Improved BGP Convergence via Ghost Flushing

Yehuda Afek
Tel-Aviv University
afek@cs.tau.ac.il

Anat Bremler-Barr
Interdisciplinary Center Herzliya
bremler@idc.ac.il

Shemer Schwarz
Tel-Aviv University
shemers@hotmail.com

Abstract

In [**?**, **?**] 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). 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).

Keywords: Routing protocol, BGP convergence

1 Introduction

1.1 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 [**?**, **?**]. In [**?**] the authors go as far as making the following recommendation:

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

In [**?**] 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 [**?**, **?**] 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 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 [**?**] 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 [**?**] 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, [**?**] shows that in this case we also may get unacceptably long convergence time. Simulation done by [**?**] 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 [**?**, **?**, **?**, **?**, **?**, **?**, **?**, **?**], 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 [**?**, **?**] in a different disguise (of course, in BGP the counting is limited since BGP is using the AS-path to avoid loops). There are known techniques to overcome this problem that add state information to the BGP messages [**?**, **?**]. Introducing these changes into BGP would make it a more expensive and complex protocol.

There are other solutions that do not require state information [**?**, **?**], such as, Reverse Poisoning (used in RIP [**?**]), Route Poisoning with Hold down timers and trigger update (that are used in Cisco's IGRP [**?**]). 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 detailed discussion of the relations between the presented technique and various counting-to-infinitely prevention techniques is given in Section 9.

A new solution to reduce the convergence time complexity was recently introduced in [**?**]. 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.

1.3 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*. Ghost flushing rule is the key contribution, the ghost buster rule mostly serves as a tool to understand why the ghost flushing rule works so well. We also analyze a third suggestion, called *reset rule* which shades more light on the convergence behavior of BGP.

The simplest and the one that makes the most difference is the *ghost flushing rule* presented in Section 5, 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 [**?**, **?**] (in which d is about 7, and h is less than 2 seconds) see Figures 6 and 7b. Notice that the reduction in the convergence latency comes with a corresponding reduction in the message complexity, message loss and retransmission.

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 (except in some pathological cases which are analyzed in Section 5.1).

Simulating and verifying the effectiveness of ghost flushing we noticed that it performs in reality (actual Internet topologies and standard BGP implementation) better than expected. Analyzing the results we concluded that the reason for the better than expected convergence is the way *minRouteAdver* in used in BGP. To better understand this behavior we devise yet another modification, the *ghost buster rule* in Section 6, in which we force additional delays similar to the *minRouteAdver*. This modification primarily serves as a tool to understand and analyze the reasons due to which the ghost flushing modification of BGP works better than expected.

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 δ 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. The second observation on the flushing rule, which results in a different variant of BGP, called **reset rule** is actually a sequence of two observations with the same principle (Section 7): delay announcements long enough to ensure that all the ghost information has been removed by the withdrawal messages. I.e., taking a global approach to first "clean" the network from ghost information and only then to allow announcement propagation. While this variant has the disadvantage of delaying announcements, also valid ones, for a long time and hence it is not practical, it shades light on the actual convergence behavior of BGP.

Table 1 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 8 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	$\frac{kdh}{k-1}$	$\frac{2Ekdh}{30(k-1)}$
with reset rule-1	$(3d+2)h$	$\frac{2E(3d+2)h}{30}$
with reset rule-2	$(d+4)h$	$\frac{2E(d+4)h}{30}$

Table 1: 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 δ is the time by which the announcement is delayed at each node, i.e., $\delta \simeq minRouteAdver$).

2 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. A router running BGP may also take a policy decision, such as, not to send or not to receive a specific announcement from different neighboring peers. For example, an AS may avoid announcing its *ASpath* to a destination, since it does not want to be a transit network for that destination.

Careless usage of the sophisticated mechanisms could lead to unsafe policies [**?**], which could result in route oscillations. Some possible mechanisms to avoid these problems were introduced in [**?**, **?**]. However, in this paper we do not deal with this problem.

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 [**?**, **?**].

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 [**?**]). 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.

3 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 [**?**, **?**] 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 [**?**], 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 [**?**] 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) [**?**] 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 [**?**] (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 [**?**] 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, $0 \leq i < m$ $ASpath^{r_i} = AS_{r_i}, AS_{r_{i+1}}, .., AS_{r_m}$.*

We define the **routing tree** to destination *dst* at convergence time as the tree induced by the routing decisions of the routers in the network when the network is in a stable state. I.e., the set of links on which traffic to *dst* would be forwarded. The graph is a tree, due to the property of a valid path in convergence time, and the assumption that each node in the graph has only one route to the destination.

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.

4 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 [**?**, **?**] 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 (we omit the use of *dst*, which is the same for all):

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 [**?**] in most cases the architecture is more complex and the average convergence latency is 3 minutes and more.

In [**?**] 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.

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 [**?**], 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'.

