A Stability-oriented Approach to Improving BGP Convergence *

Hongwei Zhang Anish Arora Zhijun Liu
Department of Computer Science and Engineering
The Ohio State University, USA
{zhangho, anish, liuzh}@cse.ohio-state.edu

Abstract

This paper shows that the elimination of fault-agnostic instability, the instability caused by fault-agnostic distributed control, substantially improves BGP convergence speed. To this end, we first classify BGP convergence instability into two categories: fault-agnostic instability *and* distribution-inherent instability*; secondly, we prove the impossibility of eliminating all distribution-inherent instability in distributed routing protocols; thirdly, we design the Grapevine Border Gateway Protocol (G-BGP) to show that all fault-agnostic instability can be eliminated. G-BGP eliminates all fault-agnostic instability under different fault and routing policy scenarios by (i) piggybacking onto BGP UPDATE messages fine-grained information about faults to the nodes affected by the faults, (ii) quickly resolving the uncertainty between link and node failure as well as the uncertainty of whether a node has changed route, and (iii) rejecting obsolete fault information.*

We evaluate G-BGP by both analysis and simulation. Analytically, we prove that, by eliminating fault-agnostic instability, G-BGP achieves optimal convergence speed in several scenarios where BGP convergence is severely delayed (e.g., when a node or a link fail-stops), and when the shortest-path-first policy is used, G-BGP asymptotically improves BGP convergence speed except in scenarios where BGP convergence speed is already optimal (e.g., when a node or a link joins). By simulating networks with up to 115 autonomous systems, we observe that G-BGP improves BGP convergence stability and speed by an order of magnitude.

Keywords*: BGP, fault-agnostic instability, distribution-inherent instability, convergence speed, path-vector routing*

* An extended abstract containing some preliminary results of this paper appeared in the 23rd IEEE Symposium on Reliable Distributed Systems. This work was sponsored by DARPA contract OSU-RF #F33615-01-C-1901, NSF grant NSF-CCR-034170, and two grants from Microsoft Research.

1. Introduction

The Border Gateway Protocol (BGP) is used to coordinate routing among autonomous systems (simply called ASes hereafter) in the Internet [14]. Theoretically, BGP does not guarantee convergence and allows persistent route oscillations in several scenarios, such as conflicting routing policies and improper IBGP configurations [3, 11, 12]. This issue has been extensively studied, and a variety of approaches have been proposed [3, 9, 11, 12].

Our problem of interest, therefore, is the scenario where BGP does converge, but its convergence exhibits instability (i.e., allowing unnecessary route changes) and is potentially slow (e.g., taking up to 15 minutes after the disconnection of a single AS [15, 16]).

- Instability during convergence is undesirable. First, it increases the probability of message reordering, which is not only undesirable for multimedia applications, but also degenerates the performance of other protocols such as TCP [5]. Second, instability increases packet loss (e.g., due to TTL expiration) [15]. And finally, instability increases delay jitter in packet delivery.
- Slow convergence of BGP is also undesirable. It not only deteriorates packet delivery, but also amplifies the effect of BGP-related problems under stressful conditions such as the Code Red/Nimda attack [23].
- Moreover, the two sub-problems are related. The interaction of unstable BGP convergence and the BGP route flap damping, for instance, can delay BGP convergence further, in addition to entailing loss of reachability for hours [18].

Related work. To improve BGP convergence, the methods of "consistency assertion" [21], "ghost flushing" [6], and "route change origin" [17] have been proposed in the literature. In parallel with our work, the methods of "root cause notification" [20] and EPIC [7] have also been proposed. Consistency assertion rejects inconsistent routes; ghost flushing withdraws old routes faster than propagating new routes; route change origin and root cause notification propagate the ID of the node that first withdraws or

changes route; EPIC propagates *path-stem*, based on forward edge sequence numbering, to identify certain invalid routes. Nevertheless, the nature of different types of instability during BGP convergence, the fundamental limits on improving BGP convergence stability and speed, and the impact of different faults and routing policies on protocol convergence behaviors are not the focus of the above works.

Moreover, none of the above methods completely eliminate fault-agnostic instability, the major cause for slow BGP convergence (to be discussed in detail in Section 3.1). Consistency assertion does not address the type of slow BGP convergence that is due to inconsistency between nodes multiple hops apart. Ghost flushing does not guarantee that an invalid route is not used by a node, even if some implicit information regarding the invalidity of the route has reached the node. Finally, route change origin, root cause notification, and EPIC do not guarantee that an invalid route will not be used when a node with multiple neighbors fail-stops.

Additionally, consistency assertion and root cause notification propagate the entry-router-ids of routes from one AS to others. Yet the entry-router-id is essentially an attribute below the AS level, and as a result of its propagation, every local change of an entry-router in an AS propagates to other (potentially distant) ASes. This propagation of local changes can happen even when the corresponding AS-path does not change, which incurs extra instability beyond what exists in BGP.

Contributions of the paper. We study the nature of instability during BGP convergence and classify the instability into two categories: fault-agnostic instability and distribution-inherent instability. Fault-agnostic instability is the major cause for slow BGP convergence, and distribution-inherent instability is intrinsic to distributed protocols. To understand the fundamental limit on improving BGP convergence, we prove that it is impossible to eliminate all distribution-inherent instability in stateful distributed routing protocols.

Then, we refine BGP to obtain a new protocol G-BGP (for *Grapevine-BGP*) that eliminates all fault-agnostic instability. We find that, by eliminating fault-agnostic instability, G-BGP converges at an asymptotically optimal speed in several scenarios where BGP convergence is severely delayed (e.g., when a node or a link fail-stops), and when the shortest-path-first policy is used, G-BGP asymptotically improves BGP convergence speed except in scenarios where BGP convergence speed is already optimal (e.g., when a node or a link joins).

We have also evaluated G-BGP by simulation with realistic Internet-type network topologies. The simulation shows that, for networks with up to 115 ASes, G-BGP improves BGP convergence stability and speed by factors of 29.4 and 10.2 respectively, and the improvement in G-BGP increases as network size increases. The simulation also shows that, when routing policies other than "shortest-path-first" are used, G-BGP improves BGP convergence stability and speed in all fault scenarios.
(Due to the limitation of space, we relegate the proofs of the theorems in this paper to a technical report [24].)

Organization of the paper. In Section 2, we present the network as well as the fault model, and we briefly describe BGP. In Section 3, we study the nature of BGP convergence instability and present the G-BGP design. In Section 4, we introduce policy graphs and use the technique to analyze the convergence behaviors of G-BGP as well as BGP. In Section 5, we present our simulation results. In Section 6, we discuss implementation as well as deployment issues for G-BGP, and we discuss approaches to reducing distribution-inherent instability. Section 7 concludes the paper.

2. Preliminaries

In this section, we present the network as well as the fault model. We also briefly describe the Border Gateway Protocol (BGP).

Network model. A network G is an undirected graph (V, E, P), where V and E are the set of nodes (i.e., BGP speakers) and the set of links in the network respectively, and P is the function that defines the routing policies of each node. V is divided into several subsets, each of which is an AS; nodes within the same AS are connected (AS partition is discussed in Section 6). Each node has a unique node-id, and all the nodes in the same AS have the same AS-id. For a node i, the id of its AS is denoted by $i.AS$. For any two nodes i and j, (i, j) is in E if i and j can communicate with each other directly, or if i and j are in the same AS. For any two ASes $\mathcal{I}$ and $\mathcal{J}$, there is a *channel* $(\mathcal{I}, \mathcal{J})$ between $\mathcal{I}$ and $\mathcal{J}$ if there exist two nodes i and j such that $i \in \mathcal{I}$, $j \in \mathcal{J}$, and $(i, j) \in E$. For an AS $\mathcal{I}$ and a node j, $\mathcal{I}$ is a neighboring AS of j if there is a channel between $\mathcal{I}$ and $j.AS$. For a node i, its neighboring node j is an *internal neighbor* if j is in the same AS as i; otherwise, j is an *external neighbor* of i.

There is a clock at each node. The ratio of clock rates between any two nodes is bounded from above by α, but no extra constraint on the absolute values of clocks is enforced. (α tends to be quite small, given today's high-precision clocks.) Message transmission between nodes is reliable, and message passing delay across a link is bounded from above by U_d.

For clarity of presentation, we only consider one destination d, an address prefix representing a set of nodes in an AS $d.AS$. (Our protocol readily applies to other destinations.)

Fault model. A node or a link is *up* if it functions correctly, and it is *down* if it fail-stops. In a network, an up node or link

can fail-stop and become down; a down node or link can become up and join the network; routing policies of ASes can change. A channel $(\mathcal{I}, \mathcal{J})$ is up if there is at least one up link between ASes $\mathcal{I}$ and $\mathcal{J}$; otherwise, the channel is down. An AS is up if there is at least one up node in the AS; otherwise, the AS is down.

The fail-stop of a node is divided into two categories: *graceful fail-stop* where a node announces to its neighbors when it fail-stops, and *gross fail-stop* where a node fail-stops silently. An AS fail-stops gracefully if all the nodes in it fail-stop gracefully. (One example of graceful fail-stop is when the destination d withdraws its address prefix.)

Due to faults, a network G may change in the sense that its topology or routing policy function changes, where the topology of G is the subgraph $G'(V', E')$ of $G(V, E)$ such that $V' = \{i : i \in V \wedge i \text{ is up}\}$ and $E' = \{(i, j) : i \in V' \wedge j \in V' \wedge (i, j) \in E \wedge (i, j) \text{ is up}\}$. To reflect changes in network topology and routing policy function, we regard the state of G as the union of the network topology, the routing policy function, and the state of all the up nodes, with the state of a node being the values of the variables maintained at the node. At a network state q, the network topology and the route of a node i are denoted by $G.q(V.q, E.q)$ and $i.AS\text{-}path.q$ respectively.

Border Gateway Protocol (BGP). In BGP, UPDATE messages are passed between nodes to convey routing information. To reduce instability, BGP employs a MRAI timer (which is 30 seconds by default) such that a node sends out at most one non-withdrawal UPDATE message within any MRAI time. Two neighboring nodes also periodically exchange Keep-Alive messages to monitor the state of the link between them.

