A Stability-oriented Approach to Improving BGP Convergence *

Hongwei Zhang Anish Arora Zhijun Liu †

Technical Report: OSU-CISRC-6/05-TR45

Department of Computer and Information Science
The Ohio State University, USA
{zhangho, anish, liuzh}@cis.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 that it is impossible to eliminate all distribution-inherent instability in any distributed routing protocol; 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) rejecting obsolete fault information, and (iii) quickly resolving the uncertainty between link and node failure as well as the uncertainty of whether a node has changed route.

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 factors of 29.4 and 10.2 respectively.

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

*This work was partially sponsored by DARPA contract OSU-RF #F33615-01-C-1901, NSF grant NSF-CCR-9972368, an Ameritech Faculty Fellowship, and two grants from Microsoft Research.

†Email: {zhangho, anish, liuzh}@cis.ohio-state.edu; Tel: +1-614-292-{1932, 1836, 7344}; Fax: +1-614-292-2911; Web: http://www.cis.ohio-state.edu/{~zhangho, ~anish, ~liuzh}.

1 Introduction

The Border Gateway Protocol (BGP) is used to coordinate routing among autonomous systems (simply called ASes hereafter) in the Internet [15]. Theoretically, BGP does not guarantee convergence and allows persistent route oscillations in several scenarios, such as conflicting routing policies and improper IBGP configurations [4, 12, 13]. In practice, however, most ASes in the Internet use the shortest-path-first route ranking policy (whereby a path with the least hop count is chosen), and as a result, BGP converges with high probability [18, 25]. In addition, for cases where the shortest-path-first policy is not used, solutions have been proposed to avoid persistent route oscillations in BGP [4, 9, 12, 13].

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 [17, 18]). Instability during convergence is undesirable. First, it increases the probability of message reordering, which is not only undesirable for multimedia applications, but also increases the probability of undesirable timer expiration in protocols such as TCP and IP. Second, instability increases delay jitter in packet delivery. And finally, instability increases packet loss (e.g., due to TTL expiration) [17]. Slow convergence of BGP is also undesirable, because 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 [26]. 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 [20].

Related work. To improve BGP convergence speed, the methods of "consistency assertions" [23] and "ghost flushing" [6] have been proposed. The former captures consistency properties between neighboring nodes, and the latter withdraws old routes faster than propagating new routes. However, "consistency assertions" do not deal with slow BGP convergence that results from inconsistency between nodes multiple hops away, and neither of the two approaches can remove all convergence-malign instability, the major cause for slow BGP convergence (to be discussed in Section 3.1). Moreover, the nature of different types of instability during BGP convergence, the fundamental limits on improving BGP convergence stability and speed, and the impact of fault types and routing policies on protocol convergence behaviors are not the focus of [23] and [6]. In addition, the "consistency assertions" method propagates the entry-router-id, which is essentially an attribute below instead of at the level of ASes, of one AS to other ASes, thus local changes of entry-router within an AS (even when the AS-path does not change) will propagate to other ASes, which are potentially far away, and leads to propagation of unnecessary route-changes.

An application-layer approach, resilient overlay networks [2], is also proposed to deal with slow BGP convergence. But the approach is not scalable in the sense that each node in a network maintains information about the whole network, and the approach does not improve the convergence behaviors of BGP.

In [17] and [18], the delayed BGP convergence and the impact of Internet policy as well as topology on BGP convergence are studied. But the study considers only the shortest-path-first policy, and does not consider scenarios where a node or a link joins. In [22], a real-time model for BGP convergence is proposed, but the analysis only considers the case when a destination node joins a network. Moreover, no solution for improving BGP convergence behavior is proposed in [17], [18] and [22].

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 any stateful distributed routing protocol.

Then, we refine BGP to obtain a new protocol G-BGP (for *Grapevine-BGP*) that eliminates all fault-agnostic instability. G-BGP achieves this via three mechanisms: First, it propagates fine-grained information about faults to the nodes that are affected by the faults; Second, it rejects obsolete fault information, by enforcing a total order on all the fault information regarding the same AS that is sent out from the AS at different time; Third, it quickly resolves the uncertainty between link and node failure as well as the uncertainty of whether a node has changed route.

Towards analyzing the convergence behaviors of G-BGP and BGP, we introduce *policy graphs* as a modelling tool for inter-AS routing. Using policy graphs, we prove that G-BGP eliminates all fault-agnostic instability under different fault and routing policy scenarios. We also prove 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 also evaluate 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.

Moreover, G-BGP is scalable along a number of dimensions. First, fault information, most of which is piggybacked in UPDATE messages, consumes little network bandwidth, and fault information is either not stored or only temporarily stored at nodes. Second, the degree of improvement in G-BGP increases as network size increases. Third, G-BGP does not expose additional intra-AS attributes and thus does not introduce additional instability that is due to local state changes within an AS. And finally, each node only maintains routes of its immediate neighbors.

Organization of the paper. In Section 2, we present the network model, fault model, as well as protocol notation, and we briefly describe BGP. In Section 3, we study the nature of BGP convergence instability and present the G-BGP design. Then, we introduce policy graph in Section 4, and in Section 5, we analyze the convergence stability as well as speed in G-BGP and BGP. We present our simulation results in Section 6. In Section 7, we discuss the implementation as well as deployment considerations of G-BGP, and we discuss approaches to reducing distribution-inherent instability. Section 8 concludes the paper.

2 Preliminaries

In this section, we present the network model, fault model, and protocol notation. 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 7). 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.

Message transmission between nodes is reliable, and message passing delay across a link is bounded from below and from above by L_d and U_d respectively.

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.)

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.

Due to faults, a network G may change dynamically 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.

Protocol notation. We write protocols using the guarded command notation [10]. At each node, the protocol consists of a finite set of variables and actions. Each action consists of two parts: guard and statement. For convenience, we associate a unique name with each action. Thus, an action has the following form:

$$\langle name \rangle :: \langle guard \rangle \longrightarrow \langle statement \rangle$$

The guard is either a boolean expression over the protocol variables of the node or a message reception operation; the statement updates zero or more protocol variables of the node, and/or sends out some message(s). An action is enabled if its guard evaluates to true. An action is executed only if it is enabled. To execute an action, its statement is executed atomically.

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 (i.e., the destination address);
$next\_hop$	:	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 [24]:

- 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.

[24]. 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 [14, 24].

3 The G-BGP protocol

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 achieve the objective, 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.

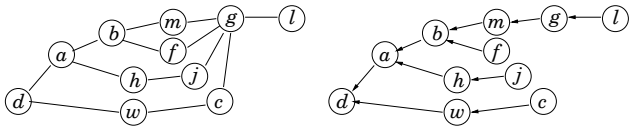
3.1 Instability during BGP convergence

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 [17], [20], 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 signalling 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) [20].

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; for the three backup routes of g at state q, route



(a) Network topology (b) Routing tree

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

$[f, b, a, d]$ 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 6 and in [20]). Moreover, distribution-inherent instability exists in every distributed routing protocol, as proved in

Theorem 1 (Impossibility of eliminating all distribution-inherent instability) *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.*

Proof: In inter-AS routing, besides network topology and export as well as import policies, route ranking policies adopted at ASes determine the route chosen by an AS. When route ranking policies are not shared among ASes (which is the common practice in Internet), a node j cannot predict the route taken by other nodes even if j can learn the whole network topology, the export and import policies of other ASes. Therefore, a node can only choose and set up its route based on the routes adopted by its import neighbors. This fact, together with unpredictability of faults and greater-than-zero link delay, results in the impossibility of completely avoiding distribution-inherent instability as explained below.

When faults occur in a network, the routes of those nodes where the faults have occurred may change. Then the export neighbors of these nodes may change accordingly. A node k may choose as its route a route r exported from one of its import neighbors k' before k learns the best route r' available to k, and r is a temporary route for k even though r may be the final stable route (i.e., the best route available) of k'. This is due to the following two reasons:

a) The best route r' available to k in a given network topology and certain routing policies may reach k after a less preferred route r available to k has reached k, because message passing delay along different routes are different as well as non-zero;

b) k cannot predict for sure the existence of r' or the delay between the receipt of r and r', because routing policies are not shared among ASes.

   Therefore, there are two alternatives k can adopt:

   - k always chooses the best route it has learned of so far: in this case, it is trivially true that k may choose r before it chooses r'.
   - k waits for some time t_k before choosing the best route it has learned after k has learned some change(s) in network state: in this case, we can always construct an instance of the problem where there exists some k that chooses a less preferred route r before k learns the best route r' available to it.

     Consider a network where the topology is a complete graph and the number of nodes is greater than 4, then for any set of values $t_{k''}$ for every node k'' in the network, we can always find four values t_{k0}, $t_{k'}$, t_k, and t_{k1} such that $t_{k0} + t_{k'} > t_k > t_{k1}$. If the routing policy at node k is such that k prefers route that goes through $k0$ and k' to route that goes through $k1$, then k chooses the route

that goes through $k1$ (i.e., the route r) before it learns the more-preferred route that goes through $k0$ and k' (i.e., the route r').

Therefore, a node k may choose and advertise to its export neighbors some temporary routes before it reaches its final stable state.

Thus, generally speaking, for a node i, when it has received some routes exported from its import neighbors after the occurrence of faults, i may choose as its route to the destination the route r from one import neighbor j such that r is a transient route of j.[1] Therefore, i has to change its route after j changes its route from r to its final stable route r', and instability (i.e., extra route-change) is incurred at i. This instability is the kind of distribution-inherent instability where the adopted route at a node is valid but is transient. This kind of distribution-inherent instability is partly inherent with the fact that the routing policies are not shared among ASes, and is impossible to completely avoid in any distributed routing protocols.

In distributed routing protocols, the propagation of the kind of instability where the adopted route at a node is valid but transient can result in the other kind of distribution-inherent instability where the adopted route is invalid because no information about the invalidity of the route could have reached the node when it decides to adopt the route. (The reasoning is the same as that for proving that there exists k that chooses a route r before learning its final stable route r. For simplicity, we omit it here.) Following the example discussed in the last paragraph, after j changes its route from r to r', we consider another node k that chooses as its route a route r'' that includes r when no (implicit or explicit) information regarding the route change from r to r' at j has reached k due to non-zero link delay. Then k will change its route at least once more later on, since j is not using route r' anymore.

Moreover, if multiple faults occur in a network simultaneously, there can also exist the kind of distribution-inherent instability where the adopted route is invalid because no information about the invalidity of the route could have reached the node when it decides to adopt the route. Following the same reasoning for proving the existence of k that chooses a route r before learning its final stable route r, we can always find two independent faults $F1$ and $F2$ that occur to two different nodes and another node k such that some information about $F1$ reaches k earlier than information about $F2$ does, which makes k choose a route r'' that has been invalidated by $F2$ before information about $F2$ could reach k because of non-zero link delay. Then k will change its route at least once more later on since r'' is an invalid route.

□

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 in Section 7.

3.2 G-BGP design

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,

[1] This is because message passing delay along links is greater than zero, routing policies is not shared among ASes, and independence of faults that happen at different time or to different ASes. Detailed proof is the same as that for proving there exists k that chooses a route r before learning its final stable route r. For simplicity, we omit it here.

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 all the nodes in an AS fail-stop, when the channel between two ASes fail-stops, or when a node joins the network, a *point of AS-failure*, a *point of channel-failure*, or a *point of node-join* is also propagated respectively.

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 on all the fault information regarding the same AS that is sent out from the AS at different time.

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 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 suspectable invalid 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].

We elaborate on the above mechanisms in the following subsections.