5 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

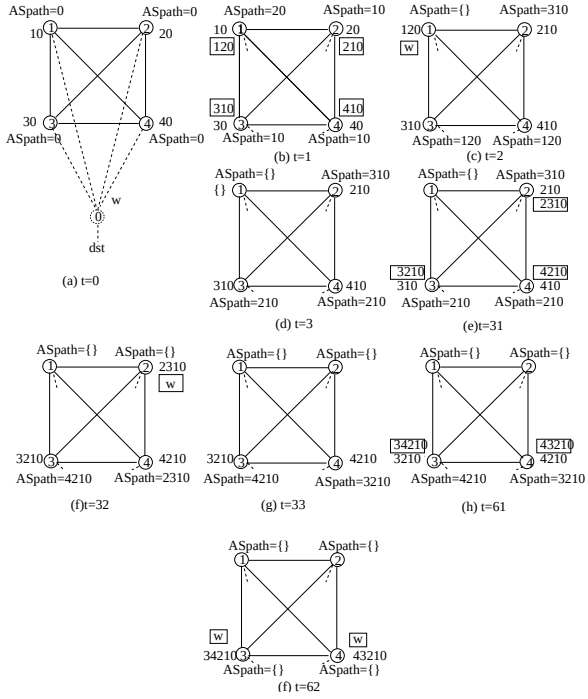

Figure 1: Illustration of the ghost information problem in a clique. Next to each node we write its current *ASpath* (with *ASpath* = .. above or below the node). This is the path to the destination *dst* that appears in this node routing table in the corresponding snapshot. Next to the node, we depict 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.

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. Since the ghost flushing rule is applied only when there is a less preferred path, the application of the rule does not effect the time convergence or message convergence in the events of $E_{shorter}$ and E_{up} in comparison to the original BGP. Notice, that the decision whether the *ASpath* is less preferred is a local decision according to the preference of the router that receives the BGP message.

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 $|ASpath| > 1$.

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

1 If ($type == Withdrawal$)
2 $\quad ASpath_{dstp} = \{\}$
3 If ($type == Announcement$)
4 $\quad$ If $AS_r \in PeerASpath_{dst}$ {The loop detection/prevention mechanism }
5 $\quad\quad ASpath_{dstp} = \{\}$
6 $\quad\quad$ Else
7 $\quad\quad ASpath_{dstp} = PeerASpath$ {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 $\quad LastAnnouncedTime_{dst} = currentTimeStamp$

Figure 2: Original and modified/new (in internal frame) BGP pseudo code Code

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. ∎

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.

5.1 Fail-over

So far we discussed the effect of the ghost flushing rule on the convergence complexity following the disconnection of a destination. Here we discuss the impact of the flushing rule in the case

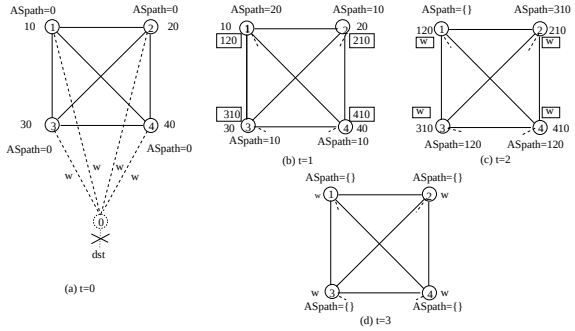

Figure 3: Depicting the same situation as in Figure 1. Clearly the scenario is much shorter in time since the flushing technique is used.

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).

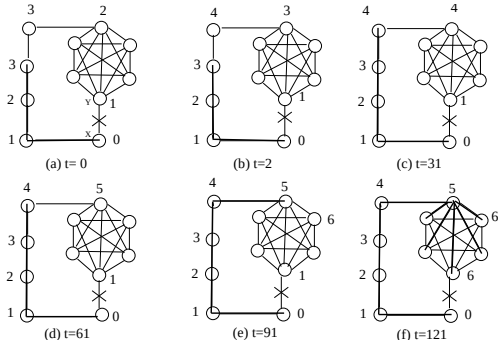

Figure 4: Illustration of the impact of ghost information in case of a fail-over. Near every node there is the length of its *ASpath*. The ghost information that cycle in the clique, as in the clique scenario, slows down the spread of the backup path.

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 [**?**] 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. In these cases the flushing rule helps not only to reduce the convergence time but also to reduce the message complexity. The message complexity of the first process, the vanishing of the ghost information, is according to lemma 5.2. The message complexity of the second process, i.e., the propagation of the new ASpath, is not any worse because of the ghost flushing rule. Since, when all the ghost information disappears, the state of the destination at the router, may change only to a more preferred ASpath.

We note that there are some extreme cases in which BGP has a better convergence time than BGP with the flushing rule. These are the cases in which while the *ASpath* of each router is incorrect, the local next hop decision induced by the *ASpath* is correct. Consider for example the scenario given on the network of Figure 5. Assume that in the stabilized network before the failure node x routes packets to destination dst through node k. After the failure of link $a - b$, x first receives a withdrawal indicating the failure in the path to dst through its neighbor k. Hence, node x changes its path to go through m and announces the change

to its neighbor s (s would change its $ASpath$ to $sxm....$ Next x receives a withdrawal about the path to dst from its neighbor m. Node x then changes its path to go through y (assume that before the failure event y's route to dst is through z). In this case node x has the correct route to dst in either BGP with or without the flushing rule. The difference between the two algorithms is with the route node s has to dst. In the standard BGP, x does not announce its new $ASpath$ to s due to the $minRouteAdver$ timer. While it delays this announcement, incidentally, the packets to dst would be routed correctly from s to dst: Node s chooses the correct next hop, although its $ASpath$ is incorrect ($sxm...$) and node x routes through y, and from there the packet is routed correctly to dst. However, if we use the flushing rule, s would receive a flushing $withdrawal$ and hence would not have a route to dst.