BGP UPDATEs are route records that include the following attributes (among others):

$nlri$	:	network layer reachability information;
$next\_hop$	:	the next hop;
AS_path	:	ordered list of ASes traversed, with more-recently-visited ASes placed in front of less-recently-visited ASes;
$local\_pref$	:	local preference;
med	:	multi-exit discriminator.

Each route r is associated with a 3-tuple $rank(r)$, defined as $\langle r.local\_pref, \frac{1}{|r.as\_path|}, \frac{1}{r.next\_hop} \rangle$. For the destination d, a node i chooses its route via the following two steps [22]:

- First, among all the routes learned from a neighboring AS, i only considers the route with the lowest *med* value;
- Second, for all the routes to be considered, i ranks them in lexical order by $rank(\cdot)$, and i selects as its route the one with the highest rank.

Given a route r available to a node i, attribute $r.local\_pref$ is determined by the *route ranking policy* of i. We call the ranking policy that assigns $r.local\_pref$ to a constant value the *shortest-path-first policy* or the *SPF policy*, where a route with the shortest AS-path ranks the highest. We call a route ranking policy other than the SPF policy a *non-SPF policy*.

Besides route ranking policy, routing policies such as export and import policies are used in BGP. The export policy of a node i defines the set of *export neighbors of* i to which i announces its route; the import policy of i defines the set of *import neighbors of* i whose routes are accepted by i. If a node i exports routes to or imports routes from a node in a neighboring AS $\mathcal{J}$, we say, for convenience, i exports routes to or imports routes from $\mathcal{J}$ respectively. It is recommended as well as the common-practice that nodes within the same AS share the same routing policies [13, 22].

3. Protocol G-BGP

The objective of this paper is to design a protocol that, given a network and a destination where BGP converges in the presence of faults, reduces the number of route changes during BGP convergence, as well as the time taken for BGP to converge. To this end, we first study the nature of BGP convergence instability and its relationship to BGP convergence speed; we then design protocol G-BGP to improve BGP convergence stability and speed.

3.1. Instability during BGP convergence

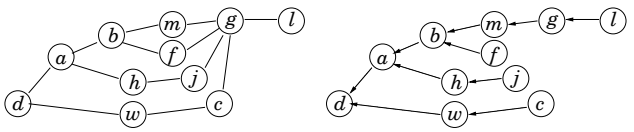
We identify fault-agnostic instability and distribution-inherent instability, analyze their causes, and discuss their relationship with BGP convergence speed.

Fault-agnostic instability. Fault-agnostic instability is the type of instability that is incurred at a node which adopts an invalid route even though some information regarding the fault that invalidates the route has reached the node. Fault-agnostic instability and its propagation are the major causes for slow BGP convergence, as observed in [15], [18], etc..

In BGP, when a fault occurs in a network, certain coarse-grained information about the result of the fault, such as an UPDATE message signaling a modified or a withdrawn route, is propagated so that the network eventually converges to a stable state. However, the coarse-grained information does not tell what exactly the fault is or where the resulting route changes first occurred. Therefore, when a node receives the coarse-grained information, the node may still adopt a route invalidated by the fault, in which case unnecessary route changes (i.e., instability) is incurred. The instability incurred at a node can propagate to others and delay the convergence of BGP. Even worse, instability can activate route-flap damping, which suppresses routes going through

unstable nodes, leading to a loss of reachability as well as a delay in BGP convergence (potentially for hours) [18].

To give an example, let us consider a network state q where the network topology and the routing tree rooted at d are shown in Figures 1(a) and 1(b) respectively; Suppose



(a) Network topology (b) Routing tree

Figure 1. An example network state. For simplicity, each node in the figure also represents its AS.

the route ranking policy is such that, for the three backup routes of g at state q, route $[f, b, a, d]$ ranks the highest, followed by $[j, h, a, d]$, and $[c, w, d]$ ranks the lowest. If a fail-stops at state q, nodes m, f, and j will withdraw their routes $[m, b, a, d]$, $[f, b, a, d]$, and $[j, h, a, d]$ respectively. However, the resulting route-withdrawal UPDATE messages do not signal the fact that a and its associated links have fail-stopped. Therefore, if f has not withdrawn $[f, b, a, d]$ when m withdraws $[m, b, a, d]$ (due to different delays along the two routes), g will adopt $[f, b, a, d]$ as its route even though $[f, b, a, d]$ has been invalidated by the fail-stop of a. Similarly, if j has not withdrawn $[j, h, a, d]$ when f withdraws $[f, b, a, d]$ later, g will adopt $[j, h, a, d]$ even though it has also been invalidated by the fail-stop of a. g will not change to its final stable route $[c, w, d]$ until j withdraws $[j, h, a, d]$. Therefore, g changes route three times during convergence, with the first two changes being unnecessary. Even worse, the unnecessary route changes at g can propagate and cause unnecessary route changes at other nodes, such as m, l, f, and b, when g announces $[g, f, b, a, d]$ or $[g, j, h, a, d]$ to its export neighbors.

Distribution-inherent instability. Distribution-inherent instability is the type of instability that is incurred (i) at a node which adopts an invalid route because no information regarding the fault that invalidates the route has reached the node, or (ii) at a node which adopts a valid route that becomes either invalid or lower-ranked than some other route later.

To give an example of type-(i) distribution-inherent instability, let us consider again the network state q as shown in Figure 1. If node b and link (a, h) fail-stop simultaneously, and if m as well as f withdraws its route earlier than j does, then no information that is generated due to the fail-stop of (a, h) will have reached g when it receives the route-withdrawal messages from m and f. Therefore, g will choose $[j, h, a, d]$ as its new route, which will be withdrawn later. Thus, an unnecessary route change is incurred at g before it chooses its final stable route $[c, w, d]$.

To give an example of type-(ii) distribution-inherent instability, let us consider a network state where the network topology is the same as that in Figure 1(a) but no node has learned any route to d. Then d will announce its existence, and other nodes such as a, b, m, and f will learn their routes gradually. If f exports its route to g earlier than m does, g will choose $[f, b, a, d]$ as its route first. When m exports $[m, b, a, d]$ to g later, g will change its route to $[m, b, a, d]$, since $[m, b, a, d]$ ranks higher than $[f, b, a, d]$ does. Thus, there is an unnecessary route change at g, even though the route $[f, b, a, d]$ adopted by g is valid.

Unlike fault-agnostic instability, distribution-inherent instability does not cause long delay in BGP convergence (as observed in Section 5 and in [18]). Moreover, distribution-inherent instability exists in every distributed routing protocol, as proved in

Proposition 1 *In a network, if message passing delay along links is greater than zero, route ranking policies are not shared among ASes, and faults are independent of one another, then it is impossible to eliminate all distribution-inherent instability in any stateful distributed routing protocol.* ■

Therefore, we focus on the mechanisms as well as the impact of eliminating fault-agnostic instability; we only briefly discuss approaches to reducing distribution-inherent instability Section 6.

3.2. The design of G-BGP

To eliminate fault-agnostic instability, we develop protocol G-BGP that refines BGP with the following mechanisms:

Propagating information about faults. In BGP, fault-agnostic instability is incurred at a node which adopts an invalid route due to the lack of fine-grained information about faults. Therefore, in G-BGP, necessary fine-grained fault information is propagated when a fault occurs; when the affected nodes receive the fault information, they are able to learn the fault or its impact and avoid using any route that is invalidated by the fault. On the other hand, fault information is propagated only if the corresponding fault invalidates the existing route of some node, and fault information is either not stored or only temporarily stored at a node for a bounded time.

When a fault occurs, the information being propagated depends on the type of the fault. In general, as a result of the fault, one or more nodes in the network may change

their next-hops in forwarding traffic, in which case necessary *points of channel-withdrawal* or *points of segment-withdrawal* are propagated to the affected nodes to reflect the fact that certain channels or route segments are not used in forwarding traffic any more. In the case when the destination withdraws its address prefix, when the channel between two ASes fail-stops, or when a node joins the network, a *point of destination-withdrawal*, a *point of channel-failure*, or a *point of node-join* is also propagated respectively.

Localized uncertainty resolution. When a node i detects that a link (i, j) has fail-stopped with the existing fault detection mechanisms in BGP (e.g., neighboring nodes periodically exchange Keep-Alive messages), i cannot ascertain whether its neighbor j and the neighboring AS $j.AS$ are up or down. This uncertainty, if left unresolved, can lead to fault-agnostic instability. For example, at the network state q as shown in Figure 1, if a fail-stops, b can only ascertain that link (b, a) has fail-stopped, but b cannot ascertain whether a is down. Therefore, only the point of channel-failure information signaling the fail-stop of (b, a) is propagated to g; thus g only knows that (b, a) has fail-stopped, but g is uncertain whether a is still up and whether $[j, h, a, d]$ is valid. In BGP, g adopts $[j, h, a, d]$ by "assuming without proof" that it is valid, which results in fault-agnostic instability.

Similarly, when a node i receives a point of segment-withdrawal or a point of node-join information which signals that nodes in an AS $\mathcal{J}$ other than $i.AS$ may have changed routes, fault-agnostic instability can occur, if i adopts a route going through $\mathcal{J}$ by simply assuming (without proof) that nodes in $\mathcal{J}$ have not changed routes.

To avoid fault-agnostic instability caused by the uncertainty regarding the state of an AS or a route, G-BGP resolves the uncertainty (when need be) by gathering proof of the state of the suspected AS or route. To expedite potential uncertainty resolution operation, G-BGP uses the mechanisms of "quickly marking questionable routes" and "collaboratively clarifying state". Moreover, uncertainty resolution in G-BGP is local in the sense that, usually, only nodes close to the suspected AS need to resolve the uncertainty, but nodes farther away do not, which is the case especially in highly connected networks such as the Internet [8].