3.2.1 Propagating fault information in a bounded manner

Towards enabling nodes to generate appropriate fault information in the presence of faults, the intra-AS coordination in BGP is enhanced as follows: 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[2], but no link between $\mathcal{J}$ and $i.AS$ is used by any node in $i.AS$, 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 $i.AS$ 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 every $\mathcal{K} \in \mathbb{S}$, i has exported route R to $\mathcal{K}$, but there is no node in $i.AS$ that has exported its route to $\mathcal{K}$ and is still using any link between $\mathcal{J}$ and $i.AS$. i also calculates the set $\mathbb{S}'$ of its neighboring ASes such that, for every $\mathcal{K}' \in \mathbb{S}'$, i has exported route R to $\mathcal{K}'$, and there is at least one node in $i.AS$ that has exported its route to $\mathcal{K}'$ and is still using a link between $\mathcal{J}$ and $i.AS$

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 $\mathbb{S} \neq \emptyset$ and/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 3.2.3). 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 suspected-segment and $\mathcal{K}'$ a suspected AS.

[2] In the case when R has become invalid, but the fault information at i does not invalidate R, i also regards R as "valid". This can happen when G-BGP is only partially deployed and the fault happens to a network region where G-BGP is not deployed.

Point of AS-failure. When all the nodes in an AS $\mathcal{I}$ fail-stop, every route going through $\mathcal{I}$ becomes invalid. To signal this fact, a *point of AS-failure* $\langle \mathcal{I} \rangle$ is generated by every node j, if any, that detects the fail-stop of $\mathcal{I}$ and whose current route goes through $\mathcal{I}$.

A special case is when the destination d withdraws its address prefix. In this case, nodes in d generate the point of AS-failure $\langle d \rangle$, since the effect of d withdrawing its address prefix is the same as that of d fail-stopping.

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 suspected 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 those going 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 suspected AS.

How G-BGP uses and propagates fault information in a bounded manner. As discussed above, 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.e., $i.AS$).

  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 and avoids using the routes that go through any withdrawn channel, withdrawn segment, fail-stopped AS, and/or fail-stopped channel. Moreover, if the highest ranked candidate route R goes through some suspected 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,[3] where U is the upper bound on the convergence time of BGP after a fault occurs. 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 will update the information, if need be, as the network state changes:

- 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] 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.

3.2.2 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 some big number M".)*

Numbering fault information. To enable the sequence-number based information-freshness-checking , 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 coordinate with other nodes in its AS to increase $(i.AS).sn$ by 1.

- 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 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 $-\infty$ for every other AS $\mathcal{K}$.

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 "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.

3.2.3 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 suspectable invalid routes" and "collaboratively clarifying state".

Quickly marking suspectable invalid 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 suspected 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 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,[4] 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 suspected AS $\mathcal{K}$, i suspects the validity of R and resolves the associated uncertainty, if either of the following conditions holds:[5]

- 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

[4] G-BGP changes the BGP UPDATE-send method, so that when a node i uses the route imported from one of its import neighbors j, i also sends its route back to j. Therefore, a node j can decide whether its route is used by some of its external neighbors in a neighboring AS. By letting nodes in the same AS share with each other this information, a node can decide whether a channel between its AS and a neighboring AS is used by some node in the neighboring AS.

[5] If there are multiple suspected ASes in R, i resolves the uncertainty, if need be, regarding these ASes in parallel.

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 suspected AS will be 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 achieve asymptotically optimal convergence 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 5.

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 that are relatively close to the suspected AS need to resolve the uncertainty, but nodes that are farther away from the suspected AS need not.

Moreover, as observed in [17], 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.

3.3 Protocol G-BGP

We present G-BGP in Figure 2, where the variables and protocol actions of each node i are presented. (For conciseness, we skip the program for "fast marking of suspectable invalid routes" and "collaborative clarification".)

Variables. Each node i maintains variables $i.poas$, $i.pocf$, $i.pocw$, $i.ponj$, $i.posw$, $i.AS\text{-}path$, $i.inval$, $i.sn$, $i.SN$, $i.change$, $i.seqChg$, $i.suspect$, $i.adv$, and $i.pAdv$:

Program $G\text{-}BGP.i$
Parameter j : node-id; J : set of AS-ids
Var $i.pocw$: set of ⟨channel, int⟩ [initialized to $\emptyset$];
$i.posw$: set of ⟨AS-id set, AS-id set, channel, int, int, int⟩ [initialized to $\emptyset$];
$i.poas$: set of ⟨AS-id, int⟩ [initialized to $\emptyset$];
$i.pocf$: set of ⟨channel, int, int⟩ [initialized to $\emptyset$];
$i.ponj$: set of ⟨AS-id, AS-id set, int, int, int⟩ [initialized to $\emptyset$];
$i.AS\text{-}path, i.j.AS\text{-}path,$: sequence of AS-ids [initialized to $\emptyset$];
$i.inval$: set of AS-ids [initialized to $\emptyset$];
$i.sn, i.j.sn, s$: int [initialized to 0];
$i.SN$: set of ⟨AS-id, int, int⟩ [initialized to $\emptyset$];
r' : sequence of AS-ids; n, k', k'' : AS-id;
e : link (i.e., pair of AS-ids); l : ⟨link, int, int⟩ or ⟨ ⟨AS-id, AS-ids⟩, int, int⟩;
t, t', wt : time;
$i.change, i.seqChg, i.suspect, i.adv, i.pAdv, i.advd$: boolean [initialized to $false$];
Action
FAULT-INFO
[]
ROUTE-ADAPT
[]
ADV-RESET

Figure 2: G-BGP: improve BGP convergence stability and speed

$\langle A1\rangle$ :: i *changes routing policy* $\longrightarrow$
if *removes* J *from* $i.im$ $\rightarrow$ $i.pocw, i.change := i.pocw \cup \{\langle j, i.AS, i.sn\rangle : j \in J\}, true$ **fi**
if *removes* $J from\ i.ex$ $\rightarrow$ $i.pocw := i.pocw \cup \{\langle i.AS, j, i.j.sn + D_{sn} : j \in J\}$;
send $UPDATE(WD(i), i.pocw, \emptyset, \emptyset, \emptyset, \emptyset)$ **to** J
fi
if i *changes ranking policy* $\rightarrow$ $i.change := true$ **fi**
[]
$\langle A2\rangle$ :: $(j = nHop(i) \wedge link\ (j,i)\ fail\text{-}stops) \vee (i = d \wedge i\ withdraws\ its\ address\ prefix) \longrightarrow$
if $channel\ (j.AS, i.AS)\ fail\text{-}stops$ $\rightarrow$ $i.pocf, i.seqChg := i.pocf \cup \{\langle\langle j.AS, i.AS\rangle, detectionTime, i.sn\rangle\}, true$;
[] $i\ withdraws\ its\ address\ prefix$ $\rightarrow$ $i.poas, i.seqChg := i.poas \cup \{\langle i.AS, i.sn\rangle\}, true$
fi
if $\neg i.change$ $\rightarrow$ $i.change, t := true, CLOCK$ **fi**
[]
$\langle A3\rangle$ :: **rcv** $UPDATE\ m(r, r.pocw, r.posw, r.poas, r.pocf, r.ponj, r.sn)$ **from** j $\longrightarrow$
$(\forall l : l \in (r.pocf \cup r.ponj) : l.tPsd := l.tPsd + L_d)$;
if $r.sn > i.j.sn \rightarrow i.j.sn := r.sn$ **fi**;
if m *is not a withdrawal-*$UPDATE \rightarrow$
$i.j.AS\text{-}path := r$;
if $\neg Obsolete(r,i) \rightarrow i.inval := i.inval \setminus \{j\}$;
$i.poas, i.pocf, i.pocw, i.posw := D(i.poas, r), D(i.pocf, r), D(i.pocw, r), D(i.posw, r)$;
[] $Obsolete(r,i) \rightarrow i.inval := i.inval \cup \{j\}$
if
[] m *is a withdrawal-*$UPDATE$ $\rightarrow$ $i.j.AS\text{-}path := \emptyset$
fi
$i.SN, i.sn, i.seqChg := adpt(i.SN, r), adpt(i.sn, r, j), chgSeq(i, r)$;
if $i.AS\text{-}path \neq \emptyset$ $\rightarrow$
$i.poas, i.pocf, i.pocw := M(i.poas, r.poas), M(i.pocf, r.pocf), M(i.pocw, r.pocw)$;
$i.posw, i.ponj := M(i.posw, r.posw), M(i.ponj, r.ponj)$
fi
$(\forall k', k'', s, l : \langle\langle k', k''\rangle, s\rangle \in i.pocw \wedge l \in i.ponj \wedge k'' \in l.ASes \wedge s \geq l.k''.sn : l.ASes := l.ASes \setminus \{\langle k'', l.k''.sn\rangle\})$;
if $\neg i.change$ $\rightarrow$ $i.change, t := true, CLOCK$ **fi**

Figure 3: Module FAULT-INFO

$\langle A4\rangle :: \ i.change \longrightarrow$
$(\forall l : l \in (i.pocf \cup i.ponj) : l.tPsd := l.tPsd + (CLOCK - t)/\alpha);$
$i.inval := (i.inval \cup \{j : Invalid(i.j.AS\text{-}path, i)\}) \setminus \{j : \neg Invalid(i.j.AS\text{-}path, i)\};$
if $mPref(i) \neq \perp \rightarrow i.AS\text{-}path, i.adv := [i.AS, mPref(i)], false;$
if $(\exists k' : Suspect(k', i)) \wedge (\neg i.suspect \vee modified(i)) \rightarrow$
$i.suspect, t, wt, i.pAdv := true, CLOCK, \max\{0, 2\alpha \times maxWait(i)\}, true;$
$[] \neg(\exists k' : Suspect(k', i)) \ \rightarrow \ i.adv := true$
fi
$[] \ mPref(i) = \perp \rightarrow i.AS\text{-}path, i.adv := \emptyset, true$
fi
[]
$\langle A5\rangle :: \ i.suspect \longrightarrow$
if $CLOCK \leq t + wt \wedge Invalid(i.AS\text{-}path, i) \ \rightarrow \ i.suspect := false;$
$[] \ CLOCK > t + wt \wedge \neg Invalid(i.AS\text{-}path, i) \ \rightarrow$
$i.suspect := false;$
$(\forall k', k'', t', s : k' \in i.AS\text{-}path \wedge \langle\langle k', k''\rangle, t', s\rangle \in i.pocf :$
$i.pocw, i.pocf := i.pocw \cup \{\langle\langle k', k''\rangle, s\rangle\}, i.pocf \setminus \{\langle\langle k', k''\rangle, t', s\rangle\});$
fi

Figure 4: Module ROUTE-ADAPT

$\langle A6\rangle :: \ (i.adv \vee i.pAdv) \wedge mrai(i) \longrightarrow$
if $i.pAdv \wedge \neg i.advd \rightarrow i.pAdv, i.advd := false, true;$
send $UPDATE(WD(i), i.pocw, i.posw, i.poas, i.pocf, i.ponj, i.sn)$ **to** $(i.ex \cup \{preNHop(i)\});$
$[] \ i.adv \rightarrow$
if $diff(i) \wedge i.AS\text{-}path \neq \emptyset \ \rightarrow$
if $nHop(i) \neq preNHop(i) \wedge preNHop(i) \notin i.inval \rightarrow$
if channel $(preNHop(i), i.AS)$ is not used $\rightarrow$
$i.pocw, i.seqChg := i.pocw \cup \{\langle\langle preNHop(i), i.AS\rangle, i.sn\rangle\}, true$
$[]$ channel $(preNHop(i), i.AS)$ is still used $\rightarrow$
$i.posw, i.seqChg := i.posw \cup \{\langle \mathbb{S}(i), \mathbb{S}'(i), i.AS, preNHop(i), i.AS, i, 0, i.sn\rangle\}, true$
fi
fi
if $preNHop(i) = \perp \rightarrow \ i.ponj, i.seqChg := i.ponj \cup addPoj(i), true$ **fi**
send $UPDATE(i.AS\text{-}path, i.pocw, i.posw, i.poas, i.pocf, i.ponj, i.sn)$ **to** $(i.ex \cup \{nHop(i)\});$
$[] \ diff(i) \wedge i.AS\text{-}path = \emptyset \ \rightarrow$
send $UPDATE(WD(i), i.pocw, i, posw, i.poas, i.pocf, i.ponj, i.sn)$ **to** $(i.ex \cup \{nHop(i)\})$
fi
if $i.seqChg \ \rightarrow \ i.sn := i.sn + 1$ **fi**
$i.poas, i.pocf, i.pocw, i.ponj, i.sn := \emptyset, \emptyset, \emptyset, \emptyset, i.sn + 1;$
$i.change, i.seqChg, i.suspect, i.adv, i.pAdv, i.advd := false, false, false, false, false, false$
fi
[]
$\langle A7\rangle :: \ i.SN \neq \emptyset \longrightarrow (\forall k, s, t : \langle k, s, t\rangle \in i.SN \wedge CLOCK > t + T_d : i.SN := i.SN \setminus \{\langle k, s, t\rangle\})$

Figure 5: Module ADV-RESET

- $i.pocw$, $i.posw$, $i.poas$, $i.pocf$, and $i.ponj$ denote the point(s) of AS-failure, point(s) of channel-failure, point(s) of channel-withdrawal, point(s) of join-change, and point(s) of segment-withdrawal that i has respectively.

