Neighbor-Specific BGP: More Flexible Routing Policies Without Sacrificing Global Stability

Yi Wang† Michael Schapira‡ Jennifer Rexford∗
∗Princeton University, Princeton, NJ, USA ‡Yale University, New Haven, CT, USA

ABSTRACT

The Border Gateway Protocol (BGP) offers network administrators considerable flexibility in controlling how traffic flows through their networks. However, the interaction between routing policies in different Autonomous Systems (ASes) can lead to protocol oscillation. The best-known sufficient conditions of BGP global routing stability impose restrictions on the kinds of local routing policies individual ASes can safely implement. In this paper, we present neighbor-specific BGP (NS-BGP), a modest extension to BGP that enables a much wider range of local policies without compromising global stability. Whereas a conventional BGP-speaking router selects a single “best” route (for each destination prefix), NS-BGP allows a router to customize the route selection on behalf of each neighbor. For example, one neighbor may prefer the shortest route, another the most secure route, and yet another the least expensive route. Surprisingly, we prove that the much more flexible NS-BGP is guaranteed to be stable under much less restrictive conditions on how routers “rank” the candidate routes. Moreover, unlike BGP, NS-BGP is able to support backup routing without oscillation and other anomalies such as “BGP wedgies”. We also show that it is safe to deploy NS-BGP incrementally, as a routing system with a partial deployment of NS-BGP is guaranteed to be stable, even in the presence of failure and other topology changes. In addition to our theoretical results, we also describe how NS-BGP can be deployed by individual ASes independently without changing to the BGP message format or collaboration from neighboring ASes.

1. INTRODUCTION

The Internet consists of tens of thousands of independently operated networks (known as Autonomous Systems, or ASes) that have different preferences for the kinds of paths that should carry their traffic. For example, an online gaming provider may prefer paths with low latency, whereas a financial institution may prioritize security over performance. Unfortunately, in today’s Border Gateway Protocol (BGP), each router selects and advertises a single best route, limiting an AS’s ability to offer customized route selection for its neighbors. As we show in this paper, with simple extensions to the protocol, a router could offer different interdomain routes to different neighbors. However, greater flexibility in selecting routes should not come at the expense of global stability—a perennial concern with today’s routing system. In this paper, we prove a surprising result: comparing to the conventional BGP, less restrictive conditions on local routing policies are sufficient to ensure global stability, when individual ASes are allowed to select different routes for different neighbors.

1.1 A Case for Neighbor-Specific BGP

In today’s BGP [1], each router selects a single best route (per destination) and only this route can be announced to its neighbors. Twenty years after BGP was first proposed, this “one-route-fits-all” design has become a frustrating limitation to Internet Service Providers (ISPs) that want to capitalize on their network connectivity by offering customized route selection service to their neighbors. We argue that such flexible route selection (which we dub “neighbor-specific BGP,” or “NS-BGP”) is beneficial for three main reasons:

- Many ISPs have rich path diversity. ISPs offering transit service usually connect to many other ASes, often at multiple locations [2,3]. As a result, it is quite common for large networks to have 5-10 paths per prefix, with some prefixes having more than 20 different paths [4].

- Different paths have different properties. The many alternative paths an ISP has could have different security [5] and performance [6] properties. In fact, alternative interdomain paths often have significantly better performance than the paths chosen by BGP [7].

- Different neighbors may want different paths. Different neighbors of an ISP may have very different preferences on the types of paths they get from the ISP. For example, financial institutions may prefer the most secure paths, while providers of interactive applications like online gaming and voice over IP may prefer paths with low latency. If such options were available, they might be willing to pay a higher price to have the paths they want. Yet some other neighbors may be perfectly happy with whatever paths the ISP provides for a relatively low price.

Ideally, an ISP would be able to offer different routes to different neighbors, regardless of whether they connect to the same edge router. Fortunately, such neighbor-specific route
selection is possible without changing the BGP message format or the way neighboring ASes exchange route announcements. As a result, an individual ISP can independently deploy NS-BGP and offer value-added route-selection services. All the changes required for an AS to deploy NS-BGP are within its own network and practically feasible, as discussed in Section 5.

1.2 Stability Concerns of Greater Flexibility

Despite the benefits of greater flexibility, enhancements to BGP should not come at the expense of global stability. In fact, even without neighbor-specific route selection, today’s BGP can easily oscillate, depending on the local policies ASes apply in selecting and exporting routes [8, 9]. Over the years, researchers have developed a reasonably good understanding of the trade-offs between local flexibility and global stability [10–13]. Rather than relying on Internet-wide coordination, researchers searched for practical constraints on local policies that would ensure global stability. In practice, policies are typically constrained by the business relationships between neighboring ASes [10]. For example, a customer AS pays its provider AS for connectivity to the rest of the Internet, whereas peer ASes carry traffic between their respective customers free of charge. These financial arrangements affect how ASes select and export routes, and how new relationships form:

- Prefer customer routes over peer or provider routes (preference condition): When selecting a route for a destination, an AS prefers a (revenue-generating) route through a customer over routes through a peer or provider.

- Export only customer routes to peers or providers (export condition): An AS can export routes through any neighbor to its customers, but can only export routes through its customers to its peers and providers. That is, an AS only provides transit services to its customers.

- No cycle of customer-provider relationships (topology condition): No AS is its own (direct or indirect) provider. That is, the AS-level topology does not contain any cycle of provider-customer edges.

Collectively, these three properties (known as the “Gao-Rexford conditions”) ensure the interdomain routing system converges to a stable state without global coordination [10].

The “Gao-Rexford” conditions reflect common business practices in today’s Internet, which may explain why the interdomain routing system is generally stable in practice. However, these conditions may be too restrictive for ISPs to offer customized route selection. In particular, ISPs may want to violate the preference condition to (1) have different preferences for different neighbors and (2) perhaps even prefer peer or provider routes for some (high-paying) customers. Therefore, we ask the following natural questions: “Would violating the preference condition lead to routing instability in NS-BGP?” and “What sufficient conditions (the equivalent of the Gao-Rexford conditions) are appropriate for NS-BGP?” Answering these questions is crucial to know if customized route selection is possible without sacrificing global stability, and without imposing onerous restrictions on how ASes exploit the extra flexibility.

1.3 Relaxing the “Prefer Customer” Condition

In this paper, we prove that the more flexible NS-BGP requires significantly less restrictive conditions to guarantee routing stability. Specifically, the “prefer customer” preference condition is no longer needed. Instead, an AS can freely choose any “exportable” path (i.e., a path consistent with the export condition) for each neighbor without compromising global stability. That is, an AS can select any route for a customer, and any customer-learned route for a peer or provider. This condition provides new understanding of the long-believed fundamental trade-off between “local flexibility” and “global stability” in interdomain routing. We make three main contributions in this paper:

- An NS-BGP model that captures neighbor-specific route selection and also simplifies the modeling of export policies. (Section 2)

- A proof of a sufficient condition for NS-BGP stability that relies only on the export and topology conditions. (Section 3)

- Observations that (1) the above NS-BGP stability conditions are robust to failures and other topology changes, (2) NS-BGP supports backup routing without oscillation and routing anomalies such as “BGP wedgies” (which can occur in BGP), and (3) the global routing system will remain stable when some ASes run NS-BGP while others still run conventional BGP (and thus NS-BGP can be safely deployed incrementally). (Section 4)

We also discuss the practical issues associated with deploying NS-BGP in Section 5, including dissemination of alternative routes within an AS, using tunneling to ensure incoming packets (from a neighboring AS or the ISP’s own local hosts) traverse the chosen paths, and different models of providing customized route selection.