Rejecting obsolete fault information. In a network, message passing and processing delay along different routes may differ, thus a fresher message containing some fault information regarding an AS may reach a node earlier than a staler message containing some obsolete fault information regarding the AS. This can lead to fault-agnostic instability, if the obsolete fault information is used. To avoid using obsolete fault information, G-BGP verifies the freshness of each piece of fault information upon receipt, which is enabled by enforcing a total order (via local sequence number) on all the fault information regarding the same AS that is sent out from the AS at different time.

We elaborate on the above mechanisms in the following subsections. (Due to the limitation of space, the formal presentation of G-BGP in the Abstract Protocol notation [10] is relegated to [24].)

3.2.1. Propagating fault information Towards enabling nodes to generate appropriate fault information in the presence of faults, the intra-AS coordination in BGP is enhanced as follows: (see Section 6 for the implementation): each node i informs the other nodes in its AS of the route of i itself, the neighboring ASes to which i has exported its route, and the neighboring ASes to which i is connected via an up-link. By the enhanced intra-AS coordination, every node j can decide (i) whether there is another node in $j.AS$ whose route goes through the same neighboring AS as j, (ii) whether there is another node in $j.AS$ that has exported route to some neighboring AS which j has exported route to, and (iii) whether the channel to a neighboring AS is up.

Then, necessary fault information is generated as follows in the presence of faults.

Point of channel-withdrawal. When a node i changes from a route $R = [\mathcal{J}, \ldots, d.AS]$ to another non-empty one $[\mathcal{J}', \ldots, d.AS]$ with $\mathcal{J}' \neq \mathcal{J}$, i will not use any link between ASes $i.AS$ and $\mathcal{J}$ in forwarding traffic to d. In this case, if R is still valid, but no link between $\mathcal{J}$ and $i.AS$ is used by any node in $i.AS$,[1] i will generate a *point of channel-withdrawal* $\langle[i.AS, \mathcal{J}]\rangle$, unless i has received it from some other node, to signal the fact that every route going through route segment $[i.AS, \mathcal{J}]$ has become invalid. For convenience, we call $(i.AS, \mathcal{J})$ a withdrawn-channel.

Special cases are when an AS changes its import or export policy. When an AS $\mathcal{I}$ changes its import policy such that nodes in $\mathcal{I}$ should not import routes from a set $\mathbb{S}$ of neighboring ASes, a node i in $\mathcal{I}$ should generate the set of points of channel-withdrawal $\{\langle[\mathcal{I}, \mathcal{K}]\rangle : \mathcal{K} \in \mathbb{S}\}$, unless i has received it from some other nodes. Similarly, when $\mathcal{I}$ changes its export policy such that nodes in $\mathcal{I}$ should not export routes to a set $\mathbb{S}'$ of neighboring ASes, a node i in $\mathcal{I}$ should generate and send the set of points of channel-withdrawal $\{\langle[\mathcal{K}', \mathcal{I}]\rangle : \mathcal{K}' \in \mathbb{S}'\}$ to its external neighbors, if any, to which i has exported its route.

Point of segment-withdrawal. If some node in $i.AS$ is still using a link between $\mathcal{J}$ and $i.AS$ (in forwarding traffic to the destination d) when a node i changes from a valid route $R = [\mathcal{J}, \ldots, d.AS]$ to another non-empty one $[\mathcal{J}', \ldots, d.AS]$ with $\mathcal{J}' \neq \mathcal{J}$, i should not generate the point of channel-withdrawal $\langle[i.AS, \mathcal{J}]\rangle$; otherwise, valid routes can be mistakenly regarded as invalid. In this case, i calculates the set $\mathbb{S}$ of its neighboring ASes such that, for ev-

1 This is detected via synchronization among nodes in an AS via the enhanced intra-AS coordination (to be discussed in Section 6).

ery $\mathcal{K} \in \mathbb{S}$, i has exported R to $\mathcal{K}$, but there is no node i' in $i.AS$ where $i' \neq i$, and such that i' has exported its route to $\mathcal{K}$ and i' is still using some link between $\mathcal{J}$ and $i.AS$ (in forwarding traffic to d). i also calculates the set $\mathbb{S}'$ of its neighboring ASes such that, for every $\mathcal{K}' \in \mathbb{S}'$, i has exported R to $\mathcal{K}'$, and there is at least one node i'' in $i.AS$ where $i'' \neq i$, and such that i'' has exported its route to $\mathcal{K}'$ and i'' is still using some link between $\mathcal{J}$ and $i.AS$ (in forwarding traffic to d).

Then, for every AS $\mathcal{K} \in \mathbb{S}$, route segment $[\mathcal{K}, i.AS, \mathcal{J}]$ will not be used by any node in $\mathcal{K}$ after i exports its new route to $\mathcal{K}$, thus every route going through $[\mathcal{K}, i.AS, \mathcal{J}]$ becomes invalid; However, for an AS $\mathcal{K}' \in \mathbb{S}'$, some node in $\mathcal{K}'$ may still use and some other node in $\mathcal{K}'$ may stop using route segment $[\mathcal{K}', i.AS, \mathcal{J}]$ after i exports its new route to $\mathcal{K}'$, thus i is uncertain about the validity of routes going through $[\mathcal{K}', i.AS, \mathcal{J}]$. To signal the above fact when $\mathbb{S} \neq \emptyset$ or $\mathbb{S}' \neq \emptyset$, i generates a *point of segment-withdrawal* $\langle \mathbb{S}, \mathbb{S}', i.AS, \mathcal{J}, i, t \rangle$, where t is the time passed since i changes its route (t is 0 initially and increases as the point of segment-withdrawal is propagated from one node to another). If $\mathbb{S}' \neq \emptyset$, the uncertainty regarding the validity of routes that go through $[\mathcal{K}', i.AS, \mathcal{J}]$ for some $\mathcal{K}' \in \mathbb{S}'$ is resolved, if need be, later at nodes close to $\mathcal{K}'$ (to be discussed in Section 3.2.2). For convenience, we call $[\mathcal{K}, i.AS, \mathcal{J}]$ a withdrawn-segment for every $\mathcal{K} \in \mathbb{S}$; for every $\mathcal{K}' \in \mathbb{S}'$, we call $[\mathcal{K}', i.AS, \mathcal{J}]$ a questionable segment and $\mathcal{K}'$ a questionable AS.

Point of destination-withdrawal. When the destination d withdraws its address prefix, all the routes to d become invalid. To signal this fact, a *point of destination-withdrawal* $\langle d \rangle$ is generated.

Point of channel-failure. When a node i with route $[\mathcal{J}, \ldots, d.AS]$ detects that its link to AS $\mathcal{J}$ has fail-stopped, i will check if the channel between $i.AS$ and $\mathcal{J}$ is up. If the channel is down, i knows that every route going through route segment $[i.AS, \mathcal{J}]$ becomes invalid. However, i is uncertain whether its external neighbor(s) in $\mathcal{J}$ is(are) up or down; thus i is uncertain whether the other channels associated with $\mathcal{J}$ are up or down. To signal the above fact, i generates a *point of channel-failure* $\langle [i.AS, \mathcal{J}], t \rangle$, where t is the time passed since the channel-failure is detected. The uncertainty regarding the state of $\mathcal{J}$ and its associated channels is resolved, if need be, later at nodes close to $\mathcal{J}$. For convenience, we call $\mathcal{J}$ a questionable AS.

Point of node-join. When a node i joins a network and exports its route to a set $\mathbb{S}$ of neighboring ASes, nodes in those ASes may change their routes to those going through $i.AS$, which is, however, uncertain to i. To signal the above fact, i generates a *point of node-join* $\langle i.AS, \mathbb{S}, i, t \rangle$, where t is the time passed since i joins the network. The uncertainty regarding whether nodes in an AS $\mathcal{K}$ in $\mathbb{S}$ have changed their routes to go through $i.AS$ is resolved, if need be, later at nodes close to $\mathcal{K}$. For convenience, we call every $\mathcal{K}$ in $\mathbb{S}$ a questionable AS.

How G-BGP uses and propagates fault information in a bounded manner. When a fault occurs, some node close to where the fault has occurred will generate, if need be, the corresponding fault information. The newly generated fault information, if any, is piggybacked onto the UPDATE messages that the node sends to its export neighbors.

When a node i receives an UPDATE message piggybacked with some fresh fault information, i first modifies the information, if need be, as follows:

- i changes every point of segment-withdrawal $\langle \mathbb{S}, \mathbb{S}', \mathcal{I}, \mathcal{J}, i', t \rangle$ where $i.AS \in \mathbb{S}'$ and every point of node-join $\langle \mathcal{K}, \mathbb{S}'', k', t \rangle$ where $i.AS \in \mathbb{S}''$, if any, by removing $i.AS$ from $\mathbb{S}'$ and $\mathbb{S}''$ respectively, since i is sure about the state of its own AS (i.e., $i.AS$).
- i removes every point of channel-withdrawal $\langle [\mathcal{J}, i.AS] \rangle$, every point of segment-withdrawal $\langle \mathbb{S}, \mathbb{S}', \mathcal{J}, i.AS, j', t \rangle$, the point of AS-failure $\langle i.AS \rangle$, every point of channel-failure $\langle [\mathcal{J}, i.AS], t \rangle$, and every point of node-join $\langle i.AS, \mathbb{S}, i', t \rangle$, if any, since i will not choose any route that goes through its own AS.

  i also removes every point of segment-withdrawal $\langle \mathbb{S}, \mathbb{S}', i.AS, \mathcal{J}, i, t \rangle$, if any, that is generated by i itself.

Then, i invalidates all the routes for each withdrawn address prefix, if any; i also invalidates and avoids using the routes that go through any withdrawn channel, withdrawn segment, and/or fail-stopped channel. Moreover, if the highest ranked candidate route R goes through some questionable AS, i will not choose R unless i does not invalidate R after i resolves the associated uncertainty. If i changes route after processing the UPDATE message, i sends to its export neighbors an UPDATE message piggybacked with the fault information that i knows of, then i deletes the fault information without storing it. On the other hand, if i does not change route, i will not propagate any fault information to its external neighbors; instead, i only propagates its newly-learned fault information to its internal neighbors, to guarantee that nodes in the same AS have a consistent view of faults and their impact. Therefore, information about faults only propagates to the nodes, as well as their immediate neighbors, that change routes due to the faults; consequently, the propagation of fault information is bounded.