- $i.AS\text{-}path$ denotes the current route of i, and $i.inval$ denote the set of import neighbors of i whose routes are invalid.

- $i.sn$ denotes the local sequence number of i, and $i.SN$ contains the sequence number of the latest information about faults with respect to other ASes that has reached i. An element in $i.SN$ that records the sequence number for the latest information about faults with respect to an AS j is deleted from $i.SN$, if i has not received from j any information about faults for any period of T_d time.

- $i.change$ denotes whether the network state has changed, $i.seqChg$ denotes whether the sequence number of i needs to increase after i adapts its route to network state changes; $i.suspect$ denotes whether i is in the process of resolving some uncertainty, $i.adv$ denotes whether i is going to send out UPDATE messages, $i.pAdv$ denotes whether i will send out an withdrawal-UPDATE message that piggybacks information about faults and/or route changes before i tries to resolve some uncertainty, and $i.advd$ denotes whether i has sent out a withdrawal-UPDATE after $i.pAdv$ is set to $true$.

$i.pocw$, $i.posw$, $i.poas$, $i.pocf$, $i.ponj$, $i.AS\text{-}path$, $i.inval$, and $i.SN$ are initialized to $\emptyset$; $i.change$, $i.seqChg$, $i.suspect$, $i.adv$, and $i.pAdv$ are initialized to $false$.

Moreover, for every neighboring node/AS j of i, i uses $i.j.AS\text{-}path$ and $i.j.sn$ to maintain a local copy of the value of $j.AS\text{-}path$ and $j.sn$ respectively. For convenience, $i.im$ and $i.ex$ are used to denote the set of import neighboring ASes and the set of export neighboring ASes of the AS of i; temporary variables r', n, k', k'', e, l, s, t, t', and wt are also used.

Actions. For clarity of presentation, we define the following notations:

$WD(i)$	:	used in a BGP UPDATE message to $withdraw$ the last route i advertised to its export neighbors;
$nHop(j,r)$	:	the AS in route r that is one-hop closer to d than j is in r, i.e., the next hop of j in route r;
$nHop(i)$	:	$nHop(i, i.AS\text{-}path)$;
$preRoute(i)$	:	the last route used by i before it adopts its current route;
$preNHop(i)$	:	$nHop(i, preRoute(i))$, If $preRoute(i) = \emptyset$ (i.e., i has no route previously), $preNHop(i) = \perp$;
$CLOCK$	:	the current value of system clock at an AS;
$Invalid(r,i)$	:	$r = \emptyset \vee (\exists n : n \in r \wedge n \in i.poas) \vee (\exists e : e \in r \wedge e \in (LK(i.pocf) \cup LK(i.pocw) \cup LK(i.posw))$, where $LK(i.pocf) = \{\langle a,b\rangle : \langle\langle a,b\rangle, t'\rangle \in i.pocf\}$, $LK(i.pocw) = \{\langle a,b\rangle : \langle\langle a,b\rangle, s'\rangle \in i.pocw\}$, $LK(i.posw) = \{\langle a,b\rangle : \langle\mathbb{S},\mathbb{S}',b,a,i',t,s\rangle \in i.posw\}$
$l.gAS$	:	k if l is $\langle k,s\rangle$, $\langle\langle k',k\rangle,t,s\rangle$, $\langle\langle k',k\rangle,s\rangle$, $\langle\mathbb{S},\mathbb{S}',k,k',i',t,s\rangle,s\rangle$, or $\langle k, ASes, t, s\rangle$;
$l.ASes$	:	$ASes$ if l is a point of segment-withdrawal $\langle ASes', ASes, \mathcal{I}, \mathcal{J}, i', t', s\rangle$ or a point of node-join $\langle n, ASes, t', s\rangle$;
$l.L$	:	$\langle k',k\rangle$ if l is $\langle\langle k',k\rangle,t,s\rangle$ or $\langle\langle k',k\rangle,s\rangle$;
$l.sn$ *or* $l.k.sn$	:	s if l is $\langle k,s\rangle$, $\langle\langle k',k\rangle,t,s\rangle$, $\langle\langle k',k\rangle,s\rangle$, or $\langle k, ASes, t, s\rangle$;
$l.k.sn$	:	s if l is $\langle v, ASes, t, s'\rangle$ and $\langle k,s\rangle \in l.ASes$;
$l.tPsd$	:	t' if l is a point of channel-failure $\langle\langle a,b\rangle,t',s\rangle$ or a point of node-join $\langle n, ASes, t', s\rangle$;
$IN(l,r)$	:	$l \in r.poas \vee l \in r.pocf \vee l \in r.pocw \vee l \in r.ponj$;
$Obsolete(r,i)$	:	$(\exists j, s : \langle j,s\rangle \in i.SN \wedge (\exists l : IN(l,r) \wedge l.gAS = j \wedge l.j.sn < s)) \vee$ $(\exists k, j : k \notin i.inval \wedge j \in r \wedge j \in k.AS\text{-}path \wedge nHop(j, k.AS\text{-}path) \neq nHop(j,r) \wedge$ $\neg(\exists l : IN(l,r) \wedge l.gAS = j))$;
$D(i.poas, r)$	:	$i.poas \setminus \{\langle k,s\rangle : \langle k,s\rangle \in i.poas \wedge k \in r\}$;
$D(i.pocf, r)$	:	$i.pocf \setminus \{\langle\langle k,k'\rangle,t,s\rangle : \langle\langle k,k'\rangle,t,s\rangle \in i.pocf \wedge \langle k,k'\rangle \in r\}$;
$D(i.pocw, r)$	:	$i.pocw \setminus \{\langle\langle k,k'\rangle,s\rangle : \langle\langle k,k'\rangle,s\rangle \in i.pocw \wedge \langle k,k'\rangle \in r\}$;
$D(i.posw, r)$	:	$i.posw \setminus \{\langle\mathbb{S},\mathbb{S}',k,k',i',t,s\rangle : \langle\mathbb{S},\mathbb{S}',k,k',i',t,s\rangle \in i.posw \wedge \langle k,k'\rangle \in r\}$;
$adpt(i.SN, r)$	:	$\{\langle k,s,t\rangle : \langle k,s,t\rangle \in i.SN \wedge \neg(\exists l : IN(l,r) \wedge l.k.sn > s)\} \cup$ $\{\langle k, l.k.sn, CLOCK\rangle : \langle k,s,t\rangle \in i.SN \wedge IN(l,r) \wedge s < l.k.sn\} \cup$ $\{\langle l.gAS, l.(l.gAS).sn, CLOCK\rangle : IN(l,r) \wedge \neg(\exists s,t : \langle l.gAS, s, t\rangle \in i.SN)\}$;
$adpt(i.sn, r, j)$	:	$\max\{i.sn, \max_s\{s : \langle\langle j,i\rangle,s\rangle \in r.pocw\}, \max_s\{s : \langle j, ASes, t, s\rangle \in r.ponj \wedge \langle i,s'\rangle \in ASes\}\}$; (note: $\max_s\{s : FALSE\} = -\infty$)
$chgSeq(i,r)$	:	$(\exists s, l : \langle\langle j,i\rangle,s\rangle \in r.pocw \vee (l \in r.ponj \wedge < i, s > \in l.ASes))$
$i.k.SN$	:	s if $\langle k,s,t\rangle \in i.SN$;
$M(i.poas, r.poas)$	:	$\{l : l \in (i.poas \cup r.poas) \wedge l.sn = i.(l.gAS).SN\}$;
$M(i.pocf, r.pocf)$	:	$\{l : l.sn = i.(l.gAS).SN \wedge l \in (i.pocf \cup r.pocf) \wedge \neg(\exists l', l'' : l' \in i.pocf \wedge l'' \in r.pocf \wedge l' \neq l'' \wedge l'.L = l''.L = l.L)\} \cup$ $\{\langle e, \max\{t1,t2\}, s\rangle : l' \in i.pocf \wedge l'' \in r.pocf \wedge e = l'.L = l''.L \wedge l'.sn = l''.sn = i.(l'.gAS).SN\}$
$M(i.pocw, r.pocw)$	:	$\{l : l \in (i.pocw \cup r.pocw) \wedge l.sn = i.(l.gAS).SN\}$;
$M(i.posw, r.posw)$	:	$\{l : l \in (i.posw \cup r.posw) \wedge l.sn = i.(l.gAS).SN\}$;
$M(i.ponj, r.ponj)$	:	$\{l : l \in (i.ponj \cup r.ponj) \wedge l.sn = i.(l.gAS).SN \wedge \neg(\exists l' : l' \in (i.ponj \cup r.ponj) \wedge l'.gAS = l.gAS)\} \cup$ $\{l : l \in (i.ponj \cup r.ponj) \wedge l.sn = i.(l.gAS).SN \wedge (\exists l' : l' \in (i.ponj \cup r.ponj) \wedge l'.gAS = l.gAS \wedge l'.sn < l.sn)\} \cup$ $\{\langle n, l.ASes \cap l'.ASes, \max\{l.tPsd, l'.tPsd\}, l.sn\rangle : l \in i.ponj \wedge l' \in r.ponj \wedge$ $l.gAS = l'.gAS = n \wedge l.sn = l'.sn = i.n.SN\}$
$mPref(i)$	:	the route of an import neighbor of i that is valid and highest ranked at i; $mPref(i) = \perp$ if none of the import neighbor of i has a valid route;
$nsd(nHop(i), l)$	:	node $nHop(i)$ does not send the fault information l to i;
$Suspect(k', i)$	:	$k' \in i.AS\text{-}path \wedge ((\exists k'', t' : \langle\langle k', k''\rangle, t'\rangle \in i.pocf) \vee (\exists l, s, s' : l \in i.ponj \wedge \langle k', s\rangle \in l.ASes \wedge l.gAS \notin i.AS\text{-}path \wedge$ $nsd(nHop(i), l)) \vee (\exists l : l = \langle\mathbb{S},\mathbb{S}',\mathcal{I},\mathcal{J},i',t,s\rangle \wedge l \in i.posw \wedge k' \in \mathbb{S}' \wedge [k',\mathcal{I},\mathcal{J}] \in mPref(i) \wedge nsd(nHop(i), l)))$, i.e., $Suspect(k', i) = true$ if k' is a suspected AS for i;
$modified(i)$	:	variable $i.AS\text{-}path$ is modified the current instance of $A4$ execution;
$MRAI$	:	the MRAI timer used in BGP;
$Pol(k', i)$	:	a point of channel-failure $\langle\langle k', n\rangle, t\rangle\rangle \in i.pocf$ such that $t = \max\{l.tPsd : l \in i.pocf \wedge (\exists k'', t' : \langle\langle k', k''\rangle, t'\rangle = l)\}$;
$Poj(k', i)$	:	a point of node-join l' such that $k' \in l'.ASes$ and $l'.tPsd = \max\{l.tPsd : l \in i.ponj \wedge k' \in l.ASes\}$;
$hops(k', i)$	:	the number of hops between k' and i in $i.AS\text{-}path$
$wtPol(k', i)$	:	$hops(k', nHop(i)) \times (T_m + U_d) - Pol(k', i).tPsd$
$wtPoj(k', i)$	:	$hops(k', i) \times (T_m + U_d) - Poj(k', i).tPsd$
$maxWait(i)$	:	$\max\{\max\{wtPol(k', i) : Suspect(k', i) \wedge Pol(k', i) \neq \perp\}, \max\{wtPoj(k', i) : Suspect(k', i) \wedge Poj(k', i) \neq \perp\}\}$
$advInfo(i)$	:	$i.suspect \wedge Invalid(preRoute(i), i) \wedge$ $\neg(\exists k' : k' \in preRoute(i) \wedge ((Suspect(k', i) \wedge wtPol(k', i) = maxWait(i)) \vee$ $(\exists k'' : Suspect(k'', i) \wedge k' \in Poj(k'', i).ASes \wedge wtPoj(k'', i) = maxWait(i))))$
$mrai(i)$	:	i did not send any UPDATE message within the past MRAI time;
$diff(i)$	:	$i.AS\text{-}path$ differs from the last route i has used;
$\mathbb{S}(i)$	:	the set of export neighboring ASes of i where there is no node whose route goes through $[i.AS, preNhop(i)]$;
$\mathbb{S}'(i)$	:	the set of export neighboring ASes of i where there may be no node whose route goes through $[i.AS, preNhop(i)]$;
$addPoj(i)$	:	$\{\langle i.AS, j, i, 0, i.sn\rangle : j \in i.ex \wedge j \neq nHop(i)\}$.

