Inherently safe backup routing with BGP

Lixin Gao, Timothy Griffin, and Jennifer Rexford

Abstract— **IP routing requires the cooperation of a large number of Autonomous Systems (ASes) via the Border Gateway Protocol (BGP). Each AS applies local policies for selecting routes and propagating routes to others, with important implications for the reliability and stability of the global system. In and of itself, BGP does not ensure that every pair of hosts can communicate. In addition, routing policies are not guaranteed be** *safe***, and may cause persistent protocol oscillations.** *Backup routing* **is often used to increase the reliability of the network under link and router failures, at the possible expense of safety. This paper presents two models for backup routing that increase global network reliability without compromising safety. Indeed, our models are** *inherently safe* **in the sense that they remain safe under any combination of link and router failures.**

I. Introduction

The Internet consists of thousands of autonomous systems (ASes) that interact to coordinate the delivery of IP traffic. An AS is a collection of routers and links administered by a single institution, such as a company, university, or Internet service provider. An institution may consist of multiple ASes. Neighboring ASes use the Border Gateway Protocol (BGP) to exchange routing information [1], [2], [3]. Each BGP route advertisement concerns a particular block of IP addresses and includes a list of the ASes in the path, along with a number of other attributes. Each AS applies local policies to select the best route and to decide whether or not to propagate this route to neighboring ASes, without divulging their policies and internal topology to others. In practice, BGP policies reflect the financial relationships between neighboring ASes. AS pairs typically have a *customer-provider* or *peer-peer* relationship. A customer pays its provider for access to the rest of the Internet, whereas peers agree to exchange traffic, usually without charge, for the good of their respective customers.

These relationships translate into rules that determine whether or not an AS exports its best routes to a neighboring AS [4], [5]. We term these rules *Normal Export Rules* (NER), which are as follows. An AS learns routes from its customers, peers, and providers. The AS propagates its best route for each block of addresses to its customers, and propagates its customers' routes to all neighboring ASes. That is, an AS provides transit service to ensure that each customer, and its downstream customers, can communicate with hosts in the rest of the Internet. In contrast, an AS does not export routes learned from one provider to another provider. Otherwise, these two providers might exchange traffic via their common customer. Similarly, an AS does not inform its peers about routes learned from other peers or providers. That is, an AS does not provide transit service for its providers and peers.

The interaction of locally defined routing policies can have global ramifications for the stability of the BGP system. Conflicting local policies among a collection of ASes can result in BGP route oscillations [7]. We call a collection of routing policies *safe* if they can never lead to BGP divergence. Verifying the safety of a set of routing policies is computationally expensive [8] and would require ASes to reveal their (often proprietary) routing policies. Safety is not guaranteed even if each AS conforms to NER. However, Gao and Rexford [9] present additional guidelines that guarantee safety. They present two basic ap-

L. Gao is at Smith College, gao@cs.smith.edu, T. Griffin and J. Rexford are at AT&T Labs, {griffin, jrex}@research.att.com.

proaches. In the first, each AS uses the *shortest paths rule* and prefers routes with shorter AS paths. In the second, each AS always prefers customer routes over peer and provider routes. This second approach provides more flexibility in expressing local routing policies and conforms to current practices.

These scenarios cover what might be called *normal routing*. The main problem with normal routing is that it does not guarantee that connectivity is maintained in the face of network failures. To address this, another type of routing, which we call *backup routing*, is commonly employed. The main idea behind backup routing is to designate some routes as *backup routes* that are to be used only in the event that other *primary routes* are not available. Two types of backup links are most common. First, some links to upstream providers may be designated as *multi-homed backups*, to be used only in case of failure of other primary links. Second, peer-to-peer links may be used for backup routing as suggested in [6]. This allows two or more peers to cooperate in providing backup connectivity. Two ASes within a single institution, such as large service provider, may have a similar backup arrangement to improve the reliability of the network without the overhead of installing additional links.

In order to increase network reliability, backup routing is often allowed to violate some of the rules imposed on normal routing. In particular, backup routing present two challenges for the guidelines presented in [9]. First, implementing peer-to-peer backups requires the violation of NER. Second, backup routing may violate the path preference guidelines of [9]. For example, a provider route may be preferred over a customer route if the customer route is a backup route.

The current paper investigates how the results of [9] can be extended to backup routing. We present guidelines for backup routing that guarantee that routing policies are *inherently safe*, which means they are safe under any combination of link and router failures. At the same time, the routing policies are *locally implementable*. That is, only coordination between an AS and its neighbors is required. Our results can be summarized as follows. We show that, in order to guarantee safety, if a route is marked as a backup route, then it must retain this marking as is traverses subsequent ASes. We then investigate two basic models. In Model I, there is just one class of backup route, and routing policies are not able to differentiate between routes that have traversed a single backup link or multiple backup links. We show that in Model I, we are forced to adopt the shortest paths rule for backup routes. In Model II, the "backup level" of a route is incremented each time it traverses a backup link. This model allows a generalization of [9] where each AS can now prefer customers to peers and providers within a backup level. Both Model I and Model II are locally implementable, but a full implementation of Model II may be difficult to realize given the current low-level nature of most router configuration languages.