When a node j does not change route after receiving some fault information, j will store the fault information temporarily for up to U time,[2] where U is the upper bound

on the convergence time of BGP after a fault occurs.[3] If j does not change route within U time after it receives the fault information, j will delete it permanently, since any route that can be invalidated by the fault information should have been invalidated U time after j receives the fault information. (Of course, if j changes route within U time after receiving the fault information, the information will be deleted after being piggybacked onto the UPDATE messages that j sends out.)

When a node j stores some fault information, j updates the information as the network state changes, so that the stored information abides by the definition of each type of fault information. For example, for every set of points of segment-withdrawal $\{\langle \mathbb{S}_k, \mathbb{S}'_k, \mathcal{I}, \mathcal{L}, i', t_k\rangle : k \in 1..n, n > 1\}$, if any, that are stored at j, j integrates them into a single point of segment-withdrawal $\langle \cap_{k=1}^{n} \mathbb{S}_k, \cap_{k=1}^{n} \mathbb{S}'_k, \mathcal{I}, \mathcal{L}, i', \min_{k\in 1..n} t_k\rangle$.

- When the state of its AS $j.AS$ changes such that a node i in $j.AS$ uses a route going through segment $[j.AS, \mathcal{K}]$ for some $\mathcal{K}$ and i has exported the route to a set $\mathbb{S}''$ of neighboring ASes, j knows that channel $(j.AS, \mathcal{K})$ is up and used. Thus, j deletes the point of channel-withdrawal $\langle [j.AS, \mathcal{K}]\rangle$, the point of segment-withdrawal $\langle \mathbb{S}, \mathbb{S}', j.AS, \mathcal{K}, i, t\rangle$, and/or the point of channel-failure $\langle [j.AS, \mathcal{K}], t\rangle$, if they are stored at j; moreover, j changes every point of segment-withdrawal $\langle \mathbb{S}, \mathbb{S}', j.AS, \mathcal{K}, i', t\rangle$, if any, where $i' \neq i$ to $\langle \mathbb{S} \setminus \mathbb{S}'', \mathbb{S}', j.AS, \mathcal{K}, i', t\rangle$, and if $\mathbb{S} \setminus \mathbb{S}'' = \mathbb{S}' = \emptyset$ after the change, j deletes the corresponding information.
- When j receives an UPDATE message m that contains a route R and some fresh fault information regarding an AS $\mathcal{K}$,
  - if R goes through segment $[\mathcal{K}, \mathcal{K}']$ for some $\mathcal{K}'$, j deletes the point of channel-withdrawal $\langle [\mathcal{K}, \mathcal{K}']\rangle$, the point of channel-failure $\langle [\mathcal{K}, \mathcal{K}'], t\rangle$, and/or the point of AS-failure $\langle \mathcal{K}\rangle$, if they are stored at j;
  - if R goes through segment $[\mathcal{K}'', \mathcal{K}, \mathcal{K}']$ for some $\mathcal{K}''$ and $\mathcal{K}'$, j changes every point of segment-withdrawal $\langle \mathbb{S}, \mathbb{S}', \mathcal{K}, \mathcal{K}', k, t\rangle$ where $\mathcal{K}'' \in \mathbb{S}$, if any, by removing $\mathcal{K}''$ from $\mathbb{S}$, and if $\mathbb{S} = \mathbb{S}' = \emptyset$ after the change, j deletes the corresponding information.

  Then, j reliably informs other nodes in its AS of the changes.
- For every set of points of segment-withdrawal $\{\langle \mathbb{S}_k, \mathbb{S}'_k, \mathcal{I}, \mathcal{L}, i', t_k\rangle : k \in 1..n, n > 1\}$, if any, that are stored at j, j integrates them into a single point of segment-withdrawal $\langle \cap_{k=1}^{n} \mathbb{S}_k, \cap_{k=1}^{n} \mathbb{S}'_k, \mathcal{I}, \mathcal{L}, i', \min_{k\in 1..n} t_k\rangle$; similarly, j integrates every set of points of node-join $\{\langle \mathcal{K}, \mathbb{S}''_k, k', t_k\rangle : k \in 1..n, n > 1\}$, if any, into a single point of node-join $\langle \mathcal{K}, \cap_{k=1}^{n} \mathbb{S}''_k, k', \min_{k\in 1..n} t_k\rangle$.

  If j has a point of channel-withdrawal $\langle [\mathcal{L}, \mathcal{I}]\rangle$ and a point of channel-failure $\langle [\mathcal{L}, \mathcal{I}], t\rangle$ simultaneously (which can happen as a result of the uncertainty resolution regarding the state of $\mathcal{I}$), j deletes $\langle [\mathcal{L}, \mathcal{I}], t\rangle$, so that the validity of routes going through $\mathcal{I}$ will not be suspected due to the existence of $\langle [\mathcal{L}, \mathcal{I}], t\rangle$.

3.2.2. Localized uncertainty resolution

Notations:

$hops(\mathcal{I}, \mathcal{J}, R)$	:	the number of inter-AS hops between ASes $\mathcal{I}$ and $\mathcal{J}$ in a route $R = [\mathcal{I}, \ldots, \mathcal{J}, \ldots, d.AS]$;
T_m	:	the upper bound on the time taken to process a message in BGP.

To expedite and to enhance the locality of potential uncertainty resolution, G-BGP uses the mechanisms of "quickly marking questionable routes" and "collaboratively clarifying state".

Quickly marking questionable routes. To resolve uncertainty, a node needs to obtain proof information from others. However, information flow can be slow in BGP due to the use of MRAI timer. To expedite potential uncertainty resolution after a point of segment-withdrawal or a point of node-join is generated, *purging-messages* are sent, without subject to the MRAI timer control, along invalid routes that go through the corresponding questionable segment or AS. More specifically:

- A node i sends a purging-message to each of its export neighbors, when either of the following conditions holds: (i) i will change its route not to go through a segment $[\mathcal{J}, \mathcal{K}]$ after i receives a point of segment-withdrawal $\langle \mathbb{S}, \mathbb{S}', \mathcal{J}, \mathcal{K}, j', t\rangle$ where $i.AS \in \mathbb{S}'$; (ii) i will change its route to go through an AS $\mathcal{J}$ after i receives a point of node-join $\langle \mathcal{J}, \mathbb{S}, j', t\rangle$ where $i.AS \in \mathbb{S}$.
- When a node j receives a purging-message from an import neighbor i, j marks as invalid the candidate route imported from i; moreover, if the marked candidate route is the current route of j, j re-sends a purging-message to each of its export neighbors.
- When a node j marks its route as invalid, j does not change its route immediately; instead, j waits for the

2 Alternatively, we can assign each piece of fault information a lifetime of U and decrease its lifetime as time passes by. Then a node stores fault information with a lifetime of t' for at most $(U - t')$ time.

3 In practice, given that a node tends not to change route later on if it does not upon receiving certain related fault information, we could choose a much smaller U value to reduce the transient memory overhead. In our simulation, we find that it is good enough to set U as 100 seconds.

normal BGP procedure to stabilize its route later. On the other hand, j will remove the marking, if its route has been marked as invalid for U time without being withdrawn or changed (i.e., the route has become valid), where U is the upper bound on the convergence time of BGP after a fault occurs.

Therefore, if a node i sends a purging-message to its export neighbors, another node j that has a candidate route $R = [\mathcal{K}, \ldots, i.AS, \ldots, d.AS]$ will mark R as invalid within $(hops(\mathcal{K}, i.AS, R) + 1) \times (T_m + U_d)$ time. (Given the high-performance routers and high-speed networks in today's Internet, both T_m and U_d tend to be quite small.)

Collaboratively clarifying state. When a channel $(i.AS, \mathcal{J})$ used by a node i becomes down, i generates a point of channel-failure $\langle [i.AS, \mathcal{J}], t \rangle$ to signal, in addition to the fail-stop of $(i.AS, \mathcal{J})$, the uncertainty regarding the validity of routes going through some other channel associated with $\mathcal{J}$. In this case, only the up nodes in $\mathcal{J}$, if any, know the exact state (i.e., up or down) of the channels associated with $\mathcal{J}$. Therefore, the up nodes in $\mathcal{J}$ propagate, without subject to the $MRAI$ timer control, *state-clarifiers* regarding the channels that fail-stop simultaneously to the nodes whose candidate routes go through an up channel associated with $\mathcal{J}$. More specifically,

- When a node j detects that a channel $(j.AS, \mathcal{I})$ used by some node in a neighboring AS $\mathcal{I}$ fail-stops, j first calculates the set $\mathbb{S}$ of ASes such that, for every $\mathcal{I}' \in \mathbb{S}$, j detects the fail-stop of $(j.AS, \mathcal{I}')$ within T_f time before or after j detects the fail-stop of $(j.AS, \mathcal{I})$, where T_f is the delay in detecting the fail-stop of channels. (By definition, $\mathcal{I} \in \mathbb{S}$.) Then, j sends to its external neighbors the state-clarifier $\langle \mathbb{S}, j.AS \rangle$.
- When a node k receives a state-clarifier $\langle \mathbb{S}, \mathcal{J} \rangle$ from another node k', k stores $\langle \mathbb{S}, \mathcal{J} \rangle$, if the route of k goes through a segment $[\mathcal{I}', \mathcal{J}]$ for some $\mathcal{I}' \in \mathbb{S}$; otherwise, if the route of k is imported from k' and goes through a segment $[\mathcal{K}, \mathcal{J}]$ for some $\mathcal{K} \notin \mathbb{S}$, k first invalidates all of its candidate routes, if any, that go through a segment $[\mathcal{I}'', \mathcal{J}]$ for some $\mathcal{I}'' \in \mathbb{S}$, then k re-sends the state-clarifier to its export neighbors;
- A node deletes a stored state-clarifier, if it has been stored for U time without being used, where U is the upper bound on the convergence time of BGP after a fault occurs.