G-BGP consists of three submodules: FAULT-INFO, ROUTE-ADAPT, and ADV-RESET, which are shown in Figure 3, 4, and 5 respectively:

- FAULT-INFO consists of actions $A1$, $A2$, and $A3$ that generate and propagate information about faults and network state changes, and it implements the ideas of numbered-grapevining and obsolete information removal as discussed in Section 3.2.1 and 3.2.2.
- ROUTE-ADAPT consists of actions $A4$ and $A5$ that adapt the routes of ASes according to faults and state changes, and it implements the ideas of uncertainty resolution as discussed in Section 3.2.3.
- ADV-RESET consists of actions $A6$ and $A7$ that send out BGP UPDATE messages and reset protocol variables.

3.4 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 suspected 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 Policy graph: concepts and properties

In this section, we first define the concept of policy graph for modelling inter-AS routing, then we present some properties of policy graph.

4.1 Concepts

In inter-AS routing, both network topology and export as well as import policies of nodes affect the routes available in a network. We define the concept of *policy graph* to model the above three aspects of a network. Policy graph is used to analyze the convergence properties of G-BGP and BGP in Section 5.

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

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

4.2 Properties of policy graph

In this subsection, we present the complexity of computing the policy graph at a state q and an observation of the structure of policy graph.

Computational complexity. To compute the policy graph $G_p.q(V_p.q, E_p.q)$ for a network G and destination d at state q, we use the breadth first graph search algorithm [7]: the search starts with d and visits every node in $G.q$ one by one; when the searching process visits a node i, i is added to $V_p.q$, the export neighbors of i in the policy graph (i.e., $EX(i, G_p.q)$) are calculated by checking the routing policy of i and its neighbors' in $G.q$, the set of edges $\{\langle i, j\rangle : j \in EX(i, G_p.q)\}$ is added to $E_p.q$, and every node in $EX(i, G_p.q)$ that has not been visited is added to the list of nodes to be visited (see Figure 6 for detailed description of the algorithm). Since the breadth first graph exploration of $G.q$ takes $O(|V.q|+|E.q|)$ time, the above procedure

```
Policy-graph(G.q, d, P.q)
  do each j ∈ V.q ⟶
      j.color := white;
  od
  Q := {d};
  V_p.q, E_p.q := ∅, ∅;
  do Q ≠ ∅ ⟶
      j := Q.head;
      V_p.q := V_p.q ∪ {j};
      do each i such that (j, i) ∈ E.q ⟶
          if j exports its route to i ∧ i imports routes from j →
              E_p.q := E_p.q ∪ {⟨j, i⟩};
              if i.color = white → i.color := black; Enqueue(Q, i) fi
          fi
      od
      Dequeue(Q);
  od
  return (V_p.q, E_p.q)
```

Figure 6: Algorithm to compute policy graph $G_p.q(V_p.q, E_p.q)$

to compute policy graph takes $O(|V.q| + |E.q|)$ time too[6]. This result is formalized in Proposition 1.

As for complexity in computing a policy graph, we have

Lemma 1 (Complexity of computing policy graph) *It takes $O(|V.q| + |E.q|)$ time to compute the policy graph $G_p.q(V_p.q, E_p.q)$ at a network state q.*

This is in contrast to the exponential computational complexity for dispute digraph, which is used in [12] and [22].

Structural property. The structure of an policy graph depends on the network topology and routing policy adopted at each AS. We analyze them as follows.

Due to historical and commercial reasons, the Internet topology is a hierarchical one with meshed interconnections among entities at various tiers [16, 9, 18]. On one hand, different types of ISPs, such as local ISPs, regional ISPs, national ISPs, and transit (or international) ISPs, with customer-provider relationship provide network infrastructures to form the Internet, and the resulted Internet takes the form of a hierarchy of tiers in the sense that networks or ASes of local ISPs attach to those of its regional ISPs, ASes of regional

[6]In contrast, it takes exponential time to compute the dispute digraph which is used to analyze convergence speed of BGP in [22].

ISPs attach those of its national ISPs, and ASes of national ISPs attach to those of transit ISPs. Thus, the tiers of the Internet hierarchy from lower to higher tiers are ASes of local ISPs, regional ISPs, national ISPs, and transit ISPs. On the other hand, due to business pressure, ASes of ISPs form a peering relationship with those of other ISPs at their geographical neighborhood to provide transit service for one another[7], thus a rich mesh of interconnection at various tiers (i.e., local, regional, national, and transit ISPs) exists [16]. Therefore, the policy graphs for the Internet is a hierarchical one with a rich mesh of interconnection at each tier of the hierarchy.

Moreover, [18] found that lower tier ISPs tend to possess a higher degree of peering interconnectivity than higher tier ISPs, which means that the peering-meshes among lower tier ISPs tend to be more connected than those among higher tier ISPs. However, [11] found that the average degree of AS-level Internet topology is small[8] and is between 2.6 and 2.9. Therefore, the average degree of AS is small, and the higher an AS is in the Internet hierarchy the smaller its degree tends to be. And these properties hold for the policy graphs of the Internet too.

In terms of routing policy, most ASes today import every route they hear from its neighboring ASes and do not impose any filtering [18]. Export policy adopted at each AS depends on its relationship with its neighboring ASes: an AS i exports to its provider ASes and peering ASes only the set of routes that either belong to i or are received from the customer ASes of i; i exports every route it knows to its customer ASes [19]. Therefore, given the destination d, the policy graphs of the Internet are directed hierarchical ones: starting at d, the directed edges go upwards first to reach the provider ASes PI of d, then to provider ASes of PI, and so on until reaching the transit ASes T; then the directed edges go downwards to reach the customer ASes CT of T, then to the customer ASes of CT, and so on until reaching the local ASes L; for the set of ASes that are either direct or indirect providers of d, there are bidirectional edges between peers or customer-provider pairs; for the set of ASes that are neither direct nor indirect providers of d, usually there is no edge between peers and only directed edges from a provider to its customers.

Therefore, the policy graphs of the Internet are directed hierarchical ones with meshed interconnections at various tiers of the hierarchy, and the average degree of ASes in the meshes of lower tiers tend to be larger than that of higher tiers. An example 3-tier policy graph is shown in Figure 7.

The above observations are formalized in Proposition 1 as follows.

Proposition 1 (Directed hierarchy of policy graph) *The policy graphs of the Internet are directed hierarchical ones with meshed interconnections at various tiers of the hierarchy, and the average degree of ASes in the meshes of lower tiers tend to be larger than that of higher tiers.*

[7] This is usually achieved through Network Access Points (NAPs), which are also referred to as Commercial Internet Exchanges (CIXs), Metropolitan Area Exchanges (MAEs), or Federal Internet Exchanges (FIXs) according to contexts.

[8] More specifically, 87% of ASes have degrees between 1 and 3, 9% of ASes have degrees between 4 and 9, 3.1% ASes have degrees between 10 and 27, and 0.9% of ASes have degrees larger than 28.

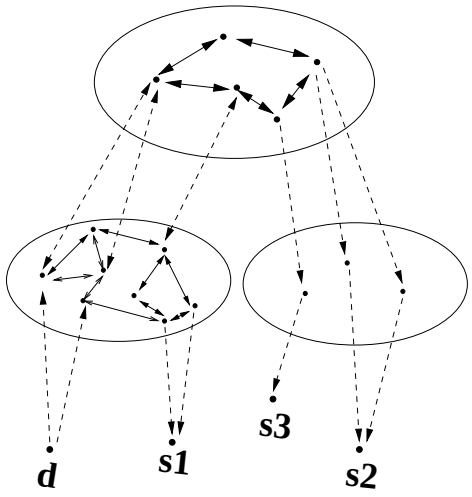


Figure 7: An example 3-tier policy graph

5 Analysis of G-BGP

Given a network topology $G'(V', E')$ and a set P' of routing policies for nodes in V', we let $\mathcal{L} \equiv (\forall i : i \in V' \Rightarrow LH.i)$, where $LH.i$ is defined as

$(i = d \Rightarrow i.AS\text{-}path = \emptyset) \wedge$
$(i \neq d \Rightarrow ((d \in V' \Rightarrow i.AS\text{-}path = \langle i.AS, mPref(i)\rangle) \wedge (d \notin V' \Rightarrow i.AS\text{-}path = \emptyset)))$
where
$mPref(i) =$ the highest ranked candidate route of i.

Then, every state in $\mathcal{L}$ is a state where each node in V' has chosen its highest-ranked candidate route, and every state in $\mathcal{L}$ is a stable state of G-BGP where no action of G-BGP is enabled.

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; and T_{change} occurs when d is up, but some node needs to change route as a result of some fault. 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.

5.1 Convergence stability

In the case of T_{down}, we have

Lemma 2 (Convergence stability after T_{down}) *When T_{down} occurs, for both the SPF and non-SPF policies, G-BGP converges with no fault-agnostic or distribution-inherent instability;*[9] *Both fault-agnostic and distribution-inherent instability can occur during BGP convergence.*

[9] *The fact that G-BGP converges is proved in Section 5.2.*

Proof: When T_{down} occurs, there are several different cases: whether d announces that it will fail-stop before it actually fail-stops, and whether d has one or more than one export neighbors before it fail-stops. We call the case of T_{down} where d withdraws its address prefix or announces that it will fail-stop before it actually fail-stops as *graceful* T_{down}, and the case of T_{down} where d does not announce that it will fail-stop before it fail-stops as *gross* T_{down}.