In addition to studying stability issues about NS-BGP, we were also curious about the implications of neighbor-specific route selection on recent theoretical results about the incentive compatibility of BGP [14, 15]. We show in Section 6 that, as in conventional BGP, rational ASes have an incentive to lie about the paths they are using in NS-BGP. Yet, we argue that this does not affect our positive results regarding NS-BGP stability. Section 7 presents related work, and Section 8 concludes the paper.

2. NEIGHBOR-SPECIFIC BGP (NS-BGP)

In this section, we formally present Neighbor-Specific BGP (NS-BGP). NS-BGP inherits everything from conventional BGP (from the message format to the way messages are disseminated between ASes) except for the way it selects routes. We first present a formal model of its neighbor-specific route selection process, and then define the notion of stable path assignment for NS-BGP in preparation for the analysis of NS-BGP stability properties in Section 3. Finally, we highlight the key novel features of the NS-BGP by contrasting it with conventional BGP.

2.1 Preliminaries

In our NS-BGP model, the topology of an interdomain routing system is described as an AS graph $G = (V, E)$, where the set of vertices (nodes) $V$ represents the ASes, and the set of edges $E$ represents links between ASes. $V$ consists of $n$ source nodes $\{1, \ldots, n\}$ and a special destination node $d$ to which all other (source) nodes attempt to establish a path. (This formulation makes sense as routes to different destination ASes/prefixes are computed independently.) $E$ consists of directed edges. That is, if nodes $u$ and $v$ have a bi-directional
link between them, we have \{u, v\} ∈ E and \{v, u\} ∈ E, where \{u, v\} is the directed edge from u to v, and \{v, u\} is the directed edge from v to u.

Similar to [9], we define a path \( P \) in G as either the empty path, denoted by \( \epsilon \), or a sequence of nodes \( (v_k, v_{k-1}, \ldots, v_0) \), \( k \geq 0 \), such that for each \( i \), \( k \geq i > 0 \), \( \{v_i, v_{i-1}\} \in E \). Each non-empty path \( P = (v_k, v_{k-1}, \ldots, v_0) \) has a direction from its first node \( v_0 \) to its last node \( v_k \). For each \( v \in V \), \( \mathcal{P}^v \) denotes the set of all simple paths (i.e., paths that do not contain repeated nodes) that has \( v \) as the first node and \( d \) as the last node, plus the empty path \( \epsilon \). If \( P = (v_k, v_{k-1}, \ldots, v_0) \) is in \( \mathcal{P}^v \), then the node \( v_k \) is called the next hop of \( v \) in path \( P \). For each \( \{u, v\} \in E, \mathcal{P}^{(u,v)} \) denotes the set of all simple paths that have \{u, v\} as the first edge (i.e., \( u \) as the first node, \( v \) as \( u \)'s next hop) and \( d \) as the last node, plus the empty path \( \epsilon \). It is easy to see that, for any non-empty path \( P \in \mathcal{P}^v \), there is a corresponding path \( P' \in \mathcal{P}^{(u,v)} \) such that \( P' = (u \, v) \, P \). Here we use \( u \, v \, P \) to denote the operation of adding a new first edge \{u, v\} to the path that starts at node \( u \), so that the new path \( P' \) starts at node \( u \), traverses the edge \{u, v\}, and then follows path \( P \) from \( v \) to \( d \). Collectively, we use \( \mathcal{P}^{(u,v)} \) to denote the set of \( P' = (u \, v) \, P \) for all \( P \in \mathcal{P}^v \) and \( \{u, v\} \in E \), plus the empty path \( \epsilon \).

2.2 Neighbor-Specific Route Selection Model

As mentioned in Section 1, BGP uses a “one-route-fits-all” route selection model that requires a router to select a single best route for all neighbors. In NS-BGP, we enable customized route selection by allowing a router to select routes on a per neighbor (or equivalently) per edge-link basis. For simplicity, we use “nodes” to denote ASes (instead of routers) in the following model. We discuss the practical issues on how to realize this AS-level route selection model in Section 5.

Edge-based ranking functions: In the NS-BGP route selection model, for each edge \{u, v\}, there is a ranking function \( \lambda^v_u \), defined over \( \mathcal{P}^{(u,v)} \), which represents how node \( v \) ranks all possible paths for edge \{u, v\} (or equivalently, for neighbor \( u \)) to reach \( v \). If \( P_1, P_2 \in \mathcal{P}^{(u,v)} \) and \( \lambda^v_u(P_1) < \lambda^v_u(P_2) \), then \( P_2 \) is said to be preferred over \( P_1 \). We require \( \lambda^v_u \) to impose a strict order (with no ties) over all paths in \( \mathcal{P}^{(u,v)} \), as \( v \) must select a single best path for \( u \).

In NS-BGP, each source node \( v \in V \) repeatedly solves the following route selection problem, whenever it receives an update of the set of available paths to destination node \( d \):

**Definition 1 (Route selection problem).** Given a set of available paths \( \mathcal{P}^v_u \subseteq \mathcal{P}^v \) to destination \( d \), choose a best path from \( \mathcal{P}^{(u,v)} \) for each edge \{u, v\} according to the ranking function \( \lambda^v_u \).

As the name “Neighbor-Specific BGP” suggests, different edges \{u, v\} and \{v, u\} that point to \( v \) from different neighbors \( u \) and \( w \) can have different ranking functions \( \lambda^v_u \) and \( \lambda^w_v \), respectively. For example, in Figure 1(a), node 1 has two different ranking functions for the two edges \{2, 1\} and \{3, 1\} (or equivalently, for its two neighbors 2 and 3): \( \lambda^3_2 = ((2 \, 1 \, d) > (2 \, 1 \, 3 \, d) > \epsilon) \) (from the most preferred path to the least preferred path), and \( \lambda^3_3 = ((3 \, 1 \, d) > (3 \, 1 \, 2 \, d) > \epsilon) \). Node 2 and 3 are similar.

**Policy abstraction:** Since the empty path \( \epsilon \in \mathcal{P}^{(v,u)} \), the ranking function \( \lambda^v_u \) can also model \( v \)'s export policy for \( u \) (in addition to modeling \( v \)'s route selection policy for \( u \)). This is because if \( v \)'s export policy does not allow announcing a path \( P \) to \( u \), it is equivalent to make \( P \) less preferred than the empty path \( \epsilon \) in the ranking function, i.e., \( \lambda^v_u(P) < \lambda^v_u(\epsilon) \).

For instance, in Figure 1(a), if \( d \) is node 1’s customer whereas both nodes 2 and 3 are node 1’s peers or providers, node 1 could rank the empty path \( \epsilon \) higher than all the paths learned from node 3 in \( \lambda^2_3 \) to enforce the “no transit service for peer or provider” export policy, e.g., \( \lambda^2_3 = ((2 \, 1 \, d) > \epsilon > (2 \, 1 \, 3 \, d)) \).

2.3 Stable Path Assignment

Section 2.2 defines the route selection model every individual node uses in NS-BGP. We now define the collective outcome of the route selection processes run by the individual nodes — the path assignment.

**Definition 2 (Path assignment).** An NS-BGP path assignment is a function \( \pi \) that maps each edge \{u, v\} ∈ \( E \) to a path \( \pi(\{u, v\}) \in \mathcal{P}^{(u,v)} \). \( \pi(\{u, v\}) = \epsilon \) means that \( \{u, v\} \) is not assigned a path to \( d \).

Only a subset of all the possible path assignments can actually exist, because an edge can only be assigned a path that “overlaps” with a path assigned to one if its next-hop edges. In other words, path assignment for different edges must be consistent.