Therefore, if a node j sends a state-clarifier $\langle j.AS, \mathbb{S} \rangle$ to its export neighbors, another node k that has a valid candidate route $R = [\mathcal{K}, \ldots, j.AS, \ldots, d.AS]$ will receive the state-clarifier within $(hops(\mathcal{K}, j.AS, R) + 1) \times (T_m + U_d)$ time.

How G-BGP resolves uncertainty. When the highest ranked candidate route $R = [\mathcal{J}, \ldots, \mathcal{K}', \mathcal{K}, \ldots, d.AS]$ of a node i goes through some questionable AS $\mathcal{K}$, i suspects the validity of R and resolves the associated uncertainty, if either of the following conditions holds:[4]

- Condition 1: i has a point of segment-withdrawal $\langle \mathbb{S}, \mathbb{S}', \mathcal{I}', \mathcal{J}', i', t \rangle$ where $\mathcal{K} \in \mathbb{S}'$ and $[\mathcal{K}, \mathcal{I}', \mathcal{J}'] \in R$, or i has a point of node-join $\langle \mathcal{I}'', \mathbb{S}'', i', t \rangle$ where $\mathcal{K} \in \mathbb{S}''$ and $[\mathcal{K}, \mathcal{I}''] \notin R$; but the point of segment-withdrawal or the point of node-join is not piggybacked in the UPDATE message that contains R.

  In this case, i regards R as invalid if R has already been marked as invalid, or i regards R as valid if R has not been marked as invalid and $t \geq \alpha \times (hops(\mathcal{J}, \mathcal{K}, R) + 1) \times (T_m + U_d)$; otherwise, i *judiciously waits* for $(\alpha \times (hops(\mathcal{J}, \mathcal{K}, R) + 1) \times (T_m + U_d) - t)$ time, after which i either regards R as invalid if R has been marked as invalid, or i regards R as valid otherwise.

  If R is regarded as valid after the uncertainty resolution, no node whose highest ranked route R' goes through the new route $[i.AS, R]$ of i will suspect the validity of R, since the UPDATE message that contains R' must also contain the point of segment-withdrawal or the point of node-join.
- Condition 2: i has a point of channel-failure $\langle [\mathcal{I}', \mathcal{K}], t \rangle$ where $\mathcal{I}' \neq \mathcal{K}'$.

  In this case, i regards R as valid if i has a state-clarifier $\langle \mathbb{S}, \mathcal{K} \rangle$ where $\mathcal{K}' \notin \mathbb{S}$, or i regards R as invalid if i does not have such a state-clarifier yet $t \geq \alpha \times (hops(\mathcal{J}, \mathcal{K}, R) + 1) \times (T_m + U_d)$; otherwise, i judiciously waits for $(\alpha \times (hops(\mathcal{J}, \mathcal{K}, R) + 1) \times (T_m + U_d) - t)$ time, after which i either regards R as valid if i has a state-clarifier $\langle \mathbb{S}, \mathcal{K} \rangle$ where $\mathcal{K}' \notin \mathbb{S}$, or i regards R as invalid otherwise.

  If R is regarded as valid after the uncertainty resolution, i changes the state-clarifier into a set of points of channel-withdrawal $\{\langle [\mathcal{K}'', \mathcal{K}] \rangle : \mathcal{K}'' \in \mathbb{S}\}$; i also deletes every point of channel-failure $\langle [\mathcal{I}'', \mathcal{K}], t \rangle$ where $\mathcal{I}'' \in \mathbb{S}$, so that no node whose route goes through the route of i, no matter before or after i changes its route, will suspect the validity of routes going through $\mathcal{K}$. (Every newly generated point of channel-withdrawal $\langle [\mathcal{I}'', \mathcal{K}] \rangle$ that corresponds to a deleted point of channel-failure $\langle [\mathcal{I}'', \mathcal{K}], t \rangle$ assumes the sequence number of $\langle [\mathcal{I}'', \mathcal{K}], t \rangle$; the remaining newly generated points of channel-withdrawal do not assume any sequence numbers and are always regarded as fresh.)

By the above method of uncertainty resolution, an invalid route going through some questionable AS will be

4 If there are multiple questionable ASes in R, i resolves the uncertainty, if need be, regarding these ASes in parallel.

discarded, and fault-agnostic instability as well as its propagation is avoided. The elimination of this type of fault-agnostic instability is essential for G-BGP to converge at an asymptotically optimal speed or to asymptotically improve BGP convergence speed in several common scenarios (e.g., when a node with multiple neighboring ASes fail-stops), as proved in Section 4.

For a valid route R that is suspected by a node i, once i resolves the uncertainty regarding the validity of R, no node whose highest ranked route goes through the new route of i will suspect the validity of R any more. Thus, uncertainty regarding a valid route is *resolved locally* in the sense that only nodes relatively close to the questionable AS need to resolve the uncertainty, but nodes farther away need not.

Moreover, as observed in [15], the link latency as well as the processing delay for BGP messages is usually significantly less than the MRAI timer. Therefore, the propagation of UPDATE messages and the piggybacked fault information is much slower than the propagation of purging messages and state-clarifiers. Thus, with high probability, a node need not wait to resolve uncertainty regarding a valid route; and the waiting would be short even if need be.

3.2.3. Rejecting obsolete fault information To avoid using obsolete fault information, G-BGP enforces a total order on all the fault information regarding the same AS that is sent out from the AS at different time. This is achieved by assigning sequence numbers to fault information such that fresher fault information regarding an AS has a larger sequence number than does staler fault information regarding the same AS. *(Unless specified otherwise, all the arithmetic operations, including comparisons, in this section are based on "modulo a big number M".)*

Numbering fault information. To enable the sequence-number based checking of the freshness of fault information, nodes within an AS $\mathcal{I}$ coordinate with each other to maintain a monotonically-increasing sequence number $\mathcal{I}.sn$ for $\mathcal{I}$. For every neighboring AS $\mathcal{J}$, nodes in $\mathcal{I}$ also maintain a local copy of $\mathcal{J}$'s sequence number, denoted by $\mathcal{I}.\mathcal{J}.sn$. We assume that the synchronization delay between $\mathcal{J}.sn$ and $\mathcal{I}.\mathcal{J}.sn$ (i.e., $\mathcal{J}.sn - \mathcal{I}.\mathcal{J}.sn$) is bounded from above by D_{sn}. To guarantee monotonicity in the sequence number of an AS, a node stores the sequence number of its AS in a persistent memory; when a fail-stopped node i joins the network, i either gets the sequence number of its AS from some other up-node in the AS, or, if there is no up-node other than i in the AS, it gets the sequence number from its persistent memory and increases it by $D_{sn} + 1$.

When piggybacking fault information onto UPDATE messages that are sent to external neighbors, a node i attaches proper sequence number to each piece of fault information that is generated by i itself or some other node in $i.AS$:

- For each piece of fault information regarding the state of $i.AS$ (i.e., a point of channel-withdrawal $\langle [i.AS, \mathcal{J}] \rangle$, a point of segment-withdrawal $\langle \mathbb{S}, \mathbb{S}', i.AS, \mathcal{J}, i', t \rangle$, the point of AS-failure $\langle i.AS \rangle$, a point of channel-failure $\langle [i.AS, \mathcal{J}], t \rangle$, or a point of node-join $\langle i.AS, \mathbb{S}, i', t \rangle$), i simply attaches the sequence number $(i.AS).sn$. Then, i coordinates with other nodes in its AS to increase $(i.AS).sn$ by 1 (see Section 6 for implementation details).
- For every point of channel-withdrawal $\langle [\mathcal{K}, i.AS] \rangle$, if any, that is generated when $i.AS$ changes its export policy such that nodes in it do not export routes to an AS $\mathcal{K}$, i attaches the sequence number $((i.AS).\mathcal{K}.sn + D_{sn})$ instead of $(i.AS).sn$, since $\langle [\mathcal{K}, i.AS] \rangle$ is about the fact that nodes in $\mathcal{K}$ will not use any link between $\mathcal{K}$ and $i.AS$ in forwarding traffic.

  When nodes in $\mathcal{K}$ receive $\langle [\mathcal{K}, i.AS] \rangle$, they coordinate with one another to increase $\mathcal{K}.sn$ by $D_{sn} + 1$.
- For every point of AS-failure $\langle \mathcal{J} \rangle$, if any, that is generated for a fail-stopped neighboring AS $\mathcal{J}$, i attaches the sequence number $((i.AS).\mathcal{J}.sn + D_{sn})$ instead of $(i.AS).sn$, since $\langle \mathcal{J} \rangle$ is about the fail-stop of $\mathcal{J}$.

For fault information that is generated by nodes outside $i.AS$, i simply piggybacks the information onto UPDATE messages without changing the sequence number of the information.

How G-BGP rejects obsolete fault information. Towards enabling nodes in an AS $\mathcal{I}$ to determine the freshness of fault information regarding another AS $\mathcal{K}$, every time a node i in $\mathcal{I}$ receives some fresh fault information regarding $\mathcal{K}$, i reliably notifies[5] the other nodes in $\mathcal{I}$ of the information, and all the nodes in $\mathcal{I}$ maintain the sequence number of the information as $\mathcal{I}.\mathcal{K}.snM$ for up to T_d time, where T_d is the maximum difference in delay in propagating UPDATE messages along different routes from one AS to another. If no node in $\mathcal{I}$ receives any fresher fault information regarding $\mathcal{K}$ within T_d time after $\mathcal{I}.\mathcal{K}.snM$ was modified the last time, nodes in $\mathcal{I}$ delete $\mathcal{I}.\mathcal{K}.snM$, which we regard as "resetting $\mathcal{I}.\mathcal{K}.snM$ to $-\infty$". At an AS $\mathcal{I}$, $\mathcal{I}.\mathcal{K}.snM$ is initially regarded as $-\infty$ for every other AS $\mathcal{K}$, since no node in $\mathcal{I}$ maintains $\mathcal{I}.\mathcal{K}.snM$ initially.