We analyze the convergence property of G-BGP as follows.

- In the case of graceful T_{down}, a point of AS-failure $\langle d \rangle$ will be propagated along UPDATE messages. When an AS i receives an UPDATE message with the point of AS-failure $\langle d \rangle$, i will add $\langle d \rangle$ to $i.poas$ by executing action $A3$. Since the current route and all the candidate routes of i include d, the execution of action $A4$ at i will invalidate the current route and every candidate route of i. Therefore, after an AS i receives an UPDATE message, i will withdraw its route to d and set $i.AS\text{-}path$ to $\emptyset$ by executing action $A6$. Moreover, when i withdraws its route by action $A6$, i also propagates the point of AS-failure $\langle d \rangle$ to its export neighbors.

  The above situation happens to every AS i in the network. Therefore, all the UPDATE messages that are propagated in the network after T_{down} are withdrawal-UPDATE messages. Therefore, every AS will only change (i.e., withdraw) its route only once before the network converges. Thus no fault-agnostic or distribution-inherent instability can occur during G-BGP convergence.

- In the case of gross T_{down} where d only has one export neighbor, a point of channel-failure $\langle \langle d, d' \rangle, timePassed \rangle$ will be propagated, where d' is the only export neighbor of d. For every AS i in the network, its current route and candidate routes must all go through link $\langle d, d' \rangle$. Therefore, the convergence behavior of G-BGP in this case is the same as that in graceful T_{down}, and no fault-agnostic or distribution-inherent instability can occur during G-BGP convergence.

- In the case of gross T_{down} where d has multiple export neighbors $d'_0, \ldots, d'_m$, multiple points of channel-failures $\langle \langle d, d'_v \rangle, timePassed \rangle$ ($v = 0, \ldots, m$) will be propagated. For every AS i, its current route and candidate routes must go through link $\langle d, d'_v \rangle$ for some $v \in [0, m]$. When i receives an UPDATE message, it will add the point(s) of channel-failures and point(s) of route-changes carried in the UPDATE message to $i.pocf$ and $i.pocw$ respectively by executing action $A3$. Then, after executing action $A4$, if there is still some candidate route r_j that has not been invalidated and goes through link $\langle d, d'_j \rangle$, i will enter the process of resolving uncertainty between link-failure and node-failure to check whether link $\langle d, d'_j \rangle$ has fail-stopped too. During the waiting period of uncertainty resolution at i, the point of channel-failure $\langle \langle d, d'_j \rangle, timePassed \rangle$ will reach i and be added to $i.pocf$, after which i will invalidate route r_j by executing action $A4$. This process will continue until there is no valid candidate route for i any more, at which point i will withdraw its current route and propagates the set of points of channel-failures it has learned to its export neighbors.

  The above situation happens to every AS i in the network. Therefore, all the UPDATE messages that are propagated in the network after T_{down} are withdrawal-UPDATE messages. Therefore, every AS will only change (i.e., withdraw) its route only once before the network converges. Thus no fault-agnostic or distribution-inherent instability can occur during G-BGP convergence.

In BGP, both fault-agnostic and distribution-inherent instability may happen in BGP after T_{down}, as discussed in Section 3.1 and [17].

□

In the case of T_{up}, we have

Lemma 3 (Convergence stability after T_{up}) *When T_{up} occurs,*

(i) For the SPF policy, G-BGP and BGP converges with no fault-agnostic instability. Furthermore, if message passing delay is proportional to the number of hops a message passes, G-BGP and BGP converge with no distribution-inherent instability.

(ii) For non-SPF policies, G-BGP converges with no fault-agnostic instability, but fault-agnostic instability can occur in BGP; distribution-inherent instability may happen in both G-BGP and BGP.

Proof: In the case of the SPF policy, whenever an AS i changes its route in G-BGP or BGP, it changes to a shorter route. Therefore, if i changes its route from that learned from one of its import neighbor j to that learned from another import neighbor j', no matter the current knowledge of i with respect to the actual AS-path of the route via j' is correct or not, the route via j' will always be shorter than that via j unless distribution-inherent instability occurs at i. Thus, if distribution-inherent instability does not occur at i, the route i chooses in the final stable network state will be via j' instead of j. Therefore, fault-agnostic instability will not happened in G-BGP and BGP. Furthermore, if message passing delay is proportional to the number of hops a message has passed, then an AS i always learns the shortest path from d to i first. Therefore an AS i will not change its route anymore once it has learned a route which is a shortest path route to d in the case of the SPF policy. Thus, there is no instability incurred in both G-BGP and BGP in this case.

We prove that, if message passing delay is not proportional to the number of hops a message has passed, distribution-inherent instability can happen in BGP and G-BGP in the case of the SPF policy as follows. We consider two ASes i and i', with i being farther away from d than i' is. Suppose i' can reach d via two routes $r1$ and $r2$, with $r2$ longer than $r1$. However, i learns route $r2$ earlier than $r1$ due to different delay along different routes. Later, i learns from one of its import neighbor j a route that contains $r2$ and sets $[i, j, \ldots, i', r2]$ as its route to d. After this, i' learns $r1$ and changes its route to $r1$, and thus the kind of distribution-inherent instability where the adopted route is valid but transient happens at this moment. Suppose that i learns another route $r3$ via another import neighbor j' before i learns the route change at i', and that $r3$ is shorter than $[i, j, \ldots, i', r2]$ but longer than $[i, j, \ldots, i', r1]$. Then i will change its route to $r3$ even though $r3$ is longer than $[i, j, \ldots, i', r1]$. Later, j informs i of the newly learned route $[i, j, \ldots, i', r1]$, and i changes its route again to $[i, j, \ldots, i', r1]$ that go through the import neighbor j. Thus the kind of distribution-inherent instability where the adopted route is invalid happens here.

The kind of distribution-inherent instability where the adopted route is valid but transient is even more likely to happen in G-BGP and BGP in the case of non-SPF policy than in the case of the SPF policy, because more preferred route of an AS is more likely to be formed later than some less preferred route of the AS in the case of non-SPF policy. Since the kind of distribution-inherent instability where the adopted route is invalid is caused by fault-agnostic instability and the kind of distribution-inherent instability where the adopted route is valid but transient, the increase in the likelihood of the kind of distribution-inherent instability where the adopted route is valid but transient also increases the likelihood of the kind of distribution-inherent instability where the adopted route is invalid.

In G-BGP, whenever the state of an AS i changes (e.g., changing route, associated links fail-stopping), information about the state change (e.g., point of AS-failure, point of channel-failure, point of channel-withdrawal, and point of node-join) will be piggybacked in the first message that is sent out from i. Therefore, when another AS j changes its state due to the state change at i, j knows the exact change at i or can resolve certain uncertainties, and will not use information that is invalidated by the state change at i. Thus fault-agnostic instability is avoided in G-BGP. However, this is not the case in BGP. One example is as follows. Suppose there exists an AS i that has two import neighbors j and j' whose routes go through the same AS k other than i, j, and j'. At certain moment t_0, i chooses the route learned from j as its route to d; sometime later, k changes its route and j changes the route it advertised to i accordingly, then i may choose the route it previously learned from j' as its new route in BGP, and fault-agnostic instability is incurred. (This will not happen in G-BGP because information about the change at k will be propagated to i and i will learn that the route it previously received from j' is already invalid.)

□

In the case of T_{change}, we have

Lemma 4 (Convergence stability after T_{change}) *When T_{change} occurs, for both the SPF and non-SPF policies, G-BGP converges with no fault-agnostic instability, but fault-agnostic instability can occur in BGP convergence; distribution-inherent instability can occur in G-BGP and BGP.*

Proof: In G-BGP, whenever the state of an AS i changes (e.g., changing route, associated links fail-stopping), information about the state change (e.g., point of AS-failure, point of channel-failure, point of channel-withdrawal, and point of node-join) will be piggybacked in the first message that is sent out from i. Therefore, when another AS j changes its state due to the state change at i, j knows the exact change at i or can resolve certain uncertainties, and will not use information that is invalidated by the state change at i. Thus fault-agnostic instability is avoided in G-BGP. However, this is not the case in BGP, as shown in the proof for Theorem 3.

□

Lemmas 2, 3, and 4 imply

Theorem 2 (fault-agnostic-instability freedom in G-BGP) *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.*

Theorem 3 (Fault-agnostic instability in BGP) *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 2 and 3, 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 simulation in Section 6.

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 4 (Distribution-inherent-instability freedom in G-BGP after T_{down}) *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.*

Proof: This claim holds as a result of Lemma 2.

□

We summarize the stability during G-BGP and BGP convergence in Table 1.

Stability		SPF Policy		Non-SPF Policy	
		FAI	DII	FAI	DII
T_{down}	G-BGP	*No*	*No*	*No*	*No*
	BGP	*Possible*	*Possible*	*Possible*	*Possible*
T_{up}	G-BGP	*No*	*Possible*	*No*	*Possible*
	BGP	*No*	*Possible*	*Possible*	*Possible*
T_{change}	G-BGP	*No*	*Possible*	*No*	*Possible*
	BGP	*Possible*	*Possible*	*Possible*	*Possible*

Table 1: Stability during G-BGP and BGP convergence. In the table, FAI and DII denote fault-agnostic and distribution-inherent instability respectively.

5.2 Convergence speed

For convenience, we define the following notations:

- $\mathcal{R}(i, V, q)$: $\max_{j \in V} dist(i, j, q)$, where $dist(i, j, q)$ denotes 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 6.

In the event of T_{down}, we have

Theorem 5 (Convergence speed after T_{down}) *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) $\mathcal{R}(i, V.q_0, q_0) \leq \mathcal{D}(q_0) \leq \mathcal{LP}(V.q_0, q_0)$.

Proof: In the case of T_{down}, there are several sub-cases: whether d gracefully or grossly fail-stops, and whether d has one or more than one export neighbors before it fail-stops. We call the case of T_{down} where d gracefully fail-stops as *graceful* T_{down}, and the case of T_{down} where d grossly fail-stops as *gross* T_{down}.

(Note: given that intra-AS coordination is quick and the coordination time is bounded from above by certain small constant, and that we are interested in inter-AS coordination in inter-AS routing, the analysis of the paper focuses at the level of inter-AS coordination. Thus, for conciseness, the unit of consideration in our analysis is by an AS instead of a node.)

We first prove the convergence properties of G-BGP. As discussed in the proof for Theorem 2, all the UPDATE messages are withdrawal-UPDATE messages in G-BGP when T_{down} occurs. Therefore, once an AS i withdraws its route at some time, it will not change its route again. Thus, to deduce the time taken for G-BGP to converge to a state in $\mathcal{L}$, we only need to deduce the time taken for the last AS to withdraw its route to d after d fail-stops.

- In the case of gross T_{down} where d has a single export neighbor d', d' detects the fail-stop of link $\langle d, d' \rangle$ and withdraws its route to d since d' has no other candidate route to d. Then, d' sends out an point of channel-failure $\langle \langle d, d' \rangle, timePassed \rangle$ which is piggybacked in every UPDATE message. An AS i other than d' withdraws its route once it receives an UPDATE message, since the current route of i and all its candidate routes go through link $\langle d, d' \rangle$. Then, the time taken for G-BGP to converge in this case depends on the time taken for the last AS to first receive an UPDATE message.