**Definition 3 (Consistent path assignment).** A consistent path assignment is a path assignment for which the following statement is true: For each \( \{u, v\} \in E \), if \( \pi(\{u, v\}) \) has \{v, w\} as its second edge (right after \( \{u, v\} \)), then \( \pi(\{u, v\}) = (u \, v \, \pi(\{v, w\})) \).

**Definition 4 (Stable path assignment).** A path assignment \( \pi \) is stable at edge \( \{u, v\} \) if the following two statements are true: (1) \( \pi \) is a consistent path assignment, (2) For every edge \( \{v, w\} \in E \), if \( \pi(\{u, v\}) \neq (u \, v \, \pi(\{v, w\})) \), then \( \lambda^v_u(\{u, v\}) \pi(\{v, w\}) < \lambda^v_u(\{u, v\}) \).

For example, in Figure 1(a), a stable path assignment is \((1 \, d), (2 \, d), (3 \, d), (1 \, 2 \, d), (1 \, 3 \, d), (2 \, 1 \, d), (2 \, 3 \, d), (3 \, 1 \, d), (3 \, 2 \, d)\).

2.4 BGP vs. NS-BGP

Our NS-BGP model differs from the conventional BGP model [9] in the following three respects.

**Ranking function(s): node-based vs. edge-based:** The conventional BGP model requires \( v \) to use a single ranking function \( \lambda^v \) for all neighbors, as shown in 1(b), offering little flexibility for node \( v \) to select the path that best meets an individual neighbor’s need. In contrast, the NS-BGP model allows...
each edge \(\{u, v\}\) to have a separate ranking function \(\lambda_u^v\), which allows \(v\) to provide customized route selection for individual neighbors, as shown in Figure 1(a).

Export policy modeling: separate vs. incorporated: Although conventional BGP supports per neighbor-based export policies, it uses a single ranking function \(\lambda\) to select routes for all neighbors. As a result, export policies must be modeled separately from the route selection process. Such separation is no longer necessary in the NS-BGP model, as node \(v\)'s export policy for neighbor \(u\) can be conveniently incorporated in the ranking function \(\lambda_u^v\). For example, if \(u\) is \(v\)'s peer or provider, in the ranking function \(\lambda_u^v\), \(v\) can simply rank the empty path \(v\) higher than all peer- or provider-learned paths to implement the “no transit service for peer or provider” export filtering policy.

Path assignment: node-based vs. edge-based: In the conventional BGP model, every node \(v\) gets assigned one path \(\pi(u)\). As a result, all of \(u\)'s neighbors learn the same path from \(u\)\footnote{In practice, an AS usually consists of multiple routers, each of which may learn different paths. Thus, neighbors connect to the AS at different edge routers might learn different paths, due to “hot potato routing”. Nevertheless, NS-BGP provides a far more flexible and systematic way for ASes to provide customized route-selection service, independent of whether neighbors connect to the same edge router or not.}. Whereas in the NS-BGP model, every edge \(\{u, v\}\) is assigned a path \(\pi(\{u, v\})\). This allows every node \(u\) to simultaneously utilize up to \(k\) paths to forward traffic from its neighbors as well as its own traffic, where \(k\) is the number of nodes \(v \in V\) such that \(\{u, v\} \in E\).

3. SUFFICIENT CONDITIONS FOR NS-BGP STABILITY

The “Gao-Rexford” conditions [10] state that, if all ASes follow the export, preference, and topology conditions, today’s BGP routing system is guaranteed to converge to a stable state. Fortunately, we find that many less restrictive conditions are sufficient to guarantee convergence under the more flexible NS-BGP. Specifically, we show that the “prefer customer” condition is no longer needed in NS-BGP—individual ASes can freely choose any “exportable” routes without compromising global stability. In this section, we first define the notion of NS-BGP safety, which implies that an NS-BGP routing system always converges to a stable path assignment. We then introduce Iterated Dominance, the machinery we use in our proof. We next present simple examples that illustrate why NS-BGP requires less restrictive conditions for safety than conventional BGP, before presenting the proof of our NS-BGP safety result.

3.1 Formal Definition of NS-BGP Safety

For any policy-based (non-shortest-path) routing protocol (such as BGP or NS-BGP), safety is a top concern, as persistent route oscillations can significantly impact end-to-end performance, and even threaten the reachability of network destinations. BGP safety can be loosely defined as a routing system that always converges to a “stable” state. Recall that a stable state is a path assignment that is not changed by any possible routing operations. Thus, once a system is in a stable state, it will never experience any further changes (provided the network topology and every node’s routing policy remain the same). To formally define NS-BGP safety, we first need to introduce the notion of “AS activation sequences”.

AS activation sequences: As in conventional BGP, the routing outcome of NS-BGP is built, hop-by-hop, as knowledge about how to reach a destination \(d\) propagates throughout the network. The process begins when \(d\) announces itself to its neighbors by sending update messages. From this moment forward, every node \(v\) repeatedly picks a path for each edge \(\{u, v\} \in E\), based on the most recent updates of routes to \(d\) it received from its neighbors. As in [8,9], the network is assumed to be asynchronous. That is, edges can be activated (i.e., get assigned new paths) at different times, and update messages can be delayed or even lost (as long as they are retransmitted eventually). We refer readers to [9] for a thorough explanation of this asynchronous environment.

Definition 5. An NS-BGP routing system is safe if it always converges to a stable path assignment from any initial path assignment, and for any AS activation sequence.

3.2 Iterated Dominance

All known conditions that guarantee the safety of conventional BGP (e.g., “No Dispute Wheel” [9] and the “Gao-Rexford” conditions [10]) share a common structure [16]. Here, we refer to this property, which is closely related to fundamental concepts in game theory [17], as “Iterated Dominance”. Iterated Dominance is an underlying structure of a routing instance, which will enable us to show that, for any activation sequence, NS-BGP is bound to converge to a unique stable state. Thus, Iterated Dominance provides us with the means to present a constructive, and general, proof for NS-BGP safety. We shall later show that the commercial setting considered in this paper is simply a special case of Iterated Dominance. To define Iterated Dominance, we first require the following definitions:

Definition 6 (Consistent paths I). We say two paths \(P_1\) and \(P_2\) are consistent if the following statement holds: For every edge \(\{i, j\}\) that is on both \(P_1\) and \(P_2\), the suffix of \(P_1\) that leads from \(j\) to \(d\) is identical to the suffix of \(P_2\) that leads from \(j\) to \(d\). Two paths that do not share any common edge are consistent.

Definition 7 (Consistent paths II). Let \(P = \{P_1, \ldots, P_k\}\) be a set of paths in \(G\). We say that a path \(Q\) in \(G\) is consistent with \(P\) if it is consistent with every path in \(P\).

Definition 8 (Feasible paths). Let \(P = \{P_1, \ldots, P_k\}\) be a set of paths in \(G\). We define the set of feasible paths \(Q\) given \(P\) be the set of all paths in \(G\) that are consistent with \(P\).

Definition 9 (Iterated Dominance). We say that Iterated Dominance holds if there exists an order over all edges in \(G\): \(e_1, \ldots, e_{|E|}\) (\(e_i \in E, 1 \leq i \leq |E|\)), for which the following three statements hold:

- There exists a set of paths \(P_{e_1}, \ldots, P_{e_{|E|}}\) such that for every \(1 \leq i \leq |E|\), \(P_{e_i}\) is a path to \(d\) that has \(e_i\) as the first edge.
- For every \(1 \leq i \leq |E|\), \(P_{e_i} = e_i P_{e_k}\) for some \(0 \leq k < i\). (We define \(e_0\) to be the empty path \(\epsilon\).)
- For every \(1 \leq i \leq |E|\), \(P_{e_i}\) is \(e_i\)'s most preferred path in the set of feasible paths given \(\{P_{e_1}, \ldots, P_{e_{|E|}}\}\).
Intuitively, this definition means that once the paths assigned to edges that come before a certain edge are fixed, that edge’s path is its most preferred feasible path. Iterated Dominance has the nice property that, if it exists in a routing system, it trivially and intuitively induces convergence to a stable path assignment.