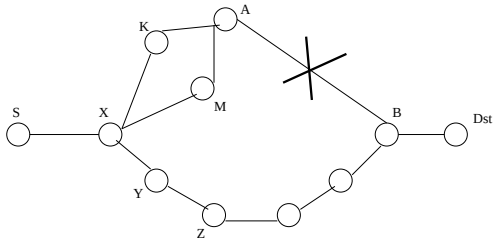

Figure 5: A scenario showing a unique case in which BGP has a better convergence time than BGP with the flushing rule.

The inverse effect of the above scenario may happen too, i.e., it could be that BGP decides that there is no route, while there is a path to the destination, and in the exact same scenario BGP with the flushing rule makes the correct decision. Consider for example the same scenario, but now y prefers a route to dst that passes through x and not through z, before the failure event. In this case using the standard BGP, the router x after receiving a withdrawal from m would not have a valid path, and will send a $withdrawal$ to s. Neither x nor s could route to dst, even though there is a route to dst.

Our simulation (see subsection 8.5), shows that in the majority of the cases the flushing rule has better time convergence than the standard BGP also in E_{longer}.

6 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.

In order to understand and analyze the effect of combining the ghost flushing rule with the *minRouteAdver* rule, we analyze a more aggressive rule, the *ghost buster rule*. Implementing it 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 [**?**], 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.

7 Reset rule

The idea behind the ghost buster rule is that the withdrawal message acts as a cleaning process that eats up the ghost information while the ghost information is being blocked by the *minRouteAdver* delay. However during this process, a segment of ghost information may traverse some steps in the network before the withdrawal catches the ghost information. Another approach is blocking the ghost information for such a long time period so as to ensure that the withdrawal has eliminated all the ghost information from the network. We have called this algorithm the reset rule. This mechanism is similar to the holddown timer plus route poisoning of IGRP (see subsection 10.2).

In this section, we try to evaluate the value of *minRouteAdver* in three diffident variants, all of which ensure that all ghost information disappears within one *minRouteAdver* interval. After that interval the propagation of the correct announcement starts. While the three variants are not the most practical, since they require a large delay of announcements, they capture interesting aspects about the convergence behavior of BGP.

At first glance, it seems like setting the *minRouteAdver* timer to be dh is enough (since this is the time it takes the withdrawal to reach the entire network). However, the value

of *minRouteAdver* should increase, assuming that we stick with the rule that the less preferred ASpath (possibly ghost information) may be forwarded by the router if it is the first announcement after *minRouteAdver* timer without an announcement. The case in which the first announcement is not delayed is important in order to reach fast convergence when the less preferred *ASpath* is a result of E_{longer} and not E_{down}. This, however, leads to the fact that *minRouteAdver* should be set higher than dh, since in the case of E_{down} ghost information traverses d hops without encountering a node that is incapable of sending an announcement, due to *minRouteAdver*.

In this section we show the mechanism in three models:

- **reset rule-1:** If $minRouteAdver \geq 3dh + 2$ and the flushing rule is employed, then we can prove that the E_{down} convergence latency is at most $3dh + 2$ seconds.
- **reset rule-2:** The following variation on the mechanism of *minRouteAdver* (that some ISPs [**?**] decided to employ) :

```
A router would send
    a new announcement
iff at least minRouteAdver
    elapsed from the previous
    sent announcement,
    and
    from previous sent withdrawal.
```

  together with the flushing rule, ensures that if $minRouteAdver \geq (d + 4) \cdot h$ then the E_{down} convergence latency is at most $(d + 4) \cdot h$ seconds.

- **reset rule-3:** When applying the previous model to synchronous networks, it is enough if $minRouteAdver > 4h$ in order to show that the E_{down} convergence time is at most $(d + 4)h$ seconds.

The huge difference between the synchronous case and the asynchronous case is due to some pathological cases. Hence in the majority of the cases, also in asynchronous networks, it is enough to have a small *minRouteAdver* to ensure that the time is at most function of $d + 4$ in the model of **reset rule-2**.

The reset rule has a disadvantage over the ghost buster rule, since the reset rule timer has the misfunction of adding a huge delay (proportional to broadcast time, that is, proportional

to dh, where d is the diameter after the failure.) The ghost buster rule adds only a small delay to each announcement. Moreover, with the ghost buster rule even small delay helps; to produce an aggressive reduction in convergence time, one should set it proportional to h (without taking into account d) and get a time convergence which is proportionally to the actual dh.

