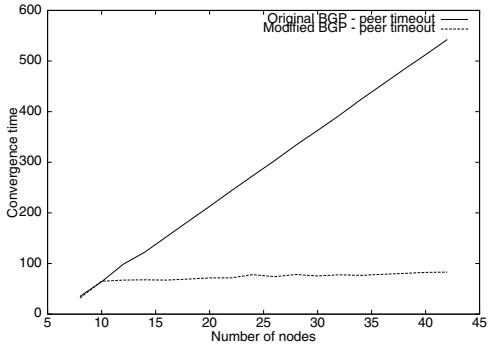


Fig. 9. The convergence time of the original BGP and the modified BGP algorithms following the failure of a $link$ in the topology of Figure 4. As can be seen the convergence latency of the original algorithm is more than 500 seconds (with 40 nodes), while the new (modified) algorithm converges after about 100 seconds.

| Edge | BGP conv. ASpath | BGP conv. NextHop | Flushing conv. ASpath | Flushing conv. NextHop |
|---|---|---|---|---|
| 1-3 | 29.76 | 21.95 | 29.60 | 1.94 |
| 1-4 | 28.10 | 0.50 | 27.35 | 14.65 |
| 1-8 | 29.31 | 21.76 | 28.69 | 1.82 |
| 3-4 | 29.19 | 17.85 | 27.90 | 15.76 |
| 3-5 | 28.85 | 19.62 | 32.06 | 16.21 |
| 3-7 | 37.10 | 12.67 | 37.00 | 20.28 |
| 4-5 | 29.09 | 18.53 | 28.78 | 16.57 |
| 4-8 | 28.60 | 0.50 | 28.32 | 12.05 |
| 5-8 | 27.91 | 0.50 | 27.00 | 0.50 |
| 5-9 | 30.10 | 20.31 | 29.53 | 18.73 |
| 7-8 | 27.71 | 18.03 | 27.17 | 1.35 |
| 8-9 | 29.68 | 22.54 | 28.18 | 1.70 |

TABLE III

THE CONVERGENCE TIME IN CASE OF FAIL-OVER.

For each link in the graph, we simulate its failure and calculate the time convergence after the failure using the BGP and the BGP with the flushing rule. We repeat the test 40 times and calculate the average. We use two definitions of convergence (in case the fail-down events are identical). The first definition, $ASpath$ *convergence*, is the convergence time until the $ASpath$ is correct. The second definition, $nextHop$ *convergence*, is the convergence time until the $nexthop$ induced by the $ASpath$ is correct, that is, until the routing according to the table is correct. We gave a detailed scenario in subsection V-A where we explained how a network may converge faster according to $nextHop$ convergence than according to $ASpath$ convergence. Our results showed that, while BGP and BGP with the flushing rule converge more or less in the same time (32.3 vs 31.9) according to the definition of $ASpath$ *convergence* , the BGP converges with the flushing rule faster than the original BGP according to the $NextHop$ *convergence* definition (15.8 vs 11.08).

## VIII. Conclusion

One conclusion from this work is how sensitive BGP is to minor modifications of some of its parameters. We believe this sensitivity is shared with most distributed algorithms in which a small twist could turn things around.

We note, that a careful look shows that the flushing rule technique does not depend on the $ASpath$ propriety, and any other metric may replace the $ASpath$ selection rule in the technique. Hence, as a by product of this work, a new stateless mechanism to overcome the counting to infinity problem is provided, which compares favorably with other known stateless mechanisms (in RIP, and IGRP).

## References

[1] C. Labovitz, R. Wattenhofer, S. Venkatachary, and A. Ahuja, "The impact of internet policy and topology on delayed routing convergence," in *Proc. Infocom*, April 2001.

[2] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanianitz, "Delayed internet routing convergence," in *Proc. Sigcomm*, September 2000.

[3] Timothy G. Griffin and Brian J. Premore, "An experimental analysis of bgp convergence time," in *Proc. , ICNP' 2001*, Nov 2001.

[4] L. Gao and J. Rexford, "Stable interent routing without global coordination," in *Proc. SIGMETRICS*, June 2000, http://www.cisco.com/ nodnik/walla.html.

[5] T. Griffin and G. Wilfong, "A safe path vector protocol," in *Proc. IEE INFOCOM*, Mar 2000.

[6] T. Griffin and G. Wilfong, "An analysis of bgp convergence properties," in *Proc. ACM SIGCOMM*, Aug 2000.

[7] Davor Obradovic, "Real-time model and convergence time of bgp," in *Proc. IEEE INFOCOM*, 2002.

[8] D. Pei, X. Zhao, L. Wang, D. Massey, A. Mankin, S. Felix Wu, and L. Zhang, "Improving bgp convergence through consistency assertions," in *Proc. IEEE INFOCOM*, 2002.

[9] R. Perlman, *Interconnections, Bridges and Routers*, Addison-Wesley, 1992, 1992.

[10] B. Halabi, *Internet Routing Architectures*, New Riders Publishing, Cisco Press, 1997.

[11] W. Zaumen and J. Garcia-Luna-Aveves, "Dynamics of distibuted shortest-path routing algorithms," in *Proc. of ACM SIGCOMM*, August 1991.

[12] J. Garcia-Luna-Aceves, "Loop-free routing using diffusing computatuions," *IEEE Transactions on Networking*, Feb 1993.

[13] G. Malkin, "Rip version 2," Tech. Rep., IETF, November 1998, RFC 2453.

[14] Charles L. Hedrick Rutgers, "An introduction to igrp," Tech. Rep., Cisco site, 1991, White Paper: http://www.cisco.com/warp/public/103/5.html.

[15] Y. Rekhter and T. Li, "A border gateway protocol 4 (bgp4)," Tech. Rep., IETF, 1999, draft-ietf-idr-bgp4-09.txt.

[16] C. Villamizar, R. Chandra, and R. Govindan, "Rfc 2439: Bgp route flap damping," Tech. Rep., IETF, 1998, http://www.faqs.org/rfcs/rfc2439.html.

[17] Route Server, "Host," Tech. Rep., Oregon, 2000, route-views.oregon-ix.net.