**Proposition 3.1.** If Iterated Dominance holds for an inter-domain routing instance, then NS-BGP is safe for that routing instance. Moreover, NS-BGP always converges to a unique stable path assignment.

**Proof.** The proof immediately follows from the Iterated Dominance property. If Iterated Dominance holds then there must be an order over the edges \( e_1, \ldots, e_k \) such that, for every \( 1 \leq k \leq |E| \) an edge \( e_k \) can be assigned its most preferred feasible path (given that \( e_1, \ldots, e_{k-1} \) are assigned \( P_{e_1}, \ldots, P_{e_{k-1}} \)), regardless of what paths are assigned to \( e_{k+1}, \ldots, e_{|E|} \). Thus, we can simulate the execution of an activation sequence of NS-BGP that shows the routing system must converge to a unique stable path assignment:

At some point in time \( e_1 \) will learn of its most preferred path \( P_{e_1} \). From that moment forward, \( e_1 \) will stick to the path \( P_{e_1} \) (which, by the definition of Iterated Dominance, is always available to \( e_1 \)). Now, consider \( e_2 \). Once \( e_1 \)'s path is fixed, by the definition of Iterated Dominance, \( e_2 \) can get its most preferred feasible path \( P_{e_2} \). Therefore, from some moment in time onwards (when update message containing \( P_{e_2} \) reaches \( e_2 \)), \( e_2 \)'s path will be fixed and never change. By definition of Iterated Dominance, we can continue iteratively fixing other edges’ paths until every edge has a fixed path. Observe that the resulting path assignment is stable, because after each edge \( e_i \) gets its path \( P_{e_i} \), it will never switch to other paths.

### 3.3 Examples of Safe NS-BGP Systems

Before presenting the formal proof of our main result, we first use an example to illustrate why safety might be easier to achieve for NS-BGP than for conventional BGP. Figure 1(b) shows a routing system in which NS-BGP will always diverge, which is called BGP BAD GADGET. In this example, \( \lambda_1 \), \( \lambda_2 \) and \( \lambda_3 \) are the ranking functions of nodes 1, 2 and 3, respectively. It is easy to construct an action sequence (presented as a sequence of path assignments) according to the ranking functions that oscillates: \(((1 \ d), (2 \ d), (3 \ d)) \rightarrow ((1 \ 2 \ d), (2 \ d), (3 \ d)) \rightarrow ((1 \ 2 \ d), (2 \ 3 \ d), (3 \ d)) \rightarrow ((1 \ d), (2 \ d), (3 \ 1 \ d)) \rightarrow ((1 \ 2 \ d), (2 \ d), (3 \ 1 \ d)) \rightarrow ((1 \ d), (2 \ d), (3 \ d)) \rightarrow ((1 \ 2 \ d), (2 \ d), (3 \ d)).\) (An underlined path indicates that it has changed from the previous path assignment.) Notice that the third path assignment is the same as the last path assignment. Therefore, the system will continue to oscillate and never terminate.

To see how NS-BGP can help in cases like this, we transformed the BGP routing system in Figure 1(b) to an “equivalent” NS-BGP system in Figure 1(a). This is an “extreme” example in that we assume every node is willing to select paths for each incoming edge (i.e., each neighbor) completely according to the edge’s (or equivalently, the neighbor’s) ranking function. For example, when selecting best path for edge \( (2, 1) \), node 1 in Figure 1(a) uses a ranking function \( \lambda_2 \) that is essentially the same as node 2’s ranking function \( \lambda_2 \) in Figure 1(b). The only difference is that, since \( \lambda_2 \) is defined over \( P^{(2,1)} \), whereas \( \lambda_3 \) is defined over \( P^3 \), only a subset of the paths in \( P^3 \) that begin with edge \( (2, 1) \) (e.g., \( (2 \ 1 \ 3 \ d) \)) are included in \( \lambda_3 \). We omit the empty path \( e \) for simplicity. It is easy to see that the transformed BGP BAD GADGET in Figure 1(a) becomes an NS-BGP GOOD GADGET, i.e., a routing system in which NS-BGP will always converge to a unique stable path assignment. In this case, the unique stable path assignment for all edges is: \(((1 \ d), (2 \ d), (3 \ d), (1 \ 2 \ d), (1 \ 3 \ d), (2 \ 1 \ d), (2 \ 3 \ d), (3 \ 1 \ d), (3 \ 2 \ d))\).

This example illustrates why safety might be easier to obtain for NS-BGP than for conventional BGP. In practice, however, relying on such completely “selfless” routing policies is unreal- istic. This prompts us to investigate the safety conditions for NS-BGP in a more realistic commercial setting that accounts for the business relationships between ASes. For example, consider Figure 2, where node \( d \) is a customer of nodes 1, 2 and 3. Node 3 is a customer of nodes 1 and 2, and node 1 is a customer of node 2. It is easy to see there is no “customer-provider” cycle in the graph so the topology condition holds. We also require nodes 1, 2 and 3 to adhere to the export condition and export only customer routes to peers or providers. Now we compare BGP and NS-BGP and analyze why the “pre- customer” condition is necessary in conventional BGP but redundant in NS-BGP. First, note that the ranking function \( \lambda_3 \) prefers provider-learned route \( (3 \ 2 \ d) \) over customer-learned route \( (3 \ d) \), violating the preference condition for the regular BGP. As a result, (not surprisingly) the routing system is a BGP BAD GADGET.

A key observation about the instability of the BGP system in Figure 2 is that the availability of route \((1 \ 3 \ d)\) to node 1 is dependent upon the unavailability of route \((3 \ 2 \ d)\) to node 3— if route \((3 \ 2 \ d)\) is available to 3, it will choose route \((3 \ 2 \ d)\) over \((3 \ d)\), and announce no route to node 1; whereas if route \((3 \ 2 \ d)\) is not available to 3, it will choose route \((3 \ d)\) and announce it to node 1 (since \((3 \ d)\) is a customer-learned route). Things work differently in NS-BGP. NS-BGP ensures that a route announced to a peer or provider does not change based on the presence or absence of any non-exportable (i.e., peer- or provider-learned) routes. That is, in this example, node 3 learning \((3 \ 2 \ d)\) (a provider-learned route) should not affect whether node 3 exports \((3 \ d)\) to node 1 (which is also a provider). Fundamentally, this is because, in NS-BGP, node 3 can announce a different route \((3 \ d)\) to node 1 than the route it selects for its own traffic, namely \((3 \ 2 \ d)\).
3.4 Safety Conditions for NS-BGP

To prepare for our analysis, we first define some terminology: We say that an edge \( e = (u, v) \in E \) is a customer edge if \( v \) is \( u \)'s customer. Similarly, we say that an edge \( e = (u, v) \) is a peer edge or a provider edge if \( v \) is \( u \)'s peer or provider, respectively. Observe that the “No customer-provider cycle” topology condition in the “Gao-Rexford” guidelines can now be interpreted as stating that there must be no cycles in the graph containing only customer edges or only provider edges. Also observe that the “Export only customer routes to peer or graph containing only customer edges or only provider edges” condition means that if a path \( P \) contains a customer edge or a peer edge, then all edges that come after that edge (towards the destination) must also be customer edges, allowing us to simply disregard all other types of paths in our analysis.