In reality there are different variant implementations of the *minRouteAdver*. A common variation is to have one such timer for each peer and not per destination. The motivation for this is to store one timer per peer and not per destination (in a BGP table there could be up to 120,000 subnetwork destinations). Hence in practice the announcement message are delayed even more, which works to our advantage and when used with the flushing rule would result in a significant reduction in the convergence latencies. We try to evaluate if in most cases *minRouteAdver* which is set to 30 seconds is high enough to act as the delay required by the *reset rule*.

We observe that the diameter d of the Internet AS graph is very small. In the matrices of shortest path $d = 8$ and looking on the real $ASpath$ lengths (where the policy routing is taken into account), yields $d = 12$. Hence for the **reset rule -2** mechanism it would be enough to assure convergence by setting the *minRouteAdver* to be larger than $d + 4 = 16h$ seconds (which is less than 30 seconds). However for the **reset rule-1**, it requires that *minRouteAdver* would be larger than $(3d + 2)h = 38h$ (which if h is larger than 1 second a reasonable assumption when including inter AS delay) gives us a *minRouteAdver* larger than 30. Hence the *minRouteAdver* would play the rule of **reset rule -1** only in some of the cases.

The proofs of the three variations of the reset rule appear in Appendix 12.

8 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. We did not simulate the Ghost Buster rule and the resets rules, since their purpose is to capture the behavior of BGP with the ghost flushing rule and understanding its effect, and as we argue above the only rule that we recommend implementing and which would than reduce the convergence time and message complexity is the ghost flushing rule.

8.1 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.

8.2 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).

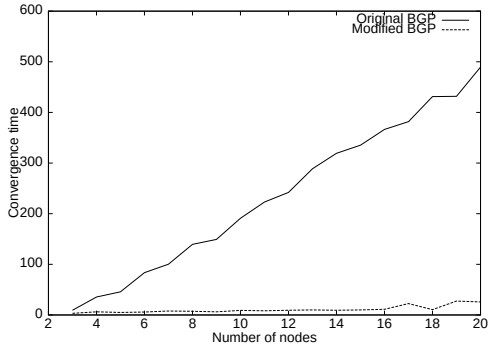

Figure 6: The convergence time of the original BGP and the modified BGP following the failure of a destination in one of the AS's in a Clique network. As can be seen for a network with 20 ASs the convergence latency of the original algorithm is around 500 seconds, while the new (modified) algorithm converges after 18 seconds.

8.3 Real Topology

We tested both algorithms performance on the topology that was studied in [**?**] (See Figure 7 which is based on the real example taken from the Internet topology). At time $t = 0$ we removed destination *dst* that is announced by $AS3$.

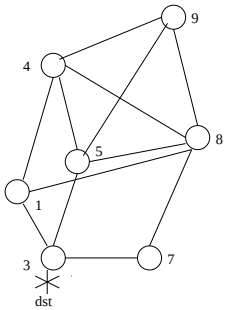

Figure 7: A topology similar to the one used by [1].

We repeated the experiments 100 times, and draw a histogram of the convergence times. As can be seen in Figures 8 and 9 the modified BGP shows dramatic and stable improvements.

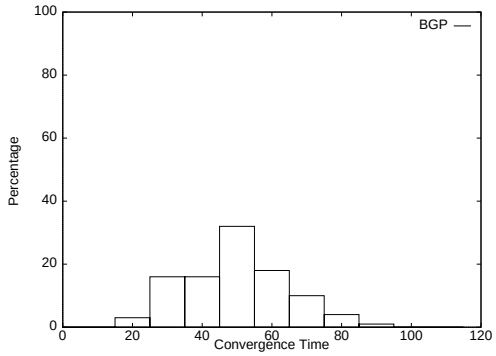

Figure 8: A histogram of convergence Times using the original BGP on the topology of Figure 7

8.4 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 [**?**] (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.

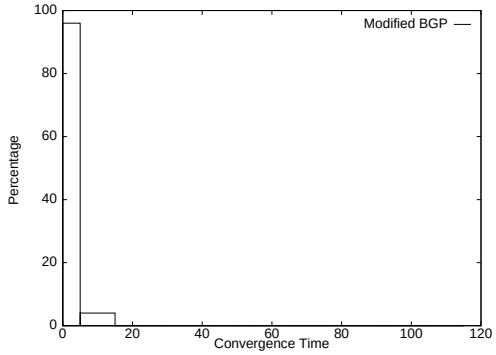

Figure 9: A histogram of convergence Times using the new (modified) BGP on the topology of Figure 7.

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

The simulation results (Table 2 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 2: Convergence Time at the core of the internet.

8.5 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 10 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).

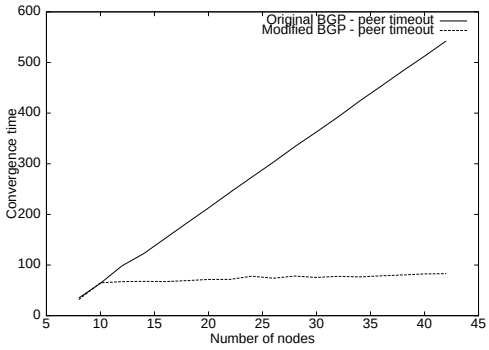

Figure 10: The convergence time of the original BGP and the modified BGP algorithms following the failure of a *link* in the topology of Figure 4. As can be seen the convergence latency of the original algorithm is more than 500 seconds (with 40 nodes), while the new (modified) algorithm converges after about 100 seconds.