  If the number of hops in the shortest path from d to an AS i in the policy graph $G_p.q_0$ is l_i and $l_i \geq 2$ (note: if $l_i \leq 1$, then i is either d or d' that does not receive any UPDATE message), then the time taken for i to first receive an UPDATE message is $\theta(l_i)$. We prove this claim by induction on l_i as follows:

  - Base: the claim trivially holds when $l_i = 2$, since every AS with l_i being 2 is an export neighbor of d'.
  - Hypothesis: the claim holds when $l_i = l$.
  - Induction: for every AS i with l_i being $l + 1$, it must have an import neighbor j with l_j being l. By hypothesis, j must have withdrawn its route and sends out an UPDATE message including the point of channel-failure within $\theta(l)$ time. Since i will receive the UPDATE message within U_d time after the message is sent out from j, i will receive the UPDATE message within $\theta(l+1)$, i.e., $\theta(l_i)$, time. Therefore, the claim holds when $l_i = l + 1$.

Since the maximum hops in the shortest path from d to an AS i in the policy graph $G_p.q_0$ is $\mathcal{R}(i, V.q_0, q_0)$, the time taken for G-BGP to converge in this case is $\theta(\mathcal{R}(i, V.q_0, q_0))$.

Moreover, given that it takes $\theta(\mathcal{R}(i, V.q_0, q_0))$ time for any information to travel from d to the node that is farthest from d, $\theta(\mathcal{R}(i, V.q_0, q_0))$ is the lower bound on the convergence time of any stateful routing protocols in the event of T_{down}. Thus $\theta(\mathcal{R}(i, V.q_0, q_0))$ is optimal convergence time achievable, and G-BGP convergence in an asymptotically optimal manner.

- In the case of graceful T_{down}, a point of AS-failure $\langle d \rangle$ is piggybacked in every UPDATE message. An AS i other than d and the export neighbors of d withdraws its route once it receives an UPDATE message, since the current route of i and all its candidate routes go through AS $\langle d \rangle$. Then, the time taken for G-BGP to converge in this case depends on the time taken for the last AS to first receive an UPDATE message. This is the same as in the case of gross T_{down} where d has a single export neighbor d'. Thus, G-BGP converges within $\theta(\mathcal{R}(i, V.q_0, q_0))$ time in this case too.

- In the case of gross T_{down} where d has multiple export neighbors, if $hops(d, i) = l'_i$ for an AS i and $l'_i \geq 1$ (note: if $l_i < 0$, then i is d), then the time taken for i to withdraw its route is $O(l'_i)$. We prove this claim by induction on l'_i as follows:

  - Base: this claim holds when $l'_i = 1$, since an AS i with l'_i being 1 must be an export neighbor of d and $i.d.ML = 1$, which means that actions $A2$ and $A4$ are executed with $i.suspect$ being $false$ within constant time, thus $O(1)$ time.
  - Hypothesis: the claim holds when $l'_i \leq l$.
  - Induction: for an AS i with $l'_i = l+1$, each of its import neighbors j must be such that $l'_j \leq l$ and j withdraws its route as well as sends out an withdrawal-UPDATE message within $O(l)$ time. After all of its import neighbors send out their withdrawal-UPDATE messages, i will execute actions $A3$ and $A4$ within U_d and thus $O(1)$ constant time, which means that i withdraws its route within $O(l) + O(1) = O(l + 1)$ time.
    Therefore, the claim holds when $l'_i = l + 1$.

  Thus, G-BGP converges within $O(D_(q_0))$ time in both the case of the SPF policy and the case of non-SPF policy.

  More tightly, we prove that G-BGP converges within $O(\max\{hops(d, i)+Dist(i) : \text{action } A5 \text{ is executed at } i\})$ time after T_{down} as follows:

  - For any AS i that executes action $A5$ and thus enters the process of resolving uncertainty between link failure and node failure, it withdraws its route within $O(hops(d, i))$ time. This claim can be proved by induction on $i.d.ML$ in the same way we prove that the time taken for an AS j to withdraw its route is $O(l'_j)$ in G-BGP if the maximum number of hops in any simple path from d to j in $G_p.d.q_0$ is l'_j. For clarity, we skip the proof here.
  - For an AS j where action $A5$ is not executed during G-BGP convergence, let i be the closest AS to j that has executed action $A5$, and let $dist(i, j)$ be the number of hops in the shortest path from i to j in $G_p.d.q_0$. Then, j will withdraw its route no later than $\theta(dist(i, j))$ time after i withdraws its route, since an withdrawal-UPDATE message will propagate to j no later than $\theta(dist(i, j))$ time after i withdraws its route. Therefore, j will withdraw its route within $O(hops(d, i) + dist(i, j))$ time after T_{down} occurs.

Therefore, all the ASes in the network will withdraw their routes within $O(\max\{hops(d,i)+Dist(i) :$ action $A5$ is executed at $i\})$ time after T_{down} occurs.

On the other hand, instead of converging within $\theta(\mathcal{R}(i, V.q_0, q_0))$ or $O(\mathcal{D}(q_0))$ time, BGP is proved to take up to $\theta(\mathcal{LP}(V.q_0, q_0))$ (thus $O(\mathcal{LP}(V.q_0, q_0))$) time to converge after T_{down} [18].

By the definitions of $\mathcal{R}(i, V.q_0, q_0)$, $\mathcal{D}(q_0)$, and $\mathcal{LP}(V.q_0, q_0)$, it is straightforward that $\theta(\mathcal{R}(i, V.q_0, q_0)) \leq O(\mathcal{D}(q_0) \leq \mathcal{LP}(V.q_0, q_0)$.

□

In the event of T_{up}, we have

Theorem 6 (Convergence speed after T_{up}) *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.*

Proof: [18] and [22] have proved that BGP converges from the initial state q_0 to a state q_0' in $\mathcal{L}$ within $\theta(\mathcal{R}(i, V.q_0', q_0'))$ time in the case of the SPF policy. This convergence result of BGP also applies to G-BGP since G-BGP behaves the same as BGP except that G-BGP is more conservative when changing routes such that fault-agnostic instability can be reduced in G-BGP. More formally, the real-time dispute digraph [22] for a network $G(V, E, P)$ in BGP is the same as that in G-BGP, therefore, the convergence result of BGP also applies to G-BGP.

Moreover, it is straightforward that the lower bound on convergence time in the event of T_{up} is $\theta(\mathcal{D}(q_0'))$, since it takes at least $\theta(\mathcal{D}(q_0'))$ time to form the longest route used by a node after convergence. When the SPF route ranking policy is used, $\mathcal{D}(q_0')$ equals to $\mathcal{R}(i, V.q_0', q_0')$. Therefore, the lower bound on convergence time is also $\theta(\mathcal{R}(i, V.q_0', q_0'))$ when the SPF policy is used. Thus, G-BGP and BGP are asymptotically optimal in convergence time in the event of T_{up}.

□

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 5 (Minimized affectation region for the SPF policy) *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}.*

Proof: We analyze the affectation region in both G-BGP and BGP under different fault scenarios when the SPF route ranking policy is used:

- In the case of an AS fail-stop, a link fail-stop, or the routing policy change at an AS, those ASes whose routes go through the fail-stopped AS, fail-stopped link, or withdrawn links before T_{change} have to change their routes since the AS or link(s) is(are) not up anymore. Therefore, these ASes compose the minimum affectation set MAS.

  The route of every AS i that is not in set MAS does not go through the fail-stopped AS, fail-stopped link, or withdrawn link(s) before T_{change}. i will not change its route after T_{change} since no AS in

MAS can offer i an shorter route than what i has before T_{change}. Therefore, i will not be affected during convergence of G-BGP and BGP.

Therefore, the affectation region in both G-BGP and BGP is minimized to be the minimum affectation set in the case of the SPF policy.

- In the case of an AS or a link join, those ASes whose routes can be shortened by the joining of the new AS or link have to change their routes, and these ASes compose the minimum affectation set MAS. For every AS i whose route cannot be shortened by the joining of the new AS or link will be change its route, since no import neighbor of i can offer i a shorter route. Therefore, the affectation region in both G-BGP and BGP is minimized in this case too.

□

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 7 (Convergence speed after T_{change}) *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.

Proof: To analyze the convergence time of G-BGP and BGP, we only need to compute the longest possible time taken for an AS to converge to its new state in $\mathcal{L}$. For convenience, we let $k' = pt(k, q_0, q_1)$.

- When the cause for the T_{change} is a graceful node fail-stop, a gross node fail-stop or a link fail-stop where the fail-stopped or suspected node i has a single export neighbor, we prove the claim by proving the fact that, for every affected AS k, it takes k $\theta(Tri(k, \mathcal{I}, q_0, q_1))$ time to converge to its new state in $\mathcal{L}$. We achieve this by induction on $hops(k, k'.AS, q_1)$.
  - Base: if $hops(k, k'.AS, q_1) = 0$ for an AS k, then the new route of k after T_{change} does not go through any affected AS or there is no new route for k. Therefore, the time taken for k to converge to its new state in $\mathcal{L}$ is equal to the time taken for the point of AS-failure (in the case of graceful node fail-stop), the point of channel-failure or the point of segment-withdrawal (in the case of gross node fail-stop with a single export-neighbor) to reach k, which

is $\theta(Tri(k, \mathcal{I}, q_0, q_1))$ since the number of hops between i and k is $\theta(Tri(k, \mathcal{I}, q_0, q_1))$ when $hops(k, k'.AS, q_1) = 0$, and there is no uncertainty to resolve and thus no waiting at ASes between i and k in the route of k before T_{change}. Thus, the claim holds for every AS k with $hops(k, k'.AS, q_1) = 0$.

- Hypothesis: the claim holds for every AS k with $hops(k, k'.AS, q_1) = h$ ($h \geq 0$).
- Induction: for every k with $hops(k, k'.AS, q_1) = h+1$ ($h \geq 0$), it must have an import neighbor k'' with $hops(k, k'.AS, q_1) = h + 1$. By hypothesis, k'' must have converged to its new state in $\mathcal{L}$ within $\theta(Tri(k'', \mathcal{I}))$ time. Within constant time after k'' converges to its new state in $\mathcal{L}$, an UPDATE message with the attached point of AS-failure or point of channel-failure will reach k from k'', at which point k learns its new route since there no need to resolve any uncertainty in the case of graceful AS fail-stop or gross fail-stop with a single export neighbor. Thus, k converges to its new state in $\mathcal{L}$ within $\theta(Tri(k'', \mathcal{I}) + 1)$ time, that is, $\theta(Tri(k, \mathcal{I}, q_0, q_1)$ time since $Tri(k, \mathcal{I}, q_0, q_1) = Tri(k'', \mathcal{I}) + 1$.

When the SPF route ranking policy is used, the minimum time required for a node j to converge is proportional to $Tri(j)$. Thus, the lower bound on the convergence time after the fault is $\theta(\max_j\{Tri(j) : j \in V.q_1 \wedge j.AS\text{-}path.q_1 \neq j.AS\text{-}path.q_0\})$. Therefore, G-BGP converges with an asymptotically optimal speed.