**Lemma 3.2.** If the Topology and Export conditions hold for an NS-BGP routing instance, then Iterated Dominance holds for that routing instance.

**Proof.** We shall show that an order over edges \( e_1, \ldots, e_{|E|} \), as in the definition of Iterated Dominance, exists. Obviously, we can set \( e_1 \) to be any edge of the form \( (u, d) \) \((u, d) \in E\) as \( (u, d) \) is the only path that edge has to \( d \). So by setting \( P_1 = (u, d) \), we have found an edge \( e_1 \) that fits the definition of Iterated Dominance. The rest of the proof shows how to prove the existence of an edge \( e_2 \), as required by the definition of Iterated Dominance. The same method can then be applied recursively to find \( e_3, \ldots, e_{|E|} \) (thus concluding the proof).

If there is another edge of the form \( (u, d) \), we can now set \( e_2 \) to be that edge for the same reason as before. We shall now show how to find \( e_2 \) as in the definition of Iterated Dominance, if this is not the case. Informally, the proof shall now proceed by iteratively applying the following procedure: Fix an edge \( e \). Go over its most preferred feasible route (given \( P_1 \)) until reaching the edge before last, \( l_1 \). If edge \( l_1 \) fits the description of \( e_2 \) then we are done. Otherwise, we apply the same procedure to \( l_1 \), moving to the edge before last on \( l_1 \)'s most preferred feasible path, called \( l_2 \) (which we regard as a new candidate to be \( e_2 \)). Thus, we create a sequence of edges \( l_1, l_2, \ldots \). We show that this procedure eventually reaches an edge that fits the description of \( e_2 \) (thus concluding the proof), because otherwise the “No customer-provider cycle” will be violated (a contradiction).

Formally: Let \( e \neq e_1 \) be some arbitrarily chosen edge. Let \( P_e \) be \( e \)'s most preferred path among all feasible paths given \( P_{e_1} \). For ease of exposition, we first consider the case in which \( e \) is a customer edge.

Now, to find \( e_2 \), we shall construct a series of edges \( l_1, \ldots, l_k, \ldots \) in the following manner: Let \((i, j)\) be the two-edge suffix of \( P_e \) (i.e., the last two edges on \( P_e \) are \((i, j)\) followed by \((j, d)\)). We set \( l_1 \) to be \((i, j)\). If \( l_1 \) prefers \((i, j)\) over all other feasible paths, then we can set \( e_2 \) to be \( l_1 \) and \( P_{e_2} \) to be \((i, j)\) (and are done). If, however, \( l_1 \)'s most preferred feasible path \( P_{l_1} \) is not \((i, j)\), we then consider the two-edge suffix of \( P_{l_1} \). We set \( l_2 \) to be the first of these two edges. For \( l_2 \), we repeat the same process we went through for \( l_1 \). That is, either \( l_2 \) prefers the two-edge suffix of \( l_1 \) over any other feasible path (in which case we set \( e_2 \) to be \( l_1 \) and are done), or we move on to \( l_3 \) (which is the first edge of \( l_2 \)'s most preferred path’s two-edge suffix). We continue this process, constructing a series of edges \( l_1, \ldots, l_k, \ldots \). If this process terminates then we must have reached an edge that fits the description of \( e_2 \).

We prove that this process must terminate by showing that if it does not terminate, we will reach a contradiction to the topology condition (“No customer-provider cycles”).

First, observe that for any edge \( l_1 \) in the series of edges \( l_1, \ldots, l_k, \ldots \), there exists a path between \( l_1 \) and \( l_{k+1} \) that consists only of customer edges. To see why this is true, consider \( l_1 \). We assumed that \( e \) was a customer edge. Therefore, by the export condition, any path assigned to \( e \) must only consist of customer edges. Since \( l_1 \) is on such a path, it must be a customer edge. Using the same argument, we know that \( l_1 \) can only be assigned paths consisting of only customer edges. Since \( l_2 \) is, by definition, on such a path (\( l_1 \)'s most preferred feasible path), we have shown that the path between \( l_1 \) and \( l_2 \) consists of customer edges only, so the claim holds for \( l_1 \). We can now repeat the same argument for \( l_2, l_3, \ldots \).

Now, if the process does not terminate, then, since the number of edges is finite, some edge \( l_i \) will eventually appear twice in the sequence \( l_1, \ldots, l_k, \ldots \). Consider the subsequence of \( l_i, \ldots, l_i \) (between \( l_i \)'s first and second appearance). Because any two consecutive edges in this cyclic sequence have a path between them that consists of only customer-edges, there must exist a customer-provider cycle (i.e., a cycle of only customer edges).

If the \( e \) is a peer edge or a provider edge and \( l_i \) is a peer edge then the edge that comes after it must be a customer edge, so the same arguments as before apply. If \( e \) is a provider edge then the process described before will either go through a customer edge or a peer edge (in which case, once again, the arguments above apply) or lead to a cycle of provider edges. ☐

We are now ready to prove the safety conditions of NS-BGP:

**Theorem 3.3 (Safety Conditions of NS-BGP).** If the Topology and Export conditions hold then NS-BGP is safe. Moreover, NS-BGP always converges to a unique stable path assignment.

**Proof.** Lemma 3.2 shows that the topology and export conditions are sufficient to guarantee Iterated Dominance. Therefore, by Proposition 3.1, NS-BGP is safe, and always converges to a unique stable path assignment. ☐

3.5 Tightness of the Safety Conditions

In this subsection we show that our NS-BGP safety conditions are “tight”, in the sense that a relaxation of either the topology condition or the export condition might result in persistent NS-BGP oscillations.

Consider the example depicted in Figure 3. This example can be viewed as an adaptation of the well-known BGP BAD GADGET instance described in [9] and Figure 1(b) to the neighbor-specific BGP setting. The top two preferred paths in edges \{1,2\}'s, \{3,4\}'s, and \{5,6\}'s ranking functions are listed (from top to bottom) in the figure. We omit the rest of the paths in the ranking functions for simplicity, as they play no roles in this example. The business relationships between the ASes are described in the figure (where the arrows point from customers to their providers). Observe that the topology condition holds as there are no customer-provider cycles. If we assume that the export constraint also holds then, by Theorem 3.3, this NS-BGP routing system is guaranteed to converge to a unique stable path assignment.

What happens if the export condition is removed (i.e., not followed)? We claim that the system will then have no stable path assignment and so will oscillate indefinitely. Observe that
if node 2 follows the export condition, it cannot export path (2 3 4 5 6) to node 1, making path (1 2 3 4 5 6) unavailable to node 1. Similarly, paths (3 4 5 6 7) and (5 6 1 2 7) are not available to nodes 3 and 5, respectively. But if the export condition is not followed, these paths will become available. Assume, to lead to a contradiction, that a stable path assignment exists when the export condition is removed. Observe that edge (1, 2) must either get the path (1 2 3 4 5 6) or (1 2 3 4) in this path assignment (as it will not settle for a less preferred path than its second preferred path (1 2 d) that is always available). Let us first consider the possibility that (1, 2)’s path in this stable assignment is (1 2 d). If that is the case, then (5, 6) must be getting the path (5 6 1 2 d). This means that node 5 will not announce (5 6 1 2 d) to node 4 (because node 6 announces (6 1 2 d), rather than (6 d), to node 5). Therefore, edge (3, 4) is assigned the path (3 4 d), which, in turn, means that edge (1, 2) can get its preferred path (1 2 3 4 d). Now we have contradiction—edge (1, 2) has an available path (1 2 3 4 d) which it prefers over the path it is assigned in the stable path assignment (1 2 d). Observe that if, instead, we assume that edge (1, 2) gets path (1 2 3 4 d) in the stable path assignment, then edge (3, 4) must get path (3 4 d) in the stable path assignment. We can continue this inference process like above and eventually reach a similar contradiction to edge (1, 2)’s assigned path.