The above topology of subsection 8.5 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 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 7. Here the back-up path is a similar length as the original path.

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 5.1 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).

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 3: The convergence time in case of fail-over.

9 Related Work: Comparison with Counting to infinity Prevention techniques

While the main goal of this work is to improve the convergence time of BGP, the ghost buster rule and ghost flushing rule may also be beneficial to the understanding of distance vector protocols and dealing with their counting to infinity problem.

Notice that counting to infinity is a slightly different problem than the BGP convergence. In counting to infinity we try to reduce the convergence time, so it wouldn't be bounded by infinity, the bound of the maximum hop count. In BGP the convergence time is bounded by n - the maximum length of a loop free $ASpath$.

A careful look at the proof leads to the following conclusion: all the mechanisms we introduced at the paper (ghost buster, flushing, and reset rule) do not use or rely on the $ASpath$. Therefore, these mechanisms can be added to any distance vector algorithm that prevents looping to infinity. Furthermore these mechanism bound the convergence latency following E_{down} or E_{longer} by a function of d.

Moreover one may adjust, distance vector protocols, such as RIP, so that the counting to infinity problem is solved, without changing the algorithm, only by either changing or adding timers. However, the *ghost buster algorithm* has a negative effect: adding the delay δ increases the convergence latency of E_{up} to $n \cdot (\delta + h)$ seconds from $n \cdot h$ seconds. However, as in BGP, in many algorithms a version of a delay δ exists anyhow (for example, aggregation of announcements, periodic updates etc).

10 Comparisons with counting to infinity solutions

In this section we review mechanisms to deal with counting to infinity that do not require additional state to the messages:

10.1 RIP mechanisms

Two mechanisms are used in the RIP routing protocol to reduce the counting to infinity problem:

- Split horizon - An announcement for a given neighbor should omit routes that their next hop is that neighbor. This mechanism is aimed at preventing loops between adjacent gateways (loop eliminated in BGP with the ASpath mechanism) and not relevant to BGP algorithm.
- Reverse Poisoning - a variant on Split horizon, where instead of omitting the routes, an announcement is sent with infinity as a distance metric.

10.2 IGRP mechanisms:

IGRP (which is based on RIP) adds the following mechanisms to the RIP solutions:

- Trigger update - The inter-domain distance vector sent a periodic updates on a regular basis (for example in IGRP this timer is by default 90 seconds). Trigger update is an update that is sent immediately after a change in order to propagate failure quickly in the network. However, trigger update, does not discriminate announcements from withdrawals as in BGP with ghost buster rule. Notice that in BGP all the updates are triggered, but with the rule of $minRouteAdver$. The fact that there is no periodic update in BGP is due to the fact that BGP runs over TCP, while RIP and IGRP over UDP or directly over IP.
- Hold-down timers - When a route to a destination is removed, i.e., the current route becomes unavailable, this destination is put into hold-down, where no new route to this destination is accepted. The hold-down time should be set to several times the broadcast time. The default value is 3 times the broadcast time plus 10 seconds, which in the new versions is set to 280 seconds. Notice that this mechanism for handling counting to infinity problem wouldn't help in the BGP clique example (in that bad scenario - a router whose route to a destination is removed does not change again to a new route).
- Route Poisoning - when an announcement increases sufficiently the metric for an existing route the route should be removed.

  There are two types of definitions for sufficiently increasing the metric: (1) Increasing the weight of the route by a factor greater than 1.1 (2) Any increase of the hop count.

(This version is called the stronger version of Route Poisoning.) The motivation for this rule is the fact that route's elongation could be evidence of a loop. Hence, the route is removed, to be on the safe side. If this path is a legitimate one - it will be reinstalled at the next periodic update.

Notice that Route Poisoning resembles the ghost flushing rule. The main different between the two methods is that the Route Poisoning sends a withdrawal in all cases of increasing the distance, while the ghost flushing rule sends a withdrawal only in the cases where it cannot send a new *ASpath*.

Combining Route Poisoning with the hold-down timer has a similar effect as the **reset rule** algorithm. Both clean all the information from the network regarding the failure destination during the hold-down timer. Notice that the **reset rule-1**, unlike Route Poisoning, sends a withdrawal only if it receives a change to a less preferable *ASpath* after *minRouteTimer* from the last change in *ASpath*. This variation results in better convergence time in some the of the E_{longer} cases.

11 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).

12 Appendix

12.1 Reset rule-1

In this subsection we prove that if we use **reset rule-1** then BGP converges in $(3d + 2)h$ time. Recall that in **reset rule-1** we used the *Ghost flushing rule* and set *minRouteAdver* to $(3d + 2)h$ seconds.

Lemma 12.1 *The converge time complexity of* ***reset rule-1*** *follows* E_{down} *is* $(3d + 2)h$ *seconds.*

Sketch of proof:

Let *dst* become unreachable. We show that there is no node or message with non-empty *ASpath* to *dst* after $(3d + 2)h$ time. For clarity and simplicity, we use in the below proofs,

the term *ASpath* to indicate *ASpath* to *dst*, since we are only interested in the convergence of the information to reach *dst*.