We investigate backup routing in the context of the *stable paths problem*, a static formalism that captures the semantics of the dynamic interdomain routing protocol [10], [11]. This allows us to define models for backup routing that are independent of BGP-specific details. In addition, we employ the results of [10], [11] in the proofs that our models are inherently safe. Section II reviews the stable paths problem (SPP) while Section II-B reviews the main result of [9]. Section III formalizes the notion of a *multi-homed backup link* and a *peer backup link* and presents SPP versions of Models I and II. In Section IV, we describe how to realize the two models using the BGP community attribute. Section V concludes the paper with a summary of future research directions.

II. Routing Policy Model

Analyzing the convergence properties of a dynamic routing protocol like BGP is very difficult. Instead, our analysis draws on the static *Shortest Path Problem* (SPP) formalism that captures the underlying semantics of interdomain routing policies as expressed in BGP configurations [10], [11]. Then, we define customer-provider and peer-peer relationships and show that the resulting constraints on the Stable Paths Problem ensure that the routing protocol does not diverge.

A. Stable Paths Problem (SPP)

The Stable Paths Problem is a *static* formalism similar to the *shortest paths problem*, that can be described in a manner independent of any *dynamic protocol* used to solve instances of this problem [10], [11]. SPP is based on the notion of *permitted paths* and *ranking functions* on these paths. Let $G = (V, E)$ be a simple, undirected graph where $V = \{0, 1, 2, \ldots, n\}$ is the set of nodes and E is the set of edges. For any node u, $neighbors(u) = \{w \mid \{u, w\} \in E\}$ is the set of *neighbors* for u. There is a special node $o \in V$, called the *origin*, that it is the destination to which all other nodes attempt to establish a path. Our examples will often use node 0 as the origin.

A *path* in G is either the empty path, denoted by ϵ, or a sequence of nodes, $(v_k\ v_{k-1}\ \ldots\ v_1\ v_0)$, $k \geq 0$, such that for each i, $k \geq i > 0$, $\{v_i, v_{i-1}\}$ is in E. Note that if $k = 0$, then (v_0) represents the trivial path consisting of the single node v_0. Each non-empty path $P = (v_k\ v_{k-1}\ \ldots\ v_1\ v_0)$ has a direction from its *first node* v_k to its *last node* v_0. If P and Q are non-empty paths such that the first node in Q is the same as the last node in P, then PQ denotes the path formed by the *concatenation* of these paths. We extend this with the convention that $\epsilon P = P\epsilon = P$, for any path P. For example, $(4\ 3\ 2)\ (2\ 1\ 0)$ represents the path $(4\ 3\ 2\ 1\ 0)$, whereas $\epsilon\ (2\ 1\ 0)$ represents the path $(2\ 1\ 0)$ This notation is most commonly used when P is a path starting with node v and $\{u, v\}$ is an edge in E. In this case $(u\ v)P$ denotes the path that starts at node u, traverses the edge $\{u, v\}$, and then follows path P from node v.

For each $v \in V$, $\mathcal{P}^v$ denotes the set of *permitted paths* from v to the origin (node o). If $P = (v\ v_k\ \ldots\ v_1\ o)$ is in $\mathcal{P}^v$, then the node v_k is called the *next hop* of path P. Let $\mathcal{P}$ be the union of all sets $\mathcal{P}^v$. For each $v \in V$, there is a non-negative, integer-valued *ranking function* λ^v, defined over $\mathcal{P}^v$, which represents how node v ranks its permitted paths. If $P_1, P_2 \in \mathcal{P}^v$ and $\lambda^v(P_1) < \lambda^v(P_2)$, then P_2 is said to be *preferred over* P_1. Let $\Lambda = \{\lambda^v \mid v \in V - \{o\}\}$.

An instance of the *Stable Paths Problem*, $S = (G,\ o,\ \mathcal{P},\ \Lambda)$, is a graph, an origin node, the set of permitted paths from each node to the origin, and the ranking functions for each node. In addition, we assume that $\mathcal{P}^o = \{(o)\}$, and for all $v \in V - \{o\}$:

(empty path is permitted) $\epsilon \in \mathcal{P}^v$,

(empty path is lowest ranked) $\lambda^v(\epsilon) = 0$, $\lambda^v(P) > 0$ for $P \neq \epsilon$,

(strictness) If $P_1 \neq P_2$ and $\lambda^v(P_1) = \lambda^v(P_2)$, then there is a u such that $P_1 = (v\ u)P_1'$ and $P_2 = (v\ u)P_2'$ (paths P_1 and P_2 have the same next-hop),