We have shown that without the export condition, not only is NS-BGP safety not guaranteed but there might not even be a stable path assignment to which it can converge. We make the observation that this is also the case if we remove the topology condition (while leaving the export condition alone). Consider the same example, only with the following business relationship changes: make nodes 3, 5, and 1 customers of nodes 2, 4 and 6, respectively. Observe that the topology condition no longer holds as we now have a customer-provider cycle (3 → 2 → 1 → 6 → 5 → 4 → 3). Also observe that paths (1 2 3 4 d), (3 4 5 6 d), and (5 6 1 2 d) are now allowed by the export condition as a result of the changes in the business relationships we made. Therefore, we can use the same analysis as above to show that no stable path assignment exists if the topology condition is removed.

4. PRACTICAL IMPLICATIONS

In this section we discuss three practical implications of the NS-BGP safety conditions presented in Section 3. Specifically, we show that:

1. Our NS-BGP safety conditions are robust, in the sense that they hold even in the presence of topology changes (e.g., the addition and removal of nodes and/or links due to new business contracts, creation, merger, or disappearance of ASes, network failures, etc.).

2. By allowing arbitrary ranking of exportable paths, NS-BGP naturally supports an important class of “backup” business relationships (i.e., an AS having a backup provider) and prevents “BGP Wedgies” [18] that can occur in conventional BGP.

3. It is safe to deploy NS-BGP incrementally. Global routing stability is guaranteed even if only some of the ASes run NS-BGP, while others continue to run BGP. Moreover, the global routing system is still guaranteed to converge to a unique stable path assignment.

4.1 Safe Under Topology Changes

We have shown in Section 3.4 (Theorem 3.3) that if the topology and export conditions hold for a routing instance, then NS-BGP is guaranteed to converge to a stable path assignment. However, does this result still hold in the presence of topology changes (e.g., the addition and removal of nodes and/or links due to new business contracts, creation, merger or disappearance of ASes, network failures, etc.)? We make the observation that our NS-BGP safety conditions are robust in the presence of topology changes.

Safety in the presence of link/node removals: We first consider topology changes that result in the removals of edges and/or vertices from the graph G in our model. Such changes can happen due to network failures (e.g., equipment malfunctions, fiber cuts) or business relationship changes (e.g., termination of a existing BGP peering relationship). To see why NS-BGP safety withstands such topology changes, simply observe that, if the topology condition and the export condition hold for some routing instance, they cannot be violated by removing edges and/or vertices from the network. Hence, after the removal of certain edges and/or vertices, we will end up with a new routing instance for which these two conditions still hold. By Theorem 3.3, NS-BGP safety of the new routing instance is guaranteed.

Safety in the presence of link/node additions: Now we consider topology changes that result in the additions of edges and/or vertices from the graph G in our model. Such changes can happen due to business relationship changes (e.g., establishment of a new BGP peering relationship or a new AS). We note that our proof of Theorem 3.3 still holds for the new routing instance after such topology changes, as long as they do not violate the topology and export conditions. That is, the new vertices and/or edges do not create “customer-provider” cycles and they follow the “export only customer routes to peer or provider” export policy. Since ASes have economic incentive to follow the two conditions, the new routing instance is guaranteed to remain safe.

4.2 Safe With Backup Relationship

As we know, if all ASes follow the “Gao-Rexford” conditions, a BGP routing system is guaranteed to be stable. However, the “Gao-Rexford” conditions only apply to routing systems with the two most common business relationships (“customer-provider” and “peer-peer”). Yet, it has been increasingly com-
BGP Wedgies: The term “BGP Wedgies”, coined in [18], refers to the following problem with BGP: It is common for an AS to have two (or more) upstream providers to avoid a single point of failure in network connectivity. In such cases, the AS usually places a relative preference on the two links its providers use to reach it: one link is defined as the “primary” (preferred), while the other one is defined as the “backup” link. A backup link is intended to be used only when the primary link is temporarily unavailable, therefore is typically much less well-provisioned in terms of bandwidth. It is expected that once the primary link is restored, all traffic should switch back from the backup link to the primary link. BGP Wedgies are anomalous situations in which, even after a failed primary link is restored, the BGP state of the routing system does not “flip back” to the intended state that existed before the link failure.

Consider the example of a Wedgie in conventional BGP, as shown in Figure 4. AS 1 is a customer of ASes 2 and 4, AS 2 is a customer of AS 3, and ASes 3 and 4 are peers. AS 1 chooses to use the link \{1, 4\} as the primary link and the link \{1, 2\} as the backup link. AS 1 instructs AS 2 to use path (2 1) only when there is no other path available (e.g., using the BGP community attribute to mark the path (2 1) as “backup only” in its route updates). Assume that the original BGP state is such that all ASes are forwarding their traffic to AS 1 along the path (2 3 4 1). Observe that this state is stable (as AS 2 does not announce path (2 1) to AS 3 when path (2 3 4 1) is available). Now, assume that link \{4, 1\} goes down for some reason. Since the path (2 3 4 1) is no longer available, AS 2 will announce path (2 1) to AS 3, which will in turn announce it to AS 4. In the end, traffic to AS 1 is forwarded along the path (4 3 2 1). Once link \{1, 4\} is restored, a BGP Wedgie occurs: although AS 4 will announce path (4 1) is available again, AS 3 will not switch back from its current customer-learned path (3 2 1) to a less preferred peer-learned path (3 4 1), and will not announce the path (3 4 1) to AS 2. As a result, AS 2 (and 3) will keep using the backup link even though the primary link becomes available again.

NS-BGP prevents Wedgies: We make the observation that BGP Wedgies cannot occur in an NS-BGP system with backup relationships. To illustrate this point, let us first revisit the example described above. Notice that the Wedgie example in Figure 4 will not occur if the routing system runs NS-BGP, because AS 3 will have AS 2’s ranking function (in this case, \(\lambda_2 = (2 3 4 1) > \epsilon\)), and selects a path for AS 2 on its behalf. So when link \{1, 4\} is restored, AS 3 will learn the path (4 1) from AS 4 again and announce the path (3 4 1) to AS 2 because (2 3 4 1) is 2’s most preferred path. Once AS 2 learns this path, it will withdraw the backup path (2 1) from AS 3 and AS 3 will switch back to use (3 4 1). Therefore, the system will be restored to the original state that existed before the link failure.

It is not coincidental that, in the above example, a Wedgie occurs when the routing system runs BGP, but no Wedgie occurs when the same system runs NS-BGP. Observe that in both examples, the backup relationship between ASes 1 and 2 requires AS 2 to prefer path (2 3 4 1) (a provider-learned route) to path (2 1) (a customer-learned route), which violates the preference condition. In the conventional BGP setting, this violation of the “Gao-Rexford” conditions results in the two stable states and the Wedgie. However, for NS-BGP, the preference condition is no longer needed—Theorem 3.3 states that as long as the topology and export conditions hold, NS-BGP will always converge to a unique stable path assignment. This implies that Wedgies (that can only occur in routing systems with at least two stable path assignments) cannot occur.

4.3 Safe in Partial Deployment

The proof of the NS-BGP safety conditions in Section 3 assumes all ASes in the network run NS-BGP, i.e., a full deployment of NS-BGP. However, the actual deployment of NS-BGP will certainly start incrementally, as any AS that has deployed NS-BGP individually can immediately start offering customized routing-selection services without collaboration. Therefore, a natural question is whether the NS-BGP safety conditions still hold in a partial deployment scenario (with some “early adopter” ASes running NS-BGP, while other ASes still running conventional BGP)?