When a node i receives an UPDATE message m containing a route R and some fault information, i checks the freshness of each piece of fault information and the validity of R as follows:

- For a piece of fault information regarding an AS $\mathcal{K}$, if the sequence number of the fault information is less than $(i.AS).\mathcal{K}.snM$ and the fault information signals a

5 See Section 6 for implementation details.

"withdrawn"-channel, a "withdrawn"-segment, a "fail-stopped" AS, or a "fail-stopped" channel that is, however, in a candidate route of i, then the fault information must be obsolete; otherwise, the fault information is fresh, in which case i updates $(i.AS).\mathcal{K}.snM$ with the sequence number of the fault information. (Note that m may contain obsolete and fresh fault information simultaneously.)

- If m contains any obsolete fault information, R must be invalid.

After the checking above, i accepts all fresh fault information and ignores all obsolete fault information; i also accepts the announced route R if m contains no obsolete fault information.

3.3. Example revisited

We revisit an example discussed in Section 3.1 and see how the network will behave if G-BGP is used. If a fail-stops when the network is at the state q as shown in Figure 1, b will detect the fail-stop of (b, a) and generate a point of channel-failure $\langle [b, a], t\rangle$. $\langle [b, a], t\rangle$ is piggybacked with UPDATE messages and propagated towards g. When g receives the route-withdrawal UPDATE message from m, g will learn, via $\langle [b, a], t\rangle$, the fail-stop of (b, a) and will not adopt $[f, b, a, d]$, even if f has not withdrawn the route. Moreover, since route $[j, h, a, d]$ goes through the questionable node a, g will resolve the uncertainty regarding the validity of $[j, h, a, d]$. By the uncertainty resolution, j will regard $[j, h, a, d]$ as invalid (possibly well before j withdraws $[j, h, a, d]$, since the uncertainty resolution is based on information flow speed that is not subject to the MRAI timer control). Then g changes its route directly to $[c, w, d]$. Therefore, there is no instability or instability propagation during the convergence.

4. Analysis of G-BGP

In the presence of the faults discussed in Section 2, three events can occur in a network: T_{down}, T_{up}, and T_{change}. T_{down} occurs when the destination d fail-stops (including d withdrawing its address prefix); T_{up} occurs when d newly joins the network; T_{change} occurs when d is up, but some node needs to change route as a result of some fault.[6]

6 Note that event T_{change} can be further divided into two categories: T_{long} and T_{short} [20], where T_{long} denotes the subcase where an existing path is replaced by a less preferred path, and T_{short} denotes the subcase where an existing path is replaced by a more preferred path. T_{long} can be triggered when a node or a link fail-stops, and T_{short} can be triggered when a node or a link joins. In this paper, we choose to use T_{change} when we do not need to differentiate T_{long} from T_{short} (e.g., when analyzing the stability of G-BGP), and we analyze each of the subcases separately when we need to differentiate them (e.g., when analyzing the convergence speed of G-BGP).

Towards analyzing the convergence properties of G-BGP and BGP in the above events, we first introduce the concept of "policy graphs".

Policy graph. Given a state q of a network G and the destination d, the *policy graph at state* q, denoted by $G_p.q$, is a directed graph $(V_p.q, E_p.q)$, where

$$\begin{aligned} V_p.q &= \{i : i \in V.q \wedge (\exists j : (j, i) \in E.q \wedge \\ &\qquad j \text{ exports its route for } d \text{ to } i \wedge \\ &\qquad i \text{ imports the route from } j)\} \\ E_p.q &= \{\langle j, i\rangle : j \in V_p.q \wedge i \in V_p.q \wedge \\ &\qquad j \text{ exports its route for } d \text{ to } i \wedge \\ &\qquad i \text{ imports the route from } j\} \end{aligned}$$

Using policy graphs, we comparatively study the convergence properties (i.e., stability and speed) of G-BGP and BGP under different event or fault scenarios; we also study the impact of route ranking policies. (Note that the proofs of the theorems in this section are relegated to [24].)

Convergence stability. For stability during G-BGP convergence, we have

Theorem 1 *When any of the events T_{down}, T_{up}, and T_{change} occurs, G-BGP converges with no fault-agnostic instability; this holds whether or not the SPF (or some non-SPF) route ranking policy is used.* ■

For stability during BGP convergence, we have

Theorem 2 *Fault-agnostic instability can occur during BGP convergence in both the event of T_{down} and T_{change}, whether or not the SPF (or some non-SPF) route ranking policy is used; during BGP convergence in the event of T_{up}, fault-agnostic instability can occur if some non-SPF policy is used, but fault-agnostic instability does not occur if the SPF policy is used.* ■

By Theorems 1 and 2, we see that G-BGP eliminates all the fault-agnostic instability that can occur during BGP convergence. The elimination of fault-agnostic instability is able to avoid the type of delayed BGP convergence that is due to the mis-interaction between BGP convergence instability and BGP route flap damping. Moreover, by eliminating fault-agnostic instability, G-BGP improves BGP convergence speed substantially and achieves asymptotically optimal convergence speed in several scenarios where BGP convergence is severely delayed (such as when a node or a link fail-stops), as shown later in this section and by our simulation in Section 5.

Furthermore, in the event of T_{down} where BGP exhibits its worst instability, the elimination of fault-agnostic instability in G-BGP also prevents distribution-inherent instability from happening, as shown by

Theorem 3 *G-BGP converges with no distribution-inherent instability in the event of T_{down}, whether or not the SPF (or some non-SPF) route ranking policy is used.* ■

Theorems 1 and 3 imply

Corollary 1 *G-BGP converges with no instability in the event of T_{down}, whether or not the SPF (or some non-SPF) route ranking policy is used.* ■

Convergence speed. For convenience, we define the following notations:

$\mathcal{R}(i, V, q)$	:	$\max_{j \in V} dist(i, j, q)$, where $dist(i, j, q)$ is the number of inter-AS hops in the shortest path from node i to j in the policy graph $G_p.q$, and each inter-AS hop in a path $\mathcal{L}$ in $G_p.q$ is a maximal-length path segment in $\mathcal{L}$ that consists of nodes from the same AS;
$\mathcal{D}(q)$	:	$\max_{j \in V.q} length(j.AS\text{-}path.q)$, where $length(j.AS\text{-}path.q)$ denotes the number of inter-AS hops in the route $j.AS\text{-}path.q$;
$\mathcal{LP}(V, q)$	:	the number of inter-AS hops in the longest simple path in the "subgraph of $G_p.q$ on the set V of nodes";
$hops(i, \mathcal{J}, q)$	:	the number of inter-AS hops between ASes $i.AS$ and $\mathcal{J}$ in the route $i.AS\text{-}path.q$.

We first analyze the convergence speed of G-BGP and BGP in the event of T_{down}, for both the SPF route ranking policy and non-SPF policies. In the event of T_{up} or T_{change}, distribution-inherent instability can happen during G-BGP convergence, which makes it difficult to asymptotically compare G-BGP and BGP convergence speed when non-SPF policies are used. Therefore, for the scenario where event T_{up} or T_{change} occurs, we only analyze the case when the SPF policy is used; we study the cases when non-SPF policies are used via simulation in Section 5.

In the event of T_{down}, we have

Theorem 4 *When a network is at a state q_0,*

(i) If d fail-stops gracefully, or if d fail-stops grossly when it has a single neighboring AS, G-BGP converges within $\theta(\mathcal{R}(i, V.q_0, q_0))$ time, which is asymptotically optimal; this holds whether or not the SPF (or some non-SPF) route ranking policy is used;

If d fail-stops grossly when it has multiple neighboring ASes, G-BGP converges within $O(\mathcal{D}(q_0))$ time, whether or not the SPF (or some non-SPF) policy is used;

(ii) If d fail-stops, it takes BGP up to $\theta(\mathcal{LP}(V.q_0, q_0))$ time to converge when the SPF policy is used and $O(n!)$ time when non-SPF policy is used.

(iii) $\mathcal{R}(i, V.q_0, q_0) \leq \mathcal{D}(q_0) \leq \mathcal{LP}(V.q_0, q_0)$. ■

In the event of T_{up}, we have

Theorem 5 *When the SPF route ranking policy is used, G-BGP as well as BGP converges to a stable state q_0' within $\theta(\mathcal{R}(i, V.q_0', q_0'))$ time in the event of T_{up}, which is asymptotically optimal.* ■

In the event of T_{change}, not every node needs to change route. A node is *affected* by a fault f if the node changes route at least once during convergence after f occurs; the set of all the nodes that are affected by a fault f is called the *affectation region of f*. Then, we have

Lemma 1 *When the SPF route ranking policy is used, the affectation region in G-BGP as well as BGP is minimal in the event of T_{change}.* ■

For the case where a network converges from a state q_0 to another state q_1, we define the following notations:

$AR(q_0, q_1)$	:	the set of nodes in $V.q_0$ that change route from q_0 to q_1, i.e., $\{k : k \in V.q_0 \wedge k.AS\text{-}path.q_0 \neq k.AS\text{-}path.q_1\}$;
$pt(k, q_0, q_1)$	:	the node in $AR(q_0, q_1)$ whose AS is in the route $k.AS\text{-}path.q_1$ and whose next -hop does not change route from q_0 to q_1;
$Tri(k, \mathcal{I}, q_0, q_1)$	:	$hops(pt(k, q_0, q_1), \mathcal{I}, q_0) + hops(k, pt(k, q_0, q_1).AS, q_1)$ for a node k in $AR(q_0, q_1)$.