(simplicity) If path $P \in \mathcal{P}^v$, then P is a simple path (no repeated nodes),

Let $S = (G, o,\ \mathcal{P}, \Lambda)$ be an instance of the Stable Paths Problem. A *path assignment* is a function π that maps each node $u \in V$ to a path $\pi(u) \in \mathcal{P}^u$. An assignment is *stable* if each node u selects a path $\pi(u)$ that is the highest ranked permitted path that is consistent with the path chosen by the next-hop node. For example, if $\pi(u) = (u\ v)P$, then $\pi(v) = P$. If no such assignment exists, then S is *unsolvable*. Figure 1 presents the SPP called BAD GADGET, which has no

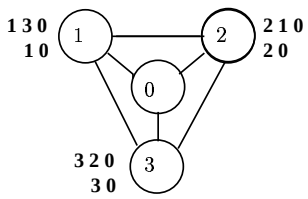

Fig. 1. BAD GADGET

solution.

A Simple Path Vector Protocol (SPVP) was defined in [10], [11] to solve the Stable Paths Problem in a distributed manner. SPVP can be thought of as an abstraction of BGP. There are two desirable properties for an instance of SPP with respect to behavior we can expect from SPVP:

Safety: An instance of the SPP is *safe* if the protocol SPVP can never diverge. The example BAD GADGET is not safe since it has no solution and so the protocol can never converge.

Inherent safety: An instance of the SPP is *inherently safe* if it is safe, and remains safe after removing any nodes, edges, or permitted paths.

Inherent safety guarantees that a system will remain safe under network failures *and* under more restrictive routing policies that filter out some permitted paths. In Section III, we introduce two approaches to providing backup paths that ensure that the SPP is inherently safe.

B. Neighbor Relationships

Motivated by the commercial contracts between autonomous systems in the Internet, we consider the possibility that adjacent nodes have either a *customer-provider* or *peer-peer* relationship. Consider a node u. We partition $neighbors(u)$ into three sets $customers(u)$, $peers(u)$, and $providers(u)$ — the customers, peers, and providers of u, respectively [9]. Relationships must be consistent between a pair of nodes. That is, if $w \in customers(u)$ then $u \in providers(w)$; similarly, if $w \in peers(u)$ then $u \in peers(w)$. We also classify a path as a customer, peer, or provider path, depending on the relationship between the first two nodes in the path. A path $(u\ w)P$ is a customer path if $w \in customers(u)$, a peer path if $w \in peers(u)$, or a provider path if $w \in providers(u)$.

Figure 2 shows an example graph where peer-peer relationships are represented by a dotted line and provider-customer relationships are represented by a solid line with an arrow pointing from the provider to the customer. Based on these relationships, we can define special cases of the stable paths problem that impose practical restrictions on the graph structure, the permitted paths, and the ranking functions. In particular, the work in [9] identified several combinations of restrictions are sufficient to guarantee that the resulting system is inherently safe. The restrictions are reasonable in the sense that they are faithful to the commercial relationships between neighboring autonomous systems. In the remainder of this section, we revisit one of these scenarios in the context of the stable paths problem and introduce terminology that will be used in the rest of the paper.

First, we assume that provider-customer relationships are hierarchical. Informally, if u is a customer of v and v is a customer of w, then w is not a customer of u. That is, the graph G does not have a cycle of edges with provider-customer relationships. Figure 2 has such a hierarchical structure. However, adding a provider-customer edge $\{0, 5\}$ would introduce a cycle involving nodes 5, 3, and 0.

Second, we assume that no permitted path has a *valley* — a provider-customer edge followed by one or more customer-provider edges. That is, a path between two nodes should not traverse an intermediate node that is lower in the hierarchy [4], [5]. For ex-

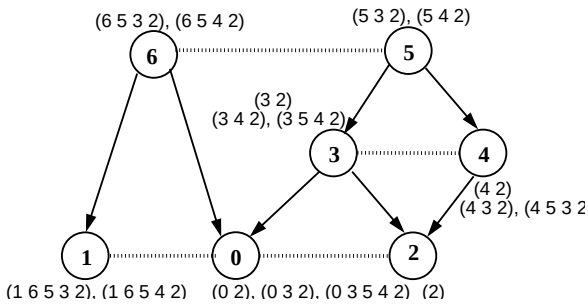

Fig. 2. AS graph with provider-customer and peer-peer relationships, with permissible paths to node 2

ample, paths $(6\ 5\ 3)$ and $(0\ 3\ 2)$ would be permissible. In contrast, $(6\ 0\ 3)$ and $(6\ 1\ 0\ 3)$ would not be permissible since they each include a valley. The path $(6\ 1\ 0\ 3)$ has a provider-customer edge $\{6, 1\}$ followed by a customer-provider edge $\{0, 3\}$. In this case, the valley includes an intermediate peer-peer edge $\{1, 0\}$.