As we shall now show, the answer to this question is YES. That is, NS-BGP can be (under reasonable and realistic assumptions) incrementally- and partially-deployed without causing routing anomalies (such as persistent protocol oscillations and Wedgies). We observe that, using the exact same techniques we have used to prove Theorem 3.3, we can actually prove a much more general result: Assume the topology and export conditions hold for a routing system, then, even if some ASes are running NS-BGP while other ASes are still running BGP, as long as the preference condition applies to the ASes running conventional BGP (it is not needed for ASes running NS-BGP), the routing system will always converge to a unique stable path assignment. That is, as long as the ASes not running NS-BGP prefer customer routes to other routes in their route selection, the system will remain safe. We note that this result holds true regardless of the number of ASes that are not running NS-BGP, and regardless of the locations of these ASes in the network. This result therefore generalizes both Theorem 3.3 (that considers the case in which all ASes are running NS-BGP) and the “Gao-Rexford” conditions [10] (that consid-

\[\text{We omit the details of the proof as it follows similar lines of the proof in Section 3.4.}\]
ers the case in which all ASes are executing BGP).

We also observe that, by the same arguments as in Section 4.1 and 4.2, the above safety conditions of a partial NS-BGP deployment still hold in the presence of network topology changes, and a routing system with even partially deployed NS-BGP may experience less BGP Wedgies.

5. DEPLOYMENT ISSUES

In this section, we discuss the implementation issues in deploying NS-BGP in practice. First, we describe how an AS can correctly forward traffic from different neighbors (and from within its own network) along different paths. We then discuss how to disseminate multiple routes to the edge routers of an AS to enable flexible route selection. Finally, we present three models an NS-BGP-enabled AS can use to provide different levels of customized route-selection services. When deploying NS-BGP, an AS can handle all these issues by itself without requiring any changes from neighboring ASes, as no BGP message format or external BGP (eBGP) configuration are needed.

5.1 Neighbor-Specific Forwarding

NS-BGP requires routers to be able to forward traffic from different neighbors along different paths. Fortunately, today’s routers already provide such capabilities. For example, the “virtual routing and forwarding (VRF)” feature commonly used for Multi-protocol Label Switching Virtual Private Networks (MPLS-VPNs) supports the installation of different forwarding-table entries for different neighbors [19]. Since an AS typically consists of a network of many routers, traffic entering from various “ingress points” of the AS must be forwarded to the correct “egress point(s)”. In conventional BGP, this is achieved in a hop-by-hop fashion, as BGP ensures that all the routers in the AS agree to forward traffic to the closest egress point that has one of (potentially) multiple “equally good” best paths to the destination. For example, in Figure 5, if R5 learns one path from R3 and another path from R4 to D, and the two routes are considered “equally good” in BGP’s route-selection process, it will choose to use the closest egress point (according to Internal Gateway Protocol, or IGP distance). However, this approach no longer works in NS-BGP, as traffic entering the AS at the same ingress point may be from different neighbors (ingress links), thus may need to be forwarded to different egress points, or different egress links of a same egress point. Fortunately, ASes have an efficient solution available—encapsulation (or tunneling). Many commercial routers deployed in today’s networks can perform MPLS or IP-in-IP encapsulation/decapsulation at line rate. To provide customized forwarding for neighbors connected at the same edge router, the tunnels need to be configured from ingress links (rather than ingress routers) to egress links (rather than egress routers). For example, in Figure 5, C1 and C2 (that connect to the same ingress point R1) can tunnel their respective traffic to R6 and R7 (that connect to the same egress point R3) independently. To avoid routers in neighboring domains having to decapsulate packets, egress routers need to remove the encapsulation header before sending the packets to the next-hop router, using technique similar to the penultimate hop popping [20]. Similar to transit traffic originated from other ASes, traffic originated within the AS itself can also be forwarded to the correct egress links using tunneling.

5.2 Route Dissemination Within an AS

A prerequisite for an edge router to provide meaningful “cus-
preference on routes into account when determining the ranking function $\lambda_i^j$. Here we describe three different models that allow ASes to (1) provide different levels of customized route selection to different types neighbors, and (2) make trade-offs between the amount of flexibility offered to their neighbors and the corresponding resource requirement for their network infrastructure.

1. **The “subscription” model**: In this model, $i$ offers several different services (ranking functions) for its neighbors to choose from, such as “shortest path”, “most secure”, and “least expensive”. A neighbor $j$ has the flexibility to decide which one to “subscribe” to. Although more flexible than the standard BGP, this is still a fairly restrictive model, as $j$ can only choose a ranking function from the set specified and provided by $i$, without any direct influence how they are determined.

2. **The “total-control” model**: In this model, $i$ gives neighbor $j$ direct and complete control in determining the ranking function $\lambda_i^j$. Therefore, $j$ is guaranteed to receive its most preferred routes among all of $i$’s available routes.

3. **The “hybrid” model**: In this model, neighbor $j$ is allowed to specify certain preference to $i$ directly (e.g., avoid paths containing an untrusted AS whenever possible). When determining the ranking function $\lambda_i^j$ for $j$, $i$ takes both $j$’s preference and its own preference into account (as the “best route” according to $j$’s preference may not be the best for $i$’s economic interest). Nevertheless, $i$ still controls how much influence (“weight”) $j$’s preference has on the final ranking function $\lambda_i^j$.

An AS can choose to implement one or more of these models simultaneously to provide a range of customized route-selection services. The choice of models requires an AS to make trade-offs between flexibility and scalability within its network [26]. For example, the “subscription” model provides the most limited flexibility among the three models, at the same time it also imposes the least implementation overhead in terms of forwarding table size, route dissemination and customized route selection (e.g., each edge router or routing control platform server only needs to run a small number of route selection processes). On the other hand, the “total-control” model, while providing the finest grain of customization, imposes the most demanding requirements on system resources and performance. Therefore, we expect an AS to only provide such service to a small number of neighbors for a relatively high price. In addition, the “hybrid” and “total-control” models could be realized in two different ways. The simpler way is that the neighbor $j$ tells the AS $i$ what $\lambda_i^j$ to use, so $i$ can select and export a single route. The other way is that $i$ announces all exportable routes to $j$, and $j$ selects amongst them itself. This approach allows the $j$ to hide its policy (ranking function) but requires $i$’s ability to export multiple routes to $j$, and $j$’s ability to directly tunnel its traffic to $i$’s egress links. Finally, it is worth pointing out that the NS-BGP safety conditions (Theorem 3.3) hold regardless of which one(s) of these three models are used.

6. **NS-BGP AND INCENTIVES**

Most studies on BGP make the implicit assumption that ASes will obediently adhere to the protocol. Recently, there has been a surge of interest in BGP’s incentive-compatibility properties [14–16, 27, 28], which is motivated by the fact that ASes are independent entities with different, sometimes competing, economic interests. Hence, the assumption that ASes will always adhere to the protocol, even if sometimes they are better off not doing so (given their individual interests), might not be realistic. Studies of BGP’s incentive-compatibility take a game-theoretic (or microeconomic) approach to answer questions like “In what incentive models will ASes be rationally motivated to adhere to BGP (and not deviate)!”? As argued in [15], ensuring BGP’s incentive-compatibility is an important practical perspective, as it provides a way to avoid anomalies in which an AS intentionally announces one AS path and forwards data packets on another.