Then, we have

Theorem 6 *When a network is at a state q_0 and when the SPF route ranking policy is used,*

(i) If a node in an AS $\mathcal{I}$ or a link associated with the node fail-stops, or if the routing policies of $\mathcal{I}$ change, G-BGP converges to a stable state q_1 within $\theta(\max_{k \in V.q_1 \wedge k \in AR(q_0, q_1)} Tri(k, \mathcal{I}, q_0, q_1))$ time, which is asymptotically optimal; it takes BGP up to $\theta(\mathcal{LP}(AR(q_0, q_1), q_0))$ time to converge in this case, and $\mathcal{LP}(AR(q_0, q_1), q_0) \geq \max_{k \in V.q_1 \wedge k \in AR(q_0, q_1)} Tri(k, \mathcal{I}, q_0, q_1)$;

(ii) If a node i or a link associated with i joins, G-BGP as well as BGP converges to a stable state q_1 within $\theta(\mathcal{R}(i, AR(q_0, q_1), q_1))$ time, which is asymptotically optimal. ■

By Theorems 4, 5, and 6, we see that G-BGP either achieves asymptotically optimal convergence speed or asymptotically improves the convergence speed of BGP in several scenarios where BGP exhibits delayed convergence (such as when a node or a link fail-stops). By Theorem 2, Lemma 1, and Theorem 6, we observe that, when a node or a link fail-stops or when an AS changes routing policies, fault-agnostic instability prevents BGP from converging at an asymptotically optimal speed (as does G-BGP), even though the affectation region is also minimal in BGP when the SPF route ranking policy is used.

On the other hand, when the SPF policy is used (as is the case in most ASes in the Internet), BGP converges at an asymptotically optimal speed when a node or a link joins. This conforms with our simulation results (as shown in Section 5) and the experimental observations [15, 16, 18] that

BGP does not experience much delay in convergence when a node or a link joins.

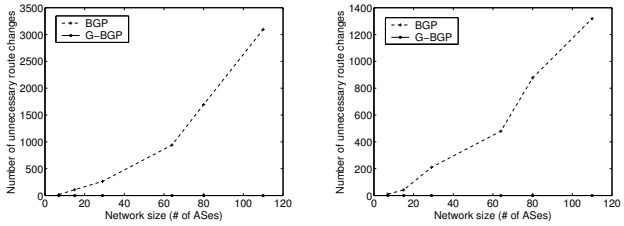
Related work revisited. As discussed in Section 1, the existing methods that try to improve BGP convergence are unable to eliminate all fault-agnostic instability. For example, fault-agnostic instability can still occur in the fault scenario discussed in Section 3.3 even if these methods are used. The existence of fault-agnostic instability in these methods prevents them from converging at optimal speeds in several important scenarios (e.g., when a node with multiple neighbors fail-stops) where G-BGP does; for the same reason, it is easy to prove that these methods do not asymptotically outperform G-BGP in any scenario. That is, even though these methods do improve BGP convergence and the improvement may even be significant in some cases, they do not give the proved guarantee and optimality that G-BGP does. (Therefore, in Section 5, we only compare G-BGP with the standard BGP.)

5. Simulation results

We implement G-BGP in SSFNet [1], a network simulator which has implemented a variety of standard Internet protocols such as BGP, OSPF, and TCP. For fidelity of simulation, we use realistic Internet-type topologies [1] to evaluate the convergence properties of G-BGP. To study the impact of network size as well as route ranking policy, we use networks of size ranging from 7 ASes to 115 ASes,[7] and we use both the SPF route ranking policy and a randomized non-SPF policy where every route r is assigned a random $rank(r)$. Then, we inject various types of faults (i.e., node fail-stop, node join, and policy change[8]) into networks to simulate the events of T_{down}, T_{up}, and T_{change}.

Event $\mathbf{T_{down}}$. When the destination d fail-stops, the number of unnecessary route changes during convergence and the convergence time of G-BGP as well as BGP are shown in Figures 2 and 3 respectively.

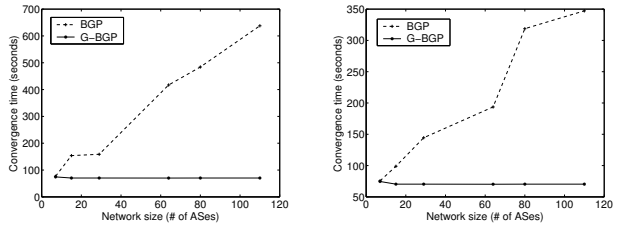
We see that G-BGP converges with no unnecessary route changes, as proved in Corollary 1. But there are many unnecessary route changes during BGP convergence, and the number increases quickly as the network size increases. If we measure convergence stability by the number of route changes during convergence (which equals to the number of unnecessary route changes plus the number of nodes), G-BGP improves BGP convergence stability by a factor of 29.4 for the network with 115 ASes. We also observe that,



(a) SPF policy

(b) Randomized non-SPF policy

Figure 2. The number of unnecessary route changes after the destination d fail-stops



(a) SPF policy

(b) Randomized non-SPF policy

Figure 3. The convergence time after the destination d fail-stops

as network size increases, the convergence time of G-BGP barely increases, but the convergence time of BGP increases quickly. For the network with 115 ASes, G-BGP reduces the convergence time of BGP by a factor of 10.2.

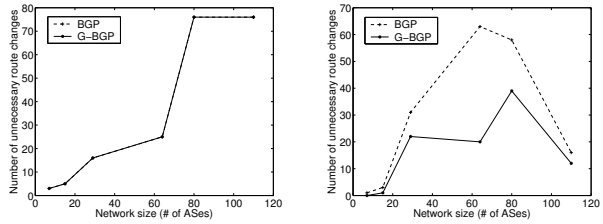
An interesting observation is that, in cases where the network size and the convergence time of BGP increase (e.g., when the network size increases from 85 ASes to 115 ASes), the convergence time of G-BGP may even decrease. The reason for this is that, as network size increases, the connectivity may increase, which reduces the average distance between nodes and thus the G-BGP convergence time. This is in contrast to BGP where, as network connectivity increases, the probability of using invalid routes and thus the convergence time increase.

Event $\mathbf{T_{up}}$. When the destination d joins, the number of unnecessary route changes during convergence and the convergence time of G-BGP as well as BGP are shown in Figures 4 and 5 respectively.

We see that, when the SPF policy is used, the number of unnecessary route changes during convergence and the

7 Given the unavailability of information on intra-AS structures, we only simulated 1 node in each AS. The observations will also apply to the case where an AS has multiple nodes in it, but the detailed study is relegated as a part of our future work.

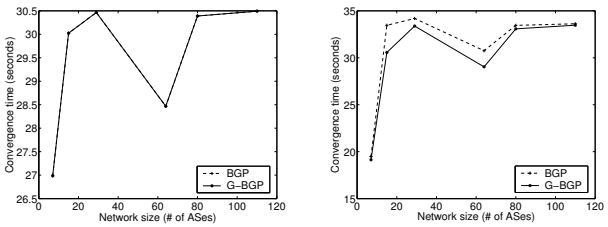
8 The impact of link fail-stop and link join is reflected via node fail-stop and node join respectively. Thus we do not simulate link fail-stop or link join.



(a) SPF policy

(b) Randomized non-SPF policy

Figure 4. The number of unnecessary route changes after the destination d joins



(a) SPF policy

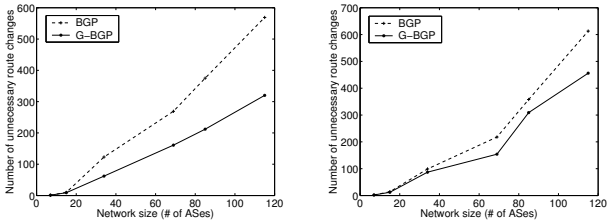
(b) Randomized non-SPF policy

Figure 5. The convergence time after the destination d joins

convergence time of BGP are the same as those of G-BGP, which is not unexpected since, as proved in Theorems 2 and 5, there is no fault-agnostic instability during BGP convergence and the convergence speed of BGP is asymptotically optimal in this case. On the other hand, when the randomized non-SPF policy is used, the number of unnecessary route changes during convergence and the convergence time of BGP are greater than those of G-BGP.

We also observe unnecessary route changes during G-BGP convergence, which is due to distribution-inherent instability. However, the time taken for G-BGP to converge is still quite short in spite of the distribution-inherent instability, which is similar to the observation in [18] that distribution-inherent instability does not cause long delay in BGP convergence.

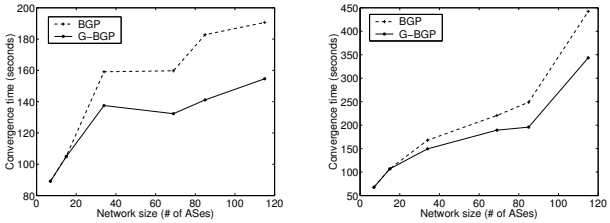
Event $\mathbf{T_{change}}$ when a node fail-stops. When a non-destination node fail-stops, the number of unnecessary route changes during convergence and the convergence time of G-BGP as well as BGP are shown in Figures 6 and 7 respectively. In this case, the patterns of difference in convergence



(a) SPF policy

(b) Randomized non-SPF policy

Figure 6. The number of unnecessary route changes after a non-destination node fail-stops



(a) SPF policy

(b) Randomized non-SPF policy

Figure 7. The convergence time after a non-destination node fail-stops

stability as well as speed between G-BGP and BGP are similar to those in the event of T_{down}.

Event $\mathbf{T_{change}}$ when a node joins. When a non-destination node joins, the patterns of difference in convergence stability as well as speed between G-BGP and BGP are similar to those in the event of T_{up}, and the results conform with Theorem 6. Due to the limitation of space, we relegate the detailed data as well as figures for this case to [24].

Event $\mathbf{T_{change}}$ when an AS changes routing policy. When an AS changes its routing policies, the patterns of difference in convergence stability as well as speed between G-BGP and BGP are similar to those in the case when a non-destination node fail-stops. Due to the limitation of space, we relegate the detailed data as well as figures for this case to [24].

6. Discussion

In this section, we discuss implementation and deployment issues for G-BGP. We also discuss approaches to reducing distribution-inherent instability.

Due to space limitation, we relegate to [24] the detailed program for G-BGP. Here we discuss mainly the issues involved in incorporating G-BGP with existing BGP standard.