Third, we impose restrictions on paths that include peer-peer edges. We allow *uphill* paths such as $(0\ 3\ 5)$ that consist of one or more customer-provider edges and *downhill* paths such as $(5\ 3\ 0)$ that consist of one or more provider-customer edges [12]. We also permit paths $P_1(u\ v)P_2$ where P_1 is either an empty path ϵ or an uphill path, $\{u, v\}$ is a peer-peer edge, and P_2 is either an empty path ϵ or a downhill path. For example, $(1\ 6\ 5\ 3)$ is a permissible path. However, we do not permit paths that have *steps* — an uphill or downhill path that is interrupted by one or more peer-peer edges. Formally,

(step) A peer-peer edge $\{u, v\}$ is a *step* in path $P_1(u\ v)P_2$ if either (1) the last edge of P_1 or the first edge of P_2 is a peer-peer link, or (2) the last edge in P_1 is a provider-customer edge, or (3) the first edge in P_2 is a customer-provider edge.

For example, the path $(5\ 4\ 3\ 2\ 0)$ is not permissible because it has two steps: $\{4, 3\}$ and $\{2, 0\}$. Likewise, the path $(1\ 0\ 2)$ is not permissible because it has two steps: $\{1, 0\}$ and $\{0, 2\}$.

Fourth, we assume that customer paths are preferable to peer and provider paths. For example, node 5 would rank the customer path $(5\ 3\ 0)$ higher than peer path $(5\ 6\ 0)$. We do not impose any restriction on the ranking of different customer paths, or of different peer and provider paths.

In summary, we consider a stable paths problem with the following four properties:

(acyclic) The directed graph induced by the customer-provider relationships is acyclic.

(no-valley) If $(v_k\ \ldots\ v_1\ v_0) \in \mathcal{P}$ and $v_{j-1} \in customers(v_j)$ for some $j = k, \ldots, 1$, then $v_{i-1} \neq providers(v_i)$ for all $i = j-1, \ldots, 1$.

(no-step) If $(v_k\ \ldots\ v_1\ v_0) \in \mathcal{P}$ and $v_{j-1} \in peers(v_j)$ for some $j = k, \ldots, 1$, then $v_{h-1} \in providers(v_h)$ for $h = k, \ldots, j-1$ and $v_{i-1} \in customers(v_i)$ for $i = j-1, \ldots, 1$.

(prefer-customer) If $w \in customers(u)$ and $v \in peers(u) \cup providers(u)$, then $\lambda((u\ w)P_1) > \lambda((u\ v)P_2)$ for all paths P_1 and P_2.

These assumptions result in the following theorem from [9], expressed in SPP terminology:

Theorem II.1: Any SPP S with the acyclic, no-valley, no-step, and prefer-customer properties is inherently safe.

The theorem is a corollary of Theorem III.2, presented later in Section III-C.

III. Backup Formulation

The restrictions outlined in Section II-B ensure that the system in Figure 2 is inherently safe. However, removing edges from graph would disconnect certain pairs of nodes. Consider the graph after removing the edge $\{5, 3\}$. The new solution to the stable paths problem may select different paths for some pairs of nodes. For example, deleting edge $\{5, 3\}$ would remove the customer path $(5\ 3\ 2)$, forcing node 5 to use the other

customer path $(5\ 4\ 2)$ to reach node 2. However, node 5 would have to use a peer path $(5\ 6\ 0)$ to reach node 0, since the customer path $(5\ 3\ 0)$ is no longer available. If $\{5, 3\}$ and $\{4, 2\}$ both fail, then nodes 1, 5, and 6 do not have *any* permissible path to 2, even though the graph is still connected. Ensuring that these nodes have a non-empty path to 2 requires relaxing the restrictions on the set of permitted paths. For example, relaxing the no-step restriction would make $(5\ 4\ 3\ 2)$ into a permissible paths with a step involving edge $\{4, 3\}$.

Enlarging the set of permissible paths has important implications on whether the system remains inherently safe. On the one extreme, every path could be permissible. However, paths with valleys violate the basic notion of hierarchical relationships between nodes. Instead, we define a slightly weaker notion of reachability where the set $\mathcal{P}$ of permitted paths can include any path that does not have a valley. That is, in contrast to Section II-B, we allow paths with steps to be used as *backup paths*. A backup path is a permitted path P with one or more *backup edges*. A path with no backup edges is a *primary path*. A backup edge can arise in two ways, depending on the relationship between the adjacent nodes:

(multi-homed backup) For each node u, the set $providers(u)$ is partitioned into $backup_providers(u)$ and $primary_providers(u)$. Each edge $\{u, v\}$ with $v \in backup_providers(u)$ is a backup edge. Any path including edge $\{u, v\}$ is a backup path.