Given NS-BGP provides a new interdomain route selection model, we are curious about its incentive-compatibility properties, and how these properties compare to BGP’s. In this section, we examine NS-BGP from a game-theoretic perspective, and explore the possibility of making it incentive compatible. Unfortunately, we find that, as in conventional BGP, rational ASes have an incentive to lie about the paths they are using in NS-BGP. Therefore, unlike the positive routing stability results presented earlier in this paper, the transition from BGP to NS-BGP does not improve the incentive-compatibility properties of a routing system. However, we argue that NS-BGP (and BGP) will remain stable even in the presence of protocol manipulation.

6.1 **Background: BGP is Not Incentive-Compatible**

In the game-theoretic framework [14], ASes are assumed to be rational, in the sense that they are interested in maximizing (some notion of) their individual gains. Each AS in this model has a utility function, representing its objective, which it tries to maximize. Informally, saying “BGP is incentive compatible in an incentive model” would mean that if all other ASes are following the rules of BGP, then the best course of action for an AS is to do the same (i.e., it has no incentive to not do so in the incentive model). We refer readers to [14, 15] for an explanation of this framework.

Unfortunately, as observed in [14], conventional BGP is not necessarily incentive compatible even in small networks. This problem is further aggravated in realistic commercial settings in which ASes might be interested in attracting traffic from customers [15] to make more profit. Here we first illustrate the incentive-related problems with BGP using a simple example. This example also helps us in our later analysis of the incentive-compatibility properties of NS-BGP.

Consider the simple example illustrated in Figure 6, in which

![Figure 6: A system that is not incentive compatible in both BGP and NS-BGP](image-url)
all three “Gao-Rexford” safety conditions hold. Assume that for AS 3, its main interest is attracting AS 1’s traffic (i.e., making AS 1 forward traffic directly to AS 3), which is more important than attracting 2’s traffic, which, in turn, is more important than the path it uses to send its outgoing traffic to d. Further, assume that 3 is bound by business contracts to provide connectivity to its customers, and thus must always announce some path to ASes 1 and 2. Observe that if AS 3 announces the path (3 4 d) (its most preferred route for outgoing traffic) to AS 2, AS 2 will choose path (2 3 4 d) and let AS 1 get its most preferred path (1 2 3 4 d). However, if AS 3, even though still only uses path (3 4 d) to forward all the traffic to d, announces the path (3 5 d) to AS 2 (but still announces path (3 4 d) to AS 1), AS 2 will choose the path (2 d) and announce it to AS 1. This way, AS 1 will choose path (1 3 4 d) and forward its traffic directly through AS 3. This example shows that AS 3 can improve its gain by announcing a path that it is not actually using to one of its customers. This inconsistency between the path AS 3 announces AS 2 and actual path it uses to forward 2’s traffic is clearly an anomaly that should not happen (and is not expected by AS 2).

One might hope that the use of cryptographic mechanisms might help overcome such problems. Indeed, it has been shown in [14] that, in some cases, cryptographic means are a sufficient condition. For example, Secure BGP (S-BGP) [29] is shown to be incentive compatible in those cases, as it ensures that an AS cannot announce a path it did not learn from one of its neighbors, preventing it from lying about bogus paths. It is also shown that protocols with weaker security mechanisms such as Secure Origin BGP (soBGP) [30] are also incentive compatible in restricted settings [16]. However, all these path verification mechanisms are futile in the above example. This is because AS 3’s manipulation only involves announcing a path that does exist, and is available to it at the time of the manipulation. Hence, AS 3’s manipulation could not have been detected by path verification mechanisms that only monitor the route announcements.

6.2 NS-BGP is Not Incentive-Compatible

We observe that the above counter-example for the incentive-compatibility of BGP can be easily extended to the NS-BGP setting. Now, assume that ASes 1 and 2 made their ranking functions known to their provider AS 3. If AS 3 honestly follows NS-BGP, it should announce path (3 4 d) to AS 2 (as it knows path (2 3 4 d) is AS 2’s most preferred path). However, as in the BGP case, doing that will cause AS 3 to lose AS 1’s (direct) traffic. If AS 3 ignores AS 2’s ranking function, and announces path (3 5 d) to AS 2 instead, it will be able to attract AS 1’s (direct) traffic and improve its gain. This simple example shows that ASes may have incentive to deviate from NS-BGP even in routing systems where the NS-BGP safety conditions hold. As in the BGP case, such manipulations cannot be prevented by enabling path verification using cryptographic mechanisms.

6.3 Not Being Incentive-Compatible Does Not Affect Stability

We argue that BGP and NS-BGP not being incentive compatible in general does not necessarily mean that the respective routing systems will become unstable in the presence of unorthodox protocol manipulations. That is, while ASes might improve certain kinds of individual gains by manipulating these protocols, such actions are unlikely to affect the global routing stability.

This is because both the BGP safety conditions (the “Gao-Rexford” conditions) and the NS-BGP safety conditions (Theorem 3.3) are motivated by and descriptive of the actual economic interests of ASes, and therefore reflect ASes’ behaviors in reality. In particular, we argue that the topology and export conditions of the NS-BGP safety conditions naturally reflect the ASes’ business interests in practice: An AS does not have an economic incentive to violate the export condition (and carry transit traffic from peers or providers for free), or the topology condition (and serve as its own direct or indirect “provider”). Given these observations, we argue that, while ASes can manipulate NS-BGP in various ways, they have no incentive (and are unlikely) to break the NS-BGP safety conditions that guarantee global routing stability.

Nevertheless, the fact that BGP and NS-BGP are not, in general, incentive compatible raises interesting research questions. For example, as argued in [15], the lack of incentive-compatibility may result in inconsistencies between the path announced by an AS and the actual path it uses to forward traffic. Although some data-path verification approaches have been proposed [31], a more general solution to the protocol manipulation problem is still an open question.

7. RELATED WORK

This paper has two main areas of related work: more flexible interdomain route selection and interdomain routing stability. Recently, there has been an increase in the interest of providing more flexibility in interdomain route selection, from theoretical formalism and modeling of policy-based routing with non-strict preferences [32], to stability conditions of correlated interdomain route selection for traffic engineering [33], to Routing Control Platform (RCP)-type systems that provide various degrees of customization support in BGP route selection [24, 25, 34].

A huge amount of effort has been put into understanding BGP’s stability properties. Griffin et al.’s seminal work modeled BGP as a distributed algorithm for solving the Stable Paths Problem, and derived a theoretic sufficient condition (i.e., “No Dispute Wheel”) for BGP stability [9]. Gao et al. proved a set of three practical conditions (i.e., the “Gao-Rexford” conditions) that guarantees BGP stability and also reflects the common business practices in today’s Internet [10]. Gao et al. later extended their results to cover backup routing with BGP protocol extension and preference guidelines [11]. Feamster et al. explored the trade-off between the expressiveness of rankings and interdomain routing safety, and found that ASes are granted with complete flexibility with export (filtering) policies (i.e., can violate the export condition of the “Gao-Rexford” conditions), only shortest-paths based ranking can guarantee stability [13].

8. CONCLUSIONS

This paper presents Neighbor-Specific BGP (NS-BGP), an extension to BGP that provides both great practical benefits to ASes that deploy it and new theoretical contributions to the understanding of the fundamental trade-off between local preference flexibility and global routing stability. The NS-BGP model we propose enables individual ASes to offer customized route-selection services to neighbors. We prove that, comparing to conventional BGP, a less restrictive sufficient condition
can guarantee the stability of the more flexible NS-BGP. Our stability conditions allow an AS to select any exportable routes for its neighbors without compromising global stability. We also show that NS-BGP supports backup routing without routing anomalies that can occur in conventional BGP (such as the “BGP Wedgies”), and NS-BGP remains stable even in partial deployment and in the presence of network failures, as long as the stability conditions are followed. We discussed the practical issues associated with deploying NS-BGP and show it can be readily deployed by individual ASes independently.

9. REFERENCES