Let us first review over the way in which ghost information is created and how it traverses over the network. The source of the ghost information is a value of *ASpath* before the E_{down}. Recall, that this value can be the node's current *ASpath* (and its corresponding next hop) or the information stored in a router for each peer, about the currently preferred *ASpath* of that peer, or the currently *ASpath* in a BGP message. After E_{down} all the old information about *dst* should disappear, however some left over of the information about *dst* before the E_{down} begin to traverse in the network.

Let us recall how an announcement message traverse over the network. When a node v receives an announcement m from a neighbor u, v stores in its memory the *ASpath* of the announcement as the last *ASpath* announced by u. When v finds out that its best *ASpath* goes through u, it chooses to route through v. If *minRouteAdver* does not elapse from the previously announced message, it announces its new *ASpath* in the announcement m'. The *ASpath* in m' will be longer by one than the *ASpath* in the announcement m. We say that the announcement m traverses, and become message m'.

In Claim 5, we showed that a ghost announcement can traverse until its *ASpath* is of the maximum length of $3d+2$ without reaching a block that cannot send a new announcement. We call this node a block node, since the node is blocked from sending new announcements.Hence the block node stop the traverse of the ghost announcement until the *minRouteAdver* timer elapses. The node is blocked, since the *minRouteAdver* timer, which is set to $(3d+2)h$, didnot elapse from its last announcement.

Because the maximum *ASpath* in an announcement that traverses over the network was of length $(3d+2)$, we can conclude that there is also no node with *ASpath* longer than $(3d+2)$ (since before E_{down}, the maximum *ASpath* was d, and after E_{down} the maximum announce *ASpath* was of length $3d+2$, and a node can change its *ASpath* only by adopting a *ASpath* that was announced to it).

Recall that due to the flushing rule, after $(3d+2)h$ time, there is no node or message with *ASpath* of length less or equal to $(3d+2)h$ (From Claim 1 of Lemma 5.1). The lemma follows, since we proved that the maximum *ASpath* is of length $(3d+2)h$ and we proved in Claim 5 that the maximum *ASpath* in the network is of length $(3d+2)h$. Hence after $(3d+2)h$ from E_{down} all the nodes and messages contain a NULL *ASpath* to *dst* . ∎

In order to prove the claim 5 we first need to begin with some definitions:

Definition 6 *A node v is blocked if, according to the minRouteAdver rule, it cannot sends any new announcements.*

Definition 7 *The routing tree to destination dst is the tree induced by the routing decisions*

of the routers in the network when the network is in a stable state before E_{down}. I.e., the set of links with which traffic to dst would be forwarded.

Definition 8 *An edge (u,v) in the graph is a cross edge if it connects two nodes in the routing tree, if u is not the child of v in the routing tree and v is not the child of u in the routing tree.*

Definition 9 *The level of a node is the length of its current ASpath in the stable state (before the failure).*

Let (u, v) be a cross link.

Corollary 2 $||ASpath_u| - |ASpath_v|| \leq 1$ *in the routing tree before the failure.*

Straight forward from the optimality of the routing tree to *dst* before the failure. ∎

Let (u, v) be a cross edge.

Claim 3 *If one of the endpoint u or v changes its original ASpath (its ASpath before the failure), then one of the end points of the cross edge is blocked.*

Let us look at the first time that one of the end points of the cross edge changes its *ASpath* state. Without the loss of generality, assume it is node u. Hence, when u changes its state it has in the memory that its peer v has the original *ASpath*. Hence, u changes its *ASpath* to a valid *ASpath* and not a NULL *ASpath*. Therefore, it sends an announcement, and it cannot send another announcement for the next *minRouteAdver* seconds. ∎

Let us look at x time unit after the link failure, where $x < (3d + 2)h$. Let (u, v) be a cross link. Let v be a node in level k. Let a ghost announcement traverse between u to v. Let v accepts the ghost *ASpath* of the announcements, and send a ghost announcement to all of its peers. Then,

Claim 4 *The new ghost announcements are with ASpath of maximum $k + 3$ length.*

Proof: Let us look at the time just before u sent the announcement to v. In this time, v still has its original *ASpath*. Otherwise u or v is blocked (from Claim 3) and hence u cannot send the announcement to v, or v could not send an announcement to its neighbor after changing its *ASpath*. Contradiction.

The original *ASpath* of u is with maximum length $k + 1$ (from 2), and if u changes its current *ASpath*, its new *ASpath* is with maximum length of $k + 1$, since otherwise it would choose to route through v, whose *ASpath* is of the length $k + 1$.

Hence, when v chooses to route through u (since for example the neighbor that v routes through to dst would change its $ASpath$ to longer one) its new $ASpath$ would be of a maximum of length $k+2$. Node v would announce to each of its peers the new $ASpath$ (of length of maximum $k+3$) to dst through it, and become blocked. ∎

Let us look at time $t \leq 3d+2$ after E_{down} event.