(peer backup) For each node u and each node $v \in peers(u)$, the edge $\{u, v\}$ is a backup edge in path P if $\{u, v\}$ is a step in the path. Otherwise, $\{u, v\}$ is not a backup edge.

In the multi-homed backup scenario, node u has an edge to provider v specifically for the purpose of having backup paths. In the peer backup scenario, node u has a peer v that can appear in backup paths to and from v. The main goal of this paper is to determine how to allow nodes to have multi-homed backup and peer backup relationships without compromising the inherent safety of the global system. To achieve this goal, we are forced to revisit the prefer-customer assumption. For simplicity, our examples consider cases where all providers are primary providers and the permissible set $\mathcal{P}$ consists of all no-valley paths.

A. Lower Ranking for Backup Paths

First, we consider how backup paths should be ranked. In the example in Figure 3, nodes 2, 3, and 4 each have a provider-customer relationships with node 1 and peer-peer relationships with each other. Node 1 also has a peer-peer relationship with 0, which is a customer of 2. Limiting the consideration to primary, valley-free paths, nodes 2, 3, and 4 would each select a path that uses the link from 2 to 0, and node 1 would select a direct path. Now, suppose that the set of permissible paths is extended to include paths with steps, such as $(2\ 1\ 0)$ and $(3\ 4\ 1\ 0)$. Nodes 2, 3, and 4 should still prefer the primary path that uses the link from 2 to 0. That is,

(prefer primary paths) If path P_1 has no backup edges and path P_2 has one or more backup edges, then $\lambda(P_1) > \lambda(P_2)$.

Note that having a backup edge is a *global* property. That is, if a path P has a backup edge, then $(u\ v)P$ has a backup edge. If the link between 0 and 2 fails, nodes 2, 3, and 4 select paths to 0 that include node 1. Each of these paths has a backup edge. It does not matter where the backup edge occurs. For example, node 2 should not prefer the path $(2\ 3\ 1\ 0)$ over path $(2\ 1\ 0)$ just because the subpath $(2\ 3\ 1)$ does not involve a step. Otherwise, the example in Figure 3 could devolve to the BAD-GADGET scenario in Figure 1.

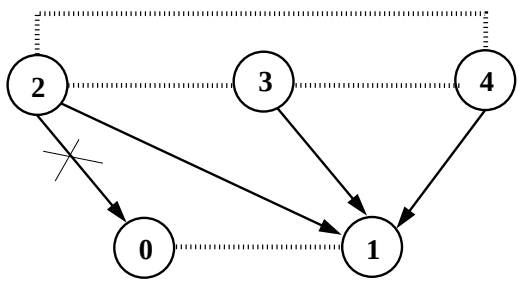

Fig. 3. Backup routes should be global

B. Model I : Preferring Shorter Backup Paths

Next, we consider how the ranking functions should differentiate between backup paths. We consider a simple SPP model where each node selects a primary path, if available, or the shortest possible backup path:

(prefer-customer-among-primary-paths) If $(u\ \ w)P_1$ and $(u\ v)P_2$ are primary paths, and $w \in customer(u)$ and $v \in peer(u) \cup provider(u)$, then $\lambda((u\ w)P_1) > \lambda((u\ v)P_2)$.

(shortest-backup-path) If $P_1, P_2 \in \mathcal{P}$ are backup paths, and $|P_1| < |P_2|$, then $\lambda(P_1) > \lambda(P_2)$.

This has the desirable property of minimizing the length of backup paths. We can prove that

Theorem III.1: Any SPP S with the acyclic and no-valley properties that obeys the prefer-primary-paths, prefer-customer-among-primary-paths, and shortest-backup-path properties at each node is inherently safe.

Proof: The full paper contains a detailed proof. Informally, ranking primary paths over backup paths reduces the problem to two subproblems related to primary paths and backup paths, respectively. The prefer-customer requirement addresses the subproblem concerning primary paths, and the shortest-path requirement addresses the subproblem concerning backup paths. Each case is treated using the sufficient condition for safety presented in [10], [11]. ■

Requiring each node to prefer shorter backup paths over longer backup paths is important. Consider the example in Figure 4, where each node prefers customer backup paths over peer and provider backup paths. Also, suppose that each node ranks among customer routes (or among peer and provider routes) based on the length of the path. Nodes 1, 2, 3, and 4 do not have any primary (step-free) paths to node 0. The prefer-customer property would result in node 4 preferring a path via 3, node 3 preferring a path via 2. Node 1 may prefer the shorter customer path via 4 over the path via 2. In addition, node 2 may prefer the shorter provider route via 1 over the peer route via 8. This system does not have a stable path assignment. Yet, if each node used path length to rank among backup paths, this problem would not arise.