In BGP, when an AS i fail-stops, the set of ASes whose routes go through i have to change their routes. During convergence, fault-agnostic instability can be incurred and invalid route be explored. Therefore, in the worst case the time taken for BGP to converge is proportional to the length of the longest invalid route that may be explored [17, 18, 22]. In the case of T_{change}, the length of the longest invalid that may be explored is the number of hops $\mathcal{LP}(\{k : k \in V.q_0 \wedge k.AS\text{-}path.q_1 \neq k.AS\text{-}path.q_0\}, q_0)$ in the longest simple path in the subgraph of the policy graph (after faults) on the set of ASes that are affected (i.e., the affectation region). Therefore, it takes BGP $\theta(\mathcal{LP}(\{k : k \in V.q_0 \wedge k.AS\text{-}path.q_1 \neq k.AS\text{-}path.q_0\}, q_0))$ time to converge after T_{change} in worst cases such as when every affected AS has no route to reach d anymore or when the alternate route of the affected ASes are very long. Thus, it takes BGP $O(\mathcal{LP}(\{k : k \in V.q_0 \wedge k.AS\text{-}path.q_1 \neq k.AS\text{-}path.q_0\}, q_0))$ time to converge after T_{change}.

- When the cause for the T_{change} is the routing policy change at an AS i, there are two cases: i removes some of its import neighbors or i removes some of its export neighbors.[10]

  If i removes some of its import neighbors and the policy change does lead to an event T_{change}, then the current link e associated with i that i uses in forwarding traffic to d is withdrawn by i too. Then a point of channel-withdrawal with sequence number of i is propagated out from i to every other ASes whose route go through i or e before T_{change}. The convergence behavior of G-BGP in this case is the same as that in the case of gross AS fail-stop or a link fail-stop where the fail-stopped or suspected AS has a single export-neighbor.

  If i removes some of its export neighbors and results in an event T_{change}, then a set of points of channel-withdrawal for every link withdrawn will be propagated out from i reaches every AS whose

[10] Since the policy is fixed in the case of the SPF policy, we do not consider the case where an AS changes from one route to another route of equal length.

route goes through one of the withdrawn link before T_{change}. The convergence behavior of G-BGP in this case is the same as that in the case of graceful AS fail-stop.

Similar to G-BGP, the convergence behavior of BGP in the case of routing policy change at an AS is the same as that in the case of graceful AS fail-stop or gross AS fail-stop where the fail-stopped AS i has a single export neighbor. Therefore, it takes BGP up to $\theta(\mathcal{LP}(\{k : k \in V.q_0 \wedge k.AS\text{-}path.q_1 \neq k.AS\text{-}path.q_0\}, q_0))$ time to converge after the routing policy change at i.

- When the cause for the T_{change} is a gross node fail-stop or a link fail-stop where the fail-stopped or suspected node i has multiple export neighbors, the convergence behavior of G-BGP differs from that in the case of gross AS fail-stop or link fail-stop where the fail-stopped or suspected AS only a single export neighbor in the sense that some AS(s) relatively close to the fail-stopped or suspected may need to resolve the uncertainty between link-failure and node-failure by executing action $A5$. The uncertainty resolution procedure can introduce delay in G-BGP convergence if compared to the optimal achievable performance, even though the extra delay is small because the uncertainty is resolved locally around the fail-stopped AS or link. If an AS j executes action $A5$ to resolve uncertainty, the maximum delay j can introduce to G-BGP convergence is $O(hops(j, i.AS, q_0) - dist(i, j, q_0))$, where $dist(i, j, q_0)$ is the number of hops in the shortest path from i to j in the policy graph $G_p.q_0$. Since no two ASes where one AS is in the route of the other AS before or after T_{change} will both execute action $A5$ to resolve uncertainty with respect to the liveness of an suspected AS, and two ASes where neither is in the route of the other before or after T_{change} execute uncertainty resolution procedure in parallel, the overall maximum delay that can be introduced to G-BGP convergence is $O(\max_j\{hops(j, i.AS, q_0) - dist(i, j, q_0) : A5\ is\ executed\ at\ j\})$. Thus, G-BGP converges within $O(\max_k\{Tri(k, \mathcal{I}, q_0, q_1) : k\ is\ affected\} + \max_j\{hops(j, i.AS, q_0) - dist(i, j, q_0) : action\ A5\ is\ executed\ at\ i\})$ time. Since $hops(j, i.AS, q_0) - dist(i, j, q_0) \leq Tri(j)$ for every node j who executed action $A5$, G-BGP converges within $O(\max_k\{Tri(k, \mathcal{I}, q_0, q_1) : k\ is\ affected\})$ time, which has been shown to be asymptotically optimal.

  In BGP, the probability that fault-agnostic instability occur in the case of a gross AS fail-stop or a link fail-stop where the fail-stopped or suspected AS i has multiple export neighbors is much higher than that in the case of a gross AS fail-stop or a link fail-stop where the fail-stopped or suspected AS i has a single export neighbor. Therefore, it can take BGP up to $\theta(\mathcal{LP}(\{k : k \in V.q_0 \wedge k.AS\text{-}path.q_1 \neq k.AS\text{-}path.q_0\}, q_0))$ time to converge after a gross AS fail-stop or a link fail-stop where the fail-stopped or suspected AS i has multiple export neighbors.

- When the cause for the T_{change} is an AS or a link join, the set of ASes whose distance can be reduced by the joining of the new AS or link are affected, and only these ASes are affected in the case of the SPF policy. Therefore, the time taken for G-BGP and BGP to converge equals to the maximum time taken for an affected AS (i.e., some of whose nodes change route when the network converges from q_0 to q_1) to change its route to a shorter one which goes through the newly joined AS or link. The convergence behavior of G-BGP and BGP in this case is the same as that in the case of T_{up} where the destination is the newly joined AS i or the endpoint of the joining link that is in the route of the other endpoint, and the ASes in the network is the set of affected ASes. By Theorem 6, it takes G-BGP and BGP $\theta(\mathcal{R}(i, V.q_0, q_0)_i)$ time to converge, which is asymptotically optimal.

□

By Theorems 5, 6, and 7, 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 3, Lemma 5, and Theorem 7, 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 6) and the experimental observations [17, 18, 20] that BGP does not experience much delay in convergence when a node or a link joins.

We summarize the convergence speed of G-BGP and BGP in Table 2.

Speed			the SPF Policy	Non-SPF Policy
T_{down}	gross T_{down} with one ex-neighbor, graceful T_{down}	G-BGP	$\theta(\mathcal{R}(i, V.q_0, q_0))$, *optimal*	*same as left*
		BGP	$O(\mathcal{LP}(V.q_0, q_0))$	*same as left*
	gross T_{down} with multiple ex-neighbors	G-BGP	$O(D_(q_0))$	*same as left*
		BGP	$O(\mathcal{LP}(V.q_0, q_0))$	*same as left*
T_{up}		G-BGP	$\theta(\mathcal{R}(i, V.q_0', q_0')), optimal$	——
		BGP	$\theta(\mathcal{R}(i, V.q_0', q_0')), optimal$	——
T_{change}	node or link fail-stop, policy change	G-BGP	$\theta(\max_k\{Tri(k, \mathcal{I}, q_0, q_1) : k$ is affected$\})$ $optimal$	——
		BGP	$O(\mathcal{LP}(AR(q_0, q_1), q_0))$	——
	node or link join	G-BGP	$\theta(\mathcal{R}(i, AR(q_0, q_1), q_1))$, $optimal$	——
		BGP	$\theta(\mathcal{R}(i, AR(q_0, q_1), q_1))$, $optimal$	——

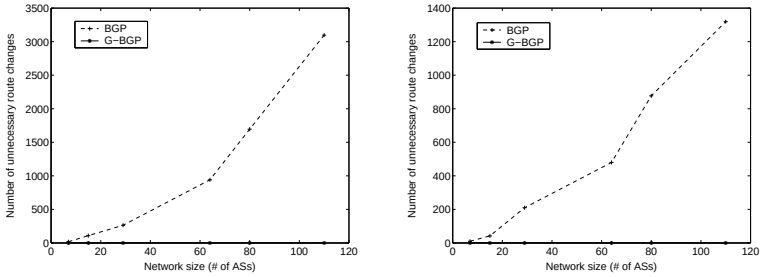
Table 2: Convergence speed of G-BGP and BGP. In the table, $\mathcal{R}(i, V.q_0', q_0') \leq D_(q_0) \leq \mathcal{LP}(V.q_0, q_0)$, and $\max_k\{Tri(k, \mathcal{I}, q_0, q_1) : k\ is\ affected\} \leq \mathcal{LP}(AR(q_0, q_1), q_0)$; $optimal$ in a box means that optimal convergence speed is achieved in G-BGP or BGP in the corresponding scenario.

6 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, 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[11]) into networks to simulate the events of T_{down}, T_{up}, and T_{change}. The simulation results are as follows.

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 8 and 9 respectively.

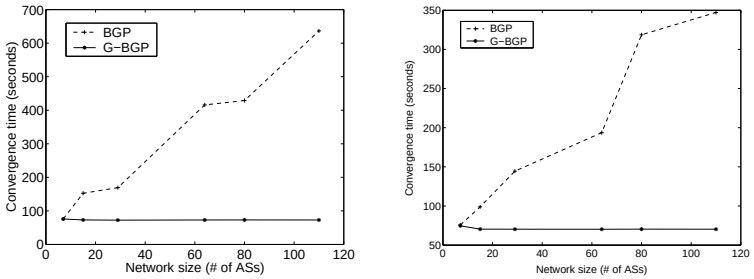
[11]The impact of link fail-stop and link join is reflected via node fail-stop and node join respectively. Thus we do not simulate link fail-stop or link join.



(a) SPF policy

(b) Randomized non-SPF policy

Figure 8: The number of unnecessary route changes after the destination d fail-stops



(a) SPF policy

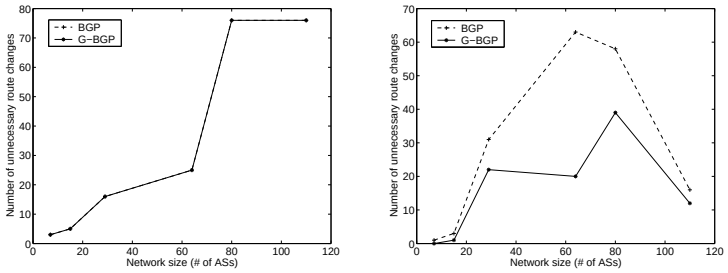
(b) Randomized non-SPF policy

Figure 9: The convergence time after the destination d fail-stops

We see that G-BGP converges with no unnecessary route changes, as proved in Corollary **??**. 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, G-BGP improves BGP convergence stability by a factor of 29.4 for the network with 115 ASes. We also observe that, 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.

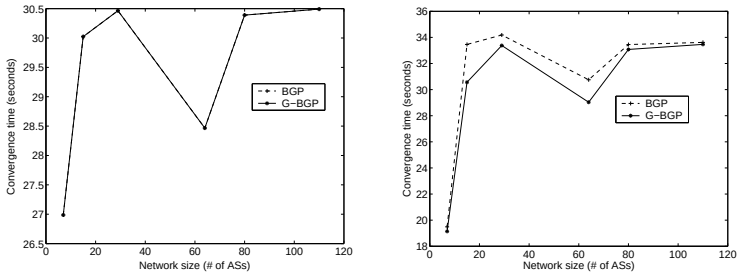
Event $\mathbf{T_{up}}$. When the destination d joins, the number of unnecessary route changes during convergence and the convergence time of G-BGP as well as BGP are shown in Figures 10 and 11 respectively.



(a) SPF policy

(b) Randomized non-SPF policy

Figure 10: The number of unnecessary route changes after the destination d joins



(a) SPF policy

(b) Randomized non-SPF policy

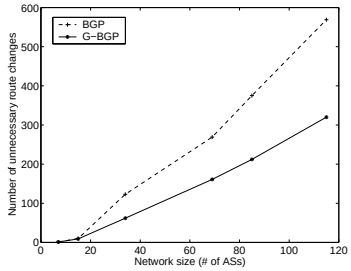
Figure 11: The convergence time after the destination d joins

We see that, when the SPF policy is used, the number of unnecessary route changes during convergence and the convergence time of BGP are the same as those of G-BGP, which is not unexpected since, as proved in Theorems 3 and 6, 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.