Claim 5 *A ghost announcement has a maximum ASpath of length* $3d+2$ *before the ghost announcement reach a block node.*

Sketch of proof:

We show that a ghost announcement may reach a maximum $ASpath$ of $3d+2$ in its traversal, before reaching a block node.

Let us look at the different courses that a ghost announcement may traverse over a network. The source of the ghost announcement is based on some left over ghost information in the network such as, the current $ASpath$, or the $ASpath$ of a node in the memory of its peerss. The length of this $ASpath$ is smaller than $d+1$.

Let us look at the last cross link that the ghost message traverses, whose two endpoints are not blocked.

If such a cross link exists, the announcement after the traverse on the link has $ASpath$ with length of $d+3$ (from Claim 4). After traversing the cross link, it can traverse the induced routing tree (where the tree is set according to the routing tree to dst before the failure). Due to the anti loop mechanism, the ghost announcement may traversethe maximum length of $d-1$ hops up the tree and then $d-1$ hops down the tree without reaching a cross link, where one of its endpoints is blocked. (Notice that the message cannot traverse to reach the root of the routing tree, since the root is the AS of the dst. The anti-looping mechanism of BGP would prevent this, since in any ghost announcement about dst, the first AS in its $ASpath$ is the AS of dst). Traversing through that cross link, the $ASpath$ may be increased by one more hop. Hence, the maximum $ASpath$ length is $3d+2$ $(= d+3+2(d-1)+1)$.

If such a cross link does not exist, then the message, with value of a maximum of $d+1$ before the failure, may have a maximum traverse up and down the induced routing tree, and then over a cross link that ended in a blocked node (1 more hops). Hence, the maximum $ASpath$ length is of $3d$ $(= d+1+2(d-1)+1$ in this scenario.

Hence, a ghost announcement has a maximum $ASpath$ of length $3d+2$ before the ghost announcement reach a blocked node.

∎

12.2 Reset rule-2

In this subsection we prove that if we use **reset rule-2** then the BGP converges in $(d+4)h$ time. In **reset rule-2**, we use the *Ghost flushing rule* and *minRouteAdver* to set $(d+4)h$, and a withdrawal message also set to the *minRouteAdver* timer.

Lemma 12.2 *The converge time complexity of* ***reset rule-2*** *follows* E_{down} *is* $(d+4)\cdot h$ *seconds.*

We first prove in the following Claim 7 that a ghost announcement reaches *ASpath* of maximum length of $d+4$ before reaching a blocked node. From Claim 1 of Lemma 5.1, after $(d+4)h$ time all the announcements with *ASpath* with length equal or less than $d+4$ disappear. And the lemma follows. ∎

Observation 6 *Using* ***reset rule-2****, if a node changes its ASpath, then the node is blocked.*

Notice that with the modification of *minRouteAdver* any change of a *ASpath* node, blocks the node. While in the original *minRouteAdver*, a rule a change to a NULL *ASpath* would not block the node. (It has no way to reach its destination.)

Claim 7 *Ghost information can have maximum ASpath of length of* $d+4$ *without reaching a blocked node.*

Sketch of proof:

The proof is similar to the proof of Claim 5, but here a ghost announcement can traverse only down the routing tree after passing the cross edge. This is due to the fact that if the message m traverses the edge (u,v), then the *ASpath* of v was changed. This can happen only if the *ASpath* of v has changed due to the change in the *ASpath* of v parent. Hence, v parent sent an announcement or withdrawal, and is blocked.

Let us look at the last cross link (u,v) unblock that the announcement has transfered, and assume v level is k. If such a cross link exists, the *ASpath* of the ghost announcement after traversing the cross link has a *ASpath* of maximum $k+3$ (from claim 4). After traversing the cross edge, the ghost announcement can have a maximum traverse length down the induced routing tree, for $d-k$ hops, until traversing a cross link whose end point is blocked (1 more). I.e., maximum length of *ASpath* $d+4$.

If no such cross link exist: Let's consider the source of the ghost announcement at some node v of level k. The source of the ghost message may be with maximum *ASpath* of length $k+2$ (since the maximum *ASpath* can occur when a node v stores the *ASpath* according to its peer at level $k+1$), and the ghost message may have a maximum traverse lenth down the

routing tree for $(d-k)$ steps and and then traverse a cross link that ended in a blocked node (1 hop) more). I.e., maximum length of *ASpath* $d+3$.

Hence, a ghost announcement has a maximum *ASpath* of length $d+4$ before the ghost announcement reaches a blocked node.

∎

12.3 Reset rule-3

In this subsection we prove that if we use the **reset rule-3** algorithm, then the BGP converges in $(d+4)h$ time. In **reset rule-3** we use the *Ghost flushing rule* and the *minRouteAdver* is set to $4h$. The **reset rule-3** applys only on a synchronous model, where the withdrawal message sets the *minRouteAdver* timer.