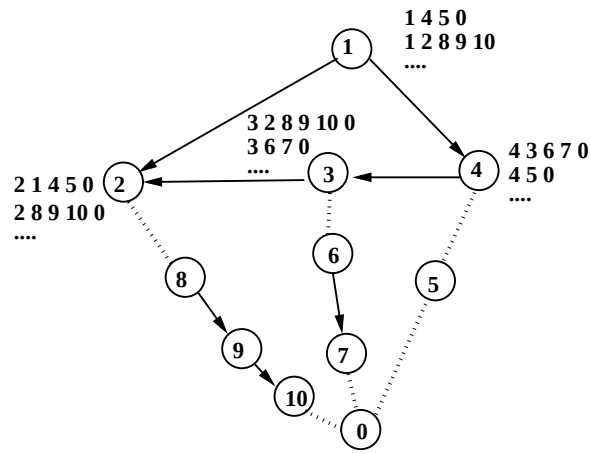

Fig. 4. A counterexample to inherent safety when customer routes can be preferred over peer routes

This example shows that a policy that prefers a customer backup paths over peer and provider backup paths may result in a system that is not inherently safe. Similar examples can be shown for preferring peers over providers, preferring providers over peers, and for preferring customers over peers. We conclude that having just two classes of paths (primary and backup) forces us to use the shortest paths rule in ranking paths.

C. Model II : Preferring Paths With Fewer Backup Edges

Model I has two disadvantages. First, the shortest-path policy does not provide any flexibility in ranking among backup paths. Second, the shortest-path policy would cause a node to favor a path with a large number of backup edges with a small number of backup edges, whenever the path with more backup edges has a shorter overall length. This may be an undesirable property since a path with a large number of backup edges imposes on a large number of backup relationships. We consider another model that ranks paths based on the number of backup edges, and ranks paths with the same number of backup edges based on the customer, peer, and provider relationships:

(prefer-fewer-backup-edges) If P_1 has s_1 steps and P_2 has s_2 backup edges, and $s_1 < s_2$, then $\lambda(P_1) > \lambda(P_2)$.

(prefer-customer-within-backup-class) If $(u\ w)P_1$ and $(u\ v)P_2$ have the same number of backup edges, and $w \in customer(u)$ and $v \in peer(u) \cup provider(u)$, then $\lambda(P_1) > \lambda(P_2)$.

Note that this model is a generalization of the system described in Section II-B. We can prove that

Theorem III.2: Any SPP S with the acyclic and no-valley properties that obeys the prefer-fewer-backup-edges and the prefer-customer-within-backup-class properties at each node is inherently safe.

Proof: The full paper contains a detailed proof. Informally, ranking paths with s_1 backup edges higher than paths with $s_2 > s_1$ backup edges reduces the problem to a collection of subproblems, each corresponding to paths with a fixed number of backup edges. Each subproblem is addressed by the prefer-customer requirement, in a manner similar to the proof of Theorem III.1. ■

Returning to the example in Figure 4, nodes 2 and 3 each have a permissible path with a single backup edge. For example, node 3 would rank the path $(3\ 6\ 7\ 0)$ with one backup edge $\{7, 0\}$ higher than the path $(3\ 2\ 8\ 9\ 10\ 0)$ with two backup edges $\{2, 8\}$ and $\{10, 0\}$. Similarly, node 2 would prefer the path $(2\ 8\ 9\ 10\ 0)$ with one backup edge $\{10, 0\}$. Node 4 has two backup paths, $(4\ 5\ 0)$ and $(4\ 3\ 6\ 7\ 0)$, each with two backup edges; hence, node 4 prefers the customer path $(4\ 3\ 6\ 7\ 0)$. The resulting system is inherently safe.

IV. Implementing backup routing in BGP

We now consider how Models I and II might be implemented in BGP. BGP exchanges route announcements between BGP speaking routers. These announcements are records containing a destination (IP prefix) and attributes associated with this destination. Route announcements include the following attributes.

nlri	:	network layer reachability information (address block for a set of destinations)
next_hop	:	next hop (address of next hop router)
as_path	:	ordered list of ASes traversed
local_pref	:	local preference
c_set	:	set of community values

The local preference attribute **local_pref** is not passed between autonomous systems, but is used internally within an autonomous system to assign a local degree of preference. Community values [15] are typically used in routing policies for deciding on the value of **local_pref** or on filtering.

The BGP attributes are used by *import policies* and *export policies* at each router to implement its *routing policies*. A *route transformation* $\mathcal{T}$ is a function on route records, $\mathcal{T}(r) = r'$, that operates by deleting, inserting, or modifying the attribute values of r. If $\mathcal{T}(r) = \langle\rangle$ (the empty record), then we say that r has been *filtered out* by $\mathcal{T}$. Suppose u and w

are autonomous systems with a direct BGP relationship. As a record r moves from w to u it undergoes three transformations. First, $r_1 = \text{export}(w \rightarrow u, r)$ represents the application of *export policies* (defined by w) to r. Second, $r_2 = \text{PVT}(u \leftarrow w, r_1)$ is the BGP-specific *path vector* transformation that adds w to the **as_path** of r_1, sets **next_hop**, and filters out the record if its **as_path** contains u. Finally, $r_3 = \text{import}(u \leftarrow w, r_2)$ represents the application of import policies (defined at u) to r_2. In particular, this is the function that assigns a **local_pref** value for r_3.