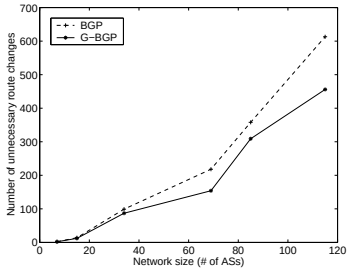
We also observe unnecessary route changes during G-BGP convergence, which is due to distribution-inherent instability. However, the time taken for G-BGP to converge is still quite short in spite of the distribution-inherent instability, which is similar to the observation in [20] that distribution-inherent 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 12 and 13 respectively.

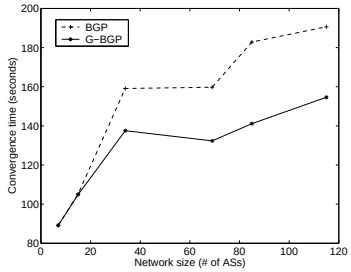


(a) SPF policy

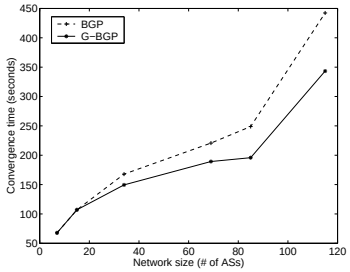


(b) Randomized non-SPF policy

Figure 12: The number of unnecessary route changes after a non-destination node fail-stops



(a) SPF policy



(b) Randomized non-SPF policy

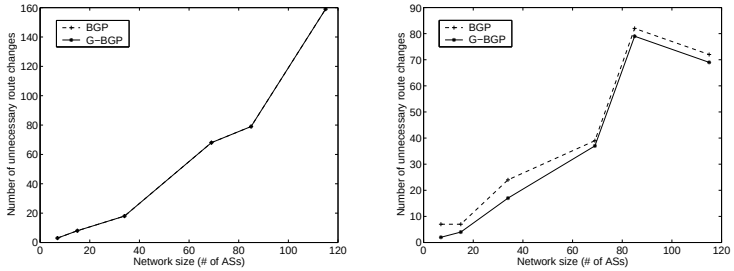
Figure 13: The convergence time after a non-destination node fail-stops

In this case, 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_{down}.

Event $\mathbf{T_{change}}$ when a node joins. When a non-destination node joins, the number of unnecessary route changes during convergence and the convergence time of G-BGP as well as BGP are shown in Figures 14 and 15 respectively.

In this case, 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 7.

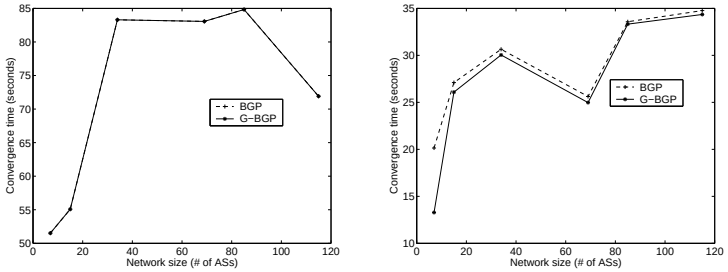
Event $\mathbf{T_{change}}$ when an AS changes routing policy. An AS may change its route ranking policy, import policy, and export policy. However, the effect of changing any of these policies is similar to each other, i.e., some node changes route. Therefore, we only simulate the case where an AS changes its export policy. When an AS changes its export policy, the number of unnecessary route changes during convergence and the convergence time of G-BGP as well as BGP are shown in Figures 16 and 17 respectively.



(a) SPF policy

(b) Randomized non-SPF policy

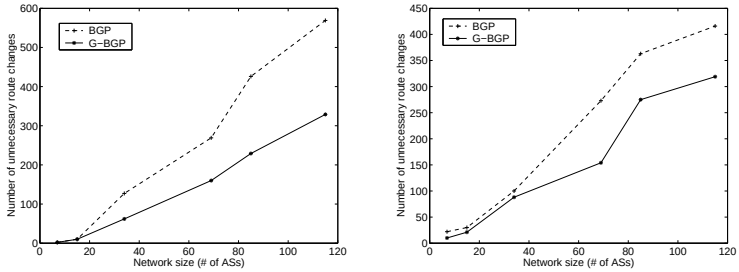
Figure 14: The number of unnecessary route changes after a non-destination node joins



(a) SPF policy

(b) Randomized non-SPF policy

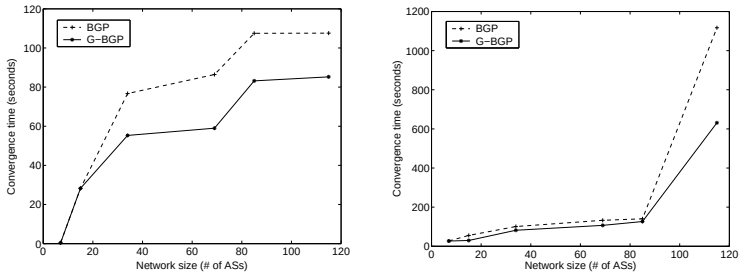
Figure 15: The convergence time after a non-destination node joins



(a) SPF policy

(b) Randomized non-SPF policy

Figure 16: The number of unnecessary route changes after an AS changes export policy



(a) SPF policy

(b) Randomized non-SPF policy

Figure 17: The convergence time after an AS changes export policy

We see that 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.

7 Discussion

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

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 [24] is used, because all the nodes in an AS maintain IBGP sessions with each other. On the other hand, if route reflection [5] or AS confederation [21] 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. (Interestingly, it has also been proven 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 [4].)

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 by BGP, G-BGP uses the following fault information: point of channel-withdrawal $POCW$, point of segment-withdrawal $POSW$, point of AS-failure $POAS$, point of channel-failure $POCF$, and point of node-join $PONJ$. Therefore, to implement G-BGP in a way that allows graceful migration of and interoperability with BGP, one key issue is to incorporate fault information into the existing BGP message format such that G-BGP and BGP can inter-operate.

In BGP [24], an UPDATE message has a variable-length field *Path Attributes* with a maximum length of 65,535 bytes. The *Path Attributes* field consists of a sequence of path attributes, such as AS_PATH. Each path attribute is a 3-tuple $<$*attribute type, attribute length, attribute value*$>$ of variable length. *Attribute Type* is a two-octet field that consists of the *Attribute Flags* octet followed by the *Attribute Type Code* octet, where Attribute Flags determine whether an attribute is *optional* or *well-known* and whether it is *transitive* or *non-transitive*. An attribute is optional if it is not required to be recognized by every router, and an attribute is transitive if it needs to be propagated by every router no matter whether the router recognizes the attribute.

We incorporate the $POCW$, $POSW$, $POAS$, $POCF$, and $PONJ$ values into the UPDATE messages of BGP by defining a new optional transitive path attribute FAULT_POINTS with type code 8. The Attribute Value for FAULT_POINTS consists of a sequence of fault information m whose format depends on the type of fault it conveys:

- If m is a point of channel-withdrawal, then it is a 7-octet field with the first octet being 2, the second and third octets being the ID of the AS that is one endpoint of the withdrawn link, the fourth and fifth octets being the ID of the AS that is the other endpoint of the withdrawn link, and the remaining two octets being the sequence number, i.e., $m =< 0, AS\text{-}id, AS\text{-}id, sn >$;

- If m is a point of segment-withdrawal, then it is a variable-length field with the first octet being 1 and the rest being $\langle Withdrawn\text{-}ASes, Suspected\text{-}ASes, AS\text{-}id, AS\text{-}id, BGP\text{-}id, t, sn \rangle$. The fields of $Suspected\text{-}ASes$ and $Suspected\text{-}ASes$ are two variable-length fields each of which has two sub-fields $\langle length, data \rangle$ where *length* is a 1-octet field specifying the length of $data$ in octets and *data* is a sequence of 2-octets for the IDs of the corresponding ASes; $AS\text{-}id$ is a 2-octet field; $BGP\text{-}id$ is a 4-octet field denoting the BGP-identifier of the node that generates the information; t is a 4-octet field denoting the time in microseconds that has passed since the information is generated; and sn denotes the sequence number of the AS that first sends out this point of segment-withdrawal, i.e., $m =< 1, \langle length, \langle AS\text{-}id \rangle^+ \rangle, \langle length, \langle AS\text{-}id \rangle^+ \rangle, AS\text{-}id, AS\text{-}id, BGP\text{-}id, t, sn >$.

- If m is a point of AS-failure, then it is a 5-octet field with the first octet being 0, the second and third octets being the ID of the AS that has fail-stopped, and the remaining two octets being the sequence number, i.e., $m =< 2, AS\text{-}id, sn >$;

- If m is a point of channel-failure, then it is a 11-octet field with the first octet being 1, the second and third octets being the ID of the AS that is suspected, the fourth and fifth octets being the ID of the AS that detects the link fail-stop, the following four octets being the time in microseconds that has passed since the detection of the link fail-stop, and the last two octets being the sequence number, i.e., $m =< 3, AS\text{-}id, AS\text{-}id, t, sn >$;

- If m is a point of node-join, then it is a variable-length field with the first octet being 4 and the rest being $\langle Suspected\text{-}ASes, BGP\text{-}id, t, sn\rangle$. $Suspected\text{-}ASes$ is a variable-length field with two subfields $\langle length, data\rangle$ where *length* is a 1-octet field specifying the length of $data$ in octets and *data* is a sequence of 2-octets for the IDs of the suspected ASes; $BGP\text{-}id$ is a 4-octet field; t is a 4-octet field denoting the time in microseconds that has passed since the detection of the AS-join; and sn denotes the sequence number of the AS that first sends out this point of node-join, i.e., $m =< 4, \langle length, \langle AS\text{-}id\rangle^{+}\rangle, BGP\text{-}id, t, sn >$.

(Remark: purging-messages and state-clarifiers used in uncertainty-resolution are incorporated, in a similar way, into BGP UPDATE messages as two optional transitive attributes.)

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 inter-operate 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 [3], 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.

8 Concluding remarks

The stability and speed of BGP convergence are closely related. To expedite BGP convergence and to avoid mis-interaction between convergence instability and instability-suppression mechanisms (such as route-flap-damping), 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. And this is enabled by (i) propagating necessary fault information to the affected nodes, (ii) enforcing a total

order on fault information regarding the same AS, and (iii) resolving uncertainty as to the state of other ASes. 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 maintained (unboundedly in time) 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 [3].

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] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. In *ACM SOSP*, pages 131–145, 2001.

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

[4] 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.

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

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

[7] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT Press and McGraw-Hill Book Company, 2000.

[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] R. Govindan and A. Reddy. An analysis of Internet inter-domain topology and route stability. In *IEEE INFOCOM*, pages 850–857, 1997.

[12] 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.

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

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

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

[16] G. Huston. Interconnection, peering and settlements: Part 1. *The Internet Protocol Journal*, pages 2–16, March 1999.

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

[18] 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.

[19] R. Mahajan, D. Wetherall, and T. Anderson. Understanding BGP misconfiguration. In *ACM SIGCOMM*, pages 3–16, 2002.

[20] 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.

[21] 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.

[22] D. Obradovic. Real-time model and convergence time of BGP. In *IEEE INFOCOM*, 2002.

[23] 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.

[24] 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.

[25] H. Tangmunarunkit, R. Govindan, S. Shenker, and D. Estrin. The impact of routing policy on Internet paths. In *IEEE INFOCOM*, pages 736–742, 2001.

[26] 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.