Enhancing intra-AS coordination. G-BGP enhances the intra-AS coordination in BGP, so that each node i informs the other nodes in its AS of the route of i itself, the neighboring ASes to which i has exported its route, and the neighboring ASes to which i is connected via an up-link. It is straightforward to implement this technique if the basic BGP [22] is used, because all the nodes in an AS maintain IBGP sessions with each other. On the other hand, if route reflection [4] or AS confederation [19] is used, nodes in an AS may not maintain IBGP sessions with each other. To enable enhanced intra-AS coordination in the latter case, G-BGP requires that, (i) when route reflection is used, a route reflector provide the required information regarding nodes within its cluster to nodes outside its cluster, and that, (ii) when AS confederation is used, a node i having a BGP session with some node in a neighboring member-AS provide the information regarding nodes in the member-AS of i itself. Similarly, nodes in an AS can coordinate with each other to share fault information and to synchronize sequence numers of fault information. (Interestingly, it has also been proved that letting route reflectors expose more detailed information about nodes within their clusters solves the problem of persistent route oscillations caused by certain "route reflection" configurations [3].)

G-BGP in the presence of AS partition. The nodes in an AS are usually connected. However, it is possible (though rare) that an AS is partitioned due to some severe faults, in which case nodes within the AS cannot maintain a consistent view of routing. However, a consistent view of routing among nodes within the same AS is required in G-BGP for the task of generating certain fault information (i.e., a point of channel-withdrawal, a point of segment-withdrawal, or a point of channel-failure), as well as the task of assigning sequence numbers to fault information.

To guarantee the correctness of G-BGP in the presence of AS partition, G-BGP can be adapted as follows: First, whenever a node i in a partitioned AS would generate a point of channel-withdrawal, a point of segment-withdrawal, or a point of channel-failure regarding a channel $(i.AS, \mathcal{J})$ under normal G-BGP operation, i generates a point of segment-withdrawal $\langle \emptyset, \mathbb{S}', i.AS, \mathcal{J}, i, t \rangle$ instead, where $\mathbb{S}'$ is the set of ASes to which i has exported its last route; Second, whenever i would generate a sequence number under normal G-BGP operation, i also attaches its node-id (e.g., BGP identifier) to signal the fact that the freshness of the corresponding fault information should be verified on the basis of node i instead of its AS $i.AS$.

Encoding fault information. Besides the information used in BGP, G-BGP uses fault information, purging-messages, and state-clarifiers. Therefore, to implement G-BGP in a way that allows graceful migration of and interoperability with BGP, one key issue is to incorporate these new types of information into the existing BGP message format such that G-BGP and BGP can inter-operate. To this end, we define a new *optional transitive* path attribute [22], and use this attribute to encode the information introduced in G-BGP. According to BGP specification, a BGP speaker propagates a transitive attribute upon receipt, whether or not the BGP speaker implements G-BGP. (For brevity, we relegate the detailed encoding specification to [24].)

Incremental deployment of G-BGP. Given that G-BGP uses an optional transitive path attribute to carry fault information, G-BGP can be incrementally deployed and interoperate well with BGP. Moreover, even in the case of partial deployment, the improvement in convergence stability and speed is guaranteed for those ASes that deploy G-BGP: when a fault occurs, information about the fault will be generated at some node that deploys G-BGP and is affected by the fault; then the fault information is propagated, along with BGP UPDATE messages, to other affected nodes; when the fault information reaches a node that deploys G-BGP, the node can use the fault information to avoid fault-agnostic instability and to expedite the network convergence.

Approaches to reducing distribution-inherent instability. Even though distribution-inherent instability does not cause much delay in BGP convergence, it may enlarge the affectation regions of faults when non-SPF route ranking policies are used. As a result, some nodes are affected, even if they do not have to change routes in the presence of faults. Therefore, the time taken for G-BGP and BGP to converge is increased by an amount depending on the number of such nodes. One way to ameliorate this issue of enlarged affectation region is to use the technique of local stabilization [2], which contains the impact of distribution-inherent instability locally around where it occurs, so that the affectation region is bounded in diameter (only as a function of the degree of fault perturbation in a network).

Moreover, to reduce type-(i) distribution-inherent instability, one approach is to reduce the delay in information sharing by propagating fault information faster; another approach is for nodes to wait conservatively before changing routes, in hope that fresher information will arrive.

7. Concluding remarks

The stability and speed of BGP convergence are closely related. We studied the nature of instability during BGP convergence, and we classified the instability into two categories: fault-agnostic instability and distribution-inherent instability. Distribution-inherent instability does not cause severe delay in BGP convergence and provably exists in every distributed routing protocol. Therefore, we focused on mechanisms to eliminate fault-agnostic instability, and we proved that the elimination of fault-agnostic instability enables G-BGP to asymptotically improve BGP convergence speed and to converge at an asymptotically optimal speed in several common scenarios where BGP convergence is severely delayed (such as when a node or a link fail-stops).

In G-BGP, fault-agnostic instability is removed by rejecting invalid routes and obsolete fault information. In general, we believe that propagating information about network dynamics (such as faults) and using better state detection techniques can help the affected nodes adapt their behaviors during convergence, which is also feasible given today's high speed networks.

The philosophy of "information hiding" in hierarchical structures is observed in G-BGP in the sense that it does not expose extra information at the intra-AS level to the inter-AS level. G-BGP does not introduce additional information that needs to be permanently maintained between far away nodes, thus G-BGP does not introduce extra instability in the presence of network dynamics. In general, "information hiding" helps contain the impact of system dynamics locally around where the dynamics occur, and to guarantee system stability, "information hiding" should be observed as a principle when we design new protocols or migrate existing protocols [2].

We mainly focused on the issues related to fault-agnostic instability in this paper. In our future work, we will study in more detail the impact of distribution-inherent instability on BGP convergence speed; we will also study the fundamental limits on approaches to reducing distribution-inherent instability.

References

[1] Modeling the global internet. In *http://www.ssfnet.org/*.

[2] A. Arora and H. Zhang. LSRP: Local stabilization in shortest path routing. In *IEEE-IFIP DSN*, 2003.

[3] A. Basu, C.-H. L. Ong, A. Rasala, F. B. Shepherd, and G. Wilfong. Route oscillations in I-BGP with route reflection. In *ACM SIGCOMM*, pages 235–247, 2002.

[4] T. Bates, R. Chandra, and E. Chen. BGP route reflection: an alternative to full mesh IBGP. In *IETF RFC 2796*, April 2000.

[5] J. Bennett and C. Partridge. Packet reordering is not pathological network behavior. *IEEE/ACM Transactions on Networking*, 7(6):789–798, 1999.

[6] A. Bremler-Barr, Y. Afek, and S. Schwarz. Improved BGP convergence via ghost flushing. In *IEEE INFOCOM*, 2003.

[7] J. Chandrashekar, Z. Duan, Z.-L. Zhang, and J. Krasky. Limiting path exploration in bgp. In *IEEE INFOCOM*, 2005.

[8] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the Internet topology. In *ACM SIGCOMM*, pages 251–262, 1999.

[9] L. Gao and J. Rexford. Stable Internet routing without global coordination. *IEEE/ACM Transactions on Networking*, 9(6):307–317, 2001.

[10] M. G. Gouda. *Elements of Network Protocol Design*. John Wiley & Sons, INC., 1998.

[11] T. G. Griffin, F. B. Shepherd, and G. Wilfong. The stable paths problem and interdomain routing. *IEEE/ACM Transactions on Networking*, 10(2):232–243, 2002.

[12] T. G. Griffin and G. Wilfong. On the correctness of IBGP configuration. In *ACM SIGCOMM*, pages 17–29, 2002.

[13] J. Hawkinson and T. Bates. Guidelines for creation, selection, and registration of an Autononmous System (AS). In *IETF RFC 1930*, March 1996.

[14] C. Huitema. *Routing in the Internet*. Prentice Hall, 2000.

[15] C. Labovitz, A. Ahuja, and A. Bose. Delayed Internet routing convergence. In *ACM SIGCOMM*, pages 175–187, 2000.

[16] C. Labovitz, A. Ahuja, R. Wattenhofer, and S. Venkatachary. The impact of Internet policy and topology on delayed routing convergence. In *IEEE INFOCOM*, pages 537–546, 2001.

[17] J. Luo, J. Xie, R. Hao, and X. Li. An approach to accelerate convergence for path vector protocol. In *IEEE GLOBECOM*, pages 2390 – 2394, 2002.

[18] Z. M. Mao, R. Govindan, G. Varghese, and R. H. Katz. Route flap damping exacerbates Internet routing convergence. In *ACM SIGCOMM*, pages 221–233, 2002.

[19] D. McPherson and J. Scudder. Autonomous system confederations for BGP. In *IETF Internet draft: http://www.ietf.org/internet-drafts/draft-ietf-idr-rfc3065bis-01.txt*, October 2003.

[20] D. Pei, M. Azuma, D. Massey, and L. Zhang. BGP-RCN: Improving BGP convergence through root cause notification. *Computer Networks*, 48:175–194, 2005.

[21] D. Pei, X. Zhao, L. Wang, D. Massey, A. Mankin, S. F. Wu, and L. Zhang. Improving BGP convergence through consistency assertions. In *IEEE INFOCOM*, pages 976–985, 2002.

[22] Y. Rekhter, T. Li, and S. Hares. A border gateway protocol 4 (BGP-4). In *IETF Draft (http://www.ietf.org/internet-drafts/draft-ietf-idr-bgp4-22.txt)*, October 2003.

[23] L. Wang, X. Zhao, D. Pei, R. Bush, D. Massey, A. Mankin, S. F. Wu, and L. Zhang. Observation and analysis of BGP behavior under stress. In *ACM SIGCOMM-USENIX IMW*, pages 183–195, 2002.

[24] H. Zhang, A. Arora, and Z. Liu. A stability-oriented approach to improving bgp convergence. In *Technical Report, OSU-CISRC-6/05-TR45, The Ohio State University (ftp://ftp.cse.ohio-state.edu/pub/tech-report/2005/TR45.pdf)*, June 2005.