A. Implementing Model II

RFC 1998 [6] suggests using communities as a way to influence a neighbor's routing policies. It applies this technique to the implementation of backup peering.

Our implementation of Models I and II uses the basic technique suggested in RFC 1998. However, we implement formally defined models and prove the correctness of the implementations. We first define an implementation of Model II, and then show that an implementation of Model I can be obtained by simplification.

We assume that each AS w has defined the following community values, and that the semantics of these labels is shared with w neighbors.

$(w : bu : i)$ tag for backup route of level i,
$(w : up)$ tag for provider routes,
$(w : down)$ tag for customer routes,
$(w : peer)$ tag for the customer routes of a peer,
$(w : peerbu)$ tag for the upstream routes of a peer.

We also assume that for any neighbor u, and any backup level i, that $lpbu(w, u, i)$ represents the local preference assigned to routes from u received at w. For primary routes, we assume that function $lpnorm(w, u)$ is defined. We assume that for each w,

```
import(w ← u, r) =
begin
  if u : bu : i ∈ r.c_set then
    r.local_pref := lpbu(w, u, i);
    r.c_set := {w : down, w : bu : i};
  else
    r.local_pref := lpnorm(w, u);
    r.c_set := {w : down};
  permit(r);
end

export(w → u, r) =
begin
  S := {w : down, w : up, w : peer, w : peerbu};
  if S ∩ r.c_set ≠ φ then permit(r);
  else deny(r);
end
```

Fig. 5. Model II routing policy at w for $u \in customers(w)$.

these functions conform to the following rules:

- If $u \in customers(w)$, $v \in peers(w) \cup providers(w)$, then $lpnorm(w, v) < lpnorm(w, u)$, and $lpbu(w, v, i) < lpbu(w, u, i)$ for each i.
- For each i, $lpbu(w, u, i) < lpnorm(w, u)$.
- For each i, u, v, $lpbu(w, u, i + 1) < lpbu(w, v, i)$.

That is, primary routes are always preferred over backup routes, and backup routes of level i are always preferred over backup routes of level $i + 1$. Within each level, customer routes are preferred over peer and provider routes.

Figure 5 shows how a node w implements this policy in exchanging BGP advertisements with a customer u. The command $deny(r)$ filters out the record r, while $permit(r)$ allows the route to pass the filter, perhaps after some modification. Figure 6 presents the routing policy for providers, while Figure 7 presents the policy for peers. These policies are complicated by the "arithmetic" required on backup levels. Some

additional explanation is required for the export policy to peers. The last two "else if" clauses are required to allow backup transit between a node's peers. In this case two peering links in a row will be encountered, and both must be treated as backup links. In addition, a peer's upstream routes (with $w : peerbu$ tag) are passed to other peers as $w : up$ routes, since if these are actually used the peer u will be acting as a (temporary) upstream provider to w. Similarly, a peer's downstream routes (with $w : peer$ tag) are passed to other peers as $w : down$ routes, since if these are actually used the peer u will be acting as a (temporary) customer of w.

Theorem IV.1: The routing policies described in Figures 5, 6, and 7 implement Model II, and are thus inherently safe.

Proof: The proof is included in the full paper. It relies on a formal definition of the translation from a set of BGP routing policies into an instance of SPP. The proof is then by induction on the length of any permitted path. ■

B. Implementing Model I

The implementation of Model II can be simplified to obtain an implementation of Model I. First, the tags $w : bu : i$ are replaced by a single value $w : bu$, and all clauses that increment the backup level are modified in the obvious way. Second, each line of the form $r.\mathbf{local_pref} := lpbu(w, w, i)$ is replaced by $r.\mathbf{local_pref} := low(w)$, where $low(w)$ is low value of local preference used for all backup routes at w.