Lemma 12.3 *The convergence time complexity of* ***reset rule-3*** *that follows* E_{down} *is* $(d+4)h$ *seconds.*

The result is straight from part 5 of the following Claim 8. ∎

Claim 8 *At round n*

1. *Nodes of level n changes their ASpath for the first time. The maximum ASpath length is* $n+2$*. The nodes send a message (announcements with maximum length of* $n+3$ */or withdrawal) about the change in their ASpath. After sending the message the nodes are blocked until round* $n+4$*.*
2. *Nodes of level* $n-1$ *have ASpath with maximum lengths of* $n+3$*.*
3. *Nodes of level* $n-2$ *have ASpath with a maximum length of* $n+2$
4. *Nodes of level* $n-3$ *have ASpath with a maximum length of* $n+1$*.*
5. *Nodes of level* $n-4$ *change their ASpath to NULL ASpath, i.e., no path to destination. The nodes send withdrawal message to their neighbors, if their last message to their neighbors was not a withdrawal message.*
6. *Nodes with a level less than* $n-5$*, have a NULL, ASpath, and do not send any message.*

Proof: By induction on time. Let $n=1$. We need only to prove part 1 of the claim. All the nodes of level 1 change their *ASpath* due to the flushing rule and claim 1. Their new *ASpath* has a maximum length of 3 (since, their neighbors are of level 2, which announce their maximum *ASpath* of length 3). The nodes of level 1 sends a message (announcement or

withdrawal) regarding the change of *ASpath*, and hence become blocked until round 4, that is, until the *minRouteAdver* timer would have elapsed.
Let the claim hold until round $n-1$, we nprove that the claim holds also for round n.

Proof of part 1 of the claim: Let us look at nodes of level n. It easy to see that from variations on claim 1 for the synchronous case, at round n, the nodes of level n change their *ASpath* for the first time. The new *ASpath* can be changed based of the last announced *ASpath* of their three types of neighbors: (1) neighbors of level $n-1$, that from the inductive step announce a *ASpath* of maximum length $n+2$ or (2) from neighbors of level $n+1$, that their last announced *ASpath* has maximum length of $n+2$, since they still have their original *ASpath*. (3) or from neighbors of level n whose last announcement was *ASpath* of length $n+1$, since they still have their original *ASpath*. Hence, their new *ASpath* is of maximum length of $n+2$. The nodes announce their change, and hence become blocked until $n+4$.

Proof of part 2 of the claim: Let us look at nodes of the level $n-1$. Their *ASpath* can be changed based on the last announced *ASpath* of their three kinds of neighbors: (1) Neighbors of level $n-2$, that from the induction claim part 1, their last message was sent at round $n-2$ with *ASpath* of maximum length of $n+1$, (2) Neighbors of level $n-1$ that from the induction claim part 1 of their last announced *ASpath* was of length $n+2$ or (3) Neighbors of level n that their last announced *ASpath* was of length $n+1$ (since they still did not change their original *ASpath*). Hence, the *ASpath* of nodes of level $n-1$ may change to *ASpath* with a maximum length of $n+2$. However, notice that since this nodes are blocked (from induction claim part 1), they would not send any message to inform of the change to the neighbors.

Proof of part 3 of the claim: Let us look at nodes of level $n-2$. The nodes *ASpath* can be changed based on the last announced *ASpath* of their three kinds of neighbors: (1) Nodes of level $n-3$ that from the induction claim part 1, their last message was sent at round $n-3$ with *ASpath* of a maximum length of n, (2) Neighbors of level $n-2$ that from the induction claim part 1 their last announced *ASpath* was of length $n+1$ or (3) Neighbors of level $n-1$, whose last announced *ASpath* was of length $n+2$ from the induction claim part 1. Hence, the *ASpath* of nodes of level $n-1$ may change to *ASpath* with a maximum length of $n+2$. However, notice that since these nodes are blocked (from induction claim part 2), they would not send any message to inform of the change to the neighbors.

Proof of part 4 of the claim: Let us look at nodes of level $n-3$. From the induction claim 3, they had *ASpath* of $n+1$ in round $n-1$. In round n the nodes of level $n-3$ did not receive any new *ASpath* (since the nodes of level $n-4$,$n-3$ and $n-2$ are in a blocked state from the induction claim part 1).

Proof of part 5 of the claim: Let us look at the nodes of level $n-4$. From the induction claim part 4, their *ASpath* at round $n-1$ is of the maximum length of n. The nodes did not receive any new message, due to the fact that at round $n-1$, the nodes of level $n-3$,$n-4$ did not send any new messages (from induction claim part 1), and nodes of

level $n-5$ did not send any announcements (from induction claim 5 they could only send a withdrawal). From claim 1 at round n, there are no nodes with $ASpath$ of length n. However, the nodes of level n cannot change to $ASpath$ other then NULL, since the last announcements received by them were with $ASpath$ of maximum length n. Hence, the nodes of level n change their $ASpath$ to NULL and send withdrawal if the last message sent by them was different from the withdrawal.

Proof of part 6 of the claim: Let us look at nodes of the level $n-5$ or lower. The claim follows from the fact that the nodes do not receive any new announcement and hence do not change their state from the previous state which was $NULL$ and do not send any new message. To prove that, let us look at round $n-1$: From the induction step part 6, nodes of a level lower than $n-5$ did not send any new messages during this round. From the induction step part 5 nodes of level $n-5$ did not send any announcements (maybe they sent a withdrawal), and nodes of level $n-4$ did not send any new announcements due to the induction step part 1.