V. Conclusions

Selecting efficient, stable routes between each pair of hosts is a major challenge for the distributed Internet routing infrastructure. In and of itself, BGP does not ensure that hosts can communicate, even if the net-

```
import(w ← u, r) =
begin
  if u ∈ backup_providers(w) then
    if u : bu : i ∈ r.c_set then
      r.local_pref := lpbu(w, u, i + 1);
      r.c_set := {w : up, w : bu : i + 1};
    else
      r.local_pref := lpbu(w, u, 1);
      r.c_set := {w : up, w : bu : 1};
  else
    if u : bu : i ∈ r.c_set then
      r.local_pref := lpbu(w, u, i);
      r.c_set := {w : up, w : bu : i};
    else
      r.local_pref := lpnorm(w, u);
      r.c_set := {w : up};
  permit(r);
end

export(w → u, r) =
begin
  if u ∈ backup_providers(w) then
    if w : down ∈ r.c_set then
      if w : bu : i ∈ r.c_set then
        r.c_set := {w : bu : i + 1};
      else
        r.c_set := {w : bu : 1};
      permit(r);
    else if w : peer ∈ r.c_set then
      if w : bu : i ∈ r.c_set then
        r.c_set := {w : bu : i + 2};
      else
        r.c_set := {w : bu : 2};
      permit(r);
    else deny(r);
  else
    if w : down ∈ r.c_set then
      permit(r);
    else if w : peer ∈ r.c_set then
      if w : bu : i ∈ r.c_set then
        r.c_set := {w : bu : i + 1};
      else
        r.c_set := {w : bu : 1};
      permit(r);
    else deny(r);
end
```

Fig. 6. Model II routing policy at w for $u \in providers(w)$.

```
import(w ← u, r) =
begin
  if u : bu : i ∈ r.c_set then
    r.local_pref := lpbu(w, u, i);
    if u : up ∈ r.c_set then
      r.c_set := {w : bu : i, w : peerbu};
    else
      r.c_set := {w : bu : i, w : peer};
  else
    r.local_pref := lpnorm(w, u);
    r.c_set := {w : peer};
  permit(r);
end

export(w → u, r) =
begin
  if w : down ∈ r.c_set then
    permit(r);
  else if w : up ∈ r.c_set then
    if w : bu : i ∈ r.c_set then
      r.c_set := {w : up, w : bu : i + 1};
    else
      r.c_set := {w : up, w : bu : 1};
    permit(r);
  else if {w : peerbu, w : bu : i} ⊆ r.c_set then
    r.c_set := {w : up, w : bu : i + 2};
    permit(r);
  else if w : peer ∈ r.c_set then
    if w : bu : i ∈ r.c_set then
      r.c_set := {w : down, w : bu : i + 2};
    else
      r.c_set := {w : down, w : bu : 2};
    permit(r);
  else deny(r);
end
```

Fig. 7. Model II routing policy at w for $u \in peers(w)$.

work is connected. In addition, conflicting local policies amongst a collection of ASes can cause the protocol to oscillate. This paper has has presented two models for backup routing that increase global network reliability without compromising the stability of the routing protocol.

Several issues remain to be addressed. First, the interaction of backup routing with prefix aggregation needs to be studied. Second, the models presented in this paper are simplified in that they ignore the internal structure of ASes. Routing policies may not actually be AS-wide, but may vary between border routers, mostly to meet traffic engineering goals. The interaction and tradeoffs between external routing (both normal and backup) and internal routing should be explored.

REFERENCES

[1] Y. Rekhter and T. Li, "A Border Gateway Protocol 4 (BGP-4)." Request for Comments 1771, March 1995.

[2] B. Halabi, *Internet Routing Architectures*. Cisco Press, 1997.

[3] J. W. Stewart, *BGP4: Inter-Domain Routing in the Internet*. Addison-Wesley, 1998.

[4] G. Huston, "Interconnection, peering, and settlements," in *Proc. INET*, June 1999.

[5] C. Alaettinoglu, "Scalable router configuration for the Internet," in *Proc. IEEE IC3N*, October 1996.

[6] E. Chen and T. Bates, "An Application of the BGP Community Attribute in Multi-home Routing." Request for Comments 1998, August 1996.

[7] K. Varadhan, R. Govindan, and D. Estrin, "Persistent route oscillations in inter-domain routing," Tech. Rep. 96-631, USC/ISI, February 1996.

[8] T. G. Griffin and G. Wilfong, "An analysis of BGP convergence properties," in *Proc. ACM SIGCOMM*, September 1999.

[9] L. Gao and J. Rexford, "Stable Internet routing without global coordination," in *Proc. ACM SIGMETRICS*, June 2000.

[10] T. G. Griffin, F. B. Shepherd, and G. Wilfong, "Policy disputes in path-vector protocols," in *Proc. Inter. Conf. on Network Protocols*, November 1999.

[11] T. Griffin and G. Wilfong, "A safe path vector protocol," in *Proc. IEEE INFOCOM*, March 2000.

[12] L. Gao, "On inferring autonomous system relationships in the Internet," in *Proc. IEEE Global Internet Symposium*, November 2000.

[13] B. Halabi, *Internet Routing Architectures*. Cisco Press, 1997.

[14] J. W. Stewart, *BGP4, Inter-Domain Routing in The Internet*. Addison-Wesley, 1998.
[15] R. Chandra, P. Traina, and T. Li, "BGP communities attribute." RFC 1997, August 1996.