Inherently Safe Backup Routing with BGP

Lixin Gao†, Timothy G. Griffin‡, and Jennifer Rexford‡

†Electrical and Computer Engineering
University of Massachusetts
Amherst, MA 01002
lgao@ecs.umass.edu

‡Internet and Networking Systems
AT&T Labs – Research
Florham Park, NJ 07932
{griffin,jrex}@research.att.com

Abstract—**The Internet consists of a large number of Autonomous Systems (ASes) that exchange routing information using 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 protocol divergence. *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 a general model for backup routing that increases network reliability while allowing each AS to apply local routing policies that are consistent with the commercial relationships it has with its neighbors. In addition, our model is *inherently safe* in the sense that the global system remains safe under any combination of link and router failures. Our model and the proof of inherent safety are cast in terms of the *stable paths problem*, a static formalism that captures the semantics of interdomain routing policies. Then, we describe how to realize our model in BGP with locally-implementable routing policies. To simplify the specification of local policies, we propose a new BGP attribute that conveys the *avoidance level* of a route. We also describe how to realize these policies without modification to BGP by using the BGP community attribute.**

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. 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 commercial relationships between neighboring ASes. AS pairs typically have a *customer-provider* or *peer-peer* relationship. A provider provides connectivity to the Internet as a service to its customers, whereas peers simply provide connectivity between their respective customers. The relationships between ASes translate into local rules that that determine whether or not an AS exports its best routes to a neighboring AS [4], [5].

These policies limit the possible paths between each pair of Internet hosts. In and of itself, BGP does not ensure that every pair of hosts can communicate, even if the underlying topology is connected. In addition, link and router failures may cause an AS to withdraw a route, forcing some of the ASes to select alternate paths. However, these alternate paths are typically constrained by the commercial relationships between the ASes. As a result, some AS pairs may not have alternate routes in certain failure scenarios. Therefore, ASes are increasingly moving beyond traditional customer-provider and peer-peer relationships to form *backup* relationships to provide connectivity in the event of a network failure.

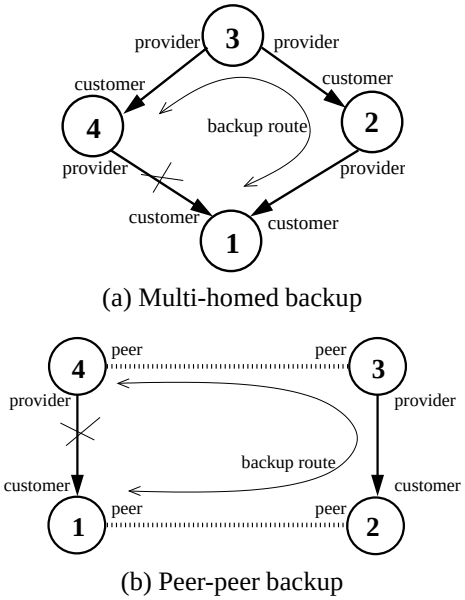


Fig. 1. Backup routes

These local backup arrangements between ASes introduce additional BGP route advertisements that announce the backup routes. On the surface, the announcement of backup routes should improve the robustness of the Internet in the face of link and router failures. However, the backup routes have negative implications on the global properties of the Internet routing system. The backup routes can introduce global BGP convergence problems that result in protocol divergence, which undermines the robustness of the network. Even if the system converges, the use of backup routes incurs a global cost in terms of increased convergence delay after the withdrawal of a route [6]. In this paper, we focus on ensuring that the system converges rather than analyzing how backup routes affect the convergence time.

Two kinds of backup arrangements are most common in practice. First, a *multi-homed backup* relationship involves using a

L. Gao was was supported in part by NSF grant ANI-9977555, and NSF CAREER Award grant ANI-9875513. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

secondary provider-customer link if the connection to the primary provider fails, as illustrated in Figure 1(a). If the link between AS 1 and AS 4 fails, then traffic to and from AS 1 travels via the secondary provider AS 2. Second, a *peer-peer backup* relationship uses an existing peer-peer link for backup routes in the event of a failure, as suggested in RFC 1998 [7]. For example, consider a university consisting of two campuses (AS 1 and AS 2) that each have a different provider, as shown in Figure 1(b). Normally the peer-peer link would only carry traffic between the two campuses. However, if AS 1 loses its connection to its upstream provider (AS 4), then traffic from AS 1 to AS 3 and AS 4 would travel via AS 2. Likewise, traffic from AS 3 and AS 4 to AS 1 would travel via AS 2. This requires AS 2 to advertise its ability to reach destinations in AS 1 to AS 3; likewise, AS 2 must advertise routes learned from AS 3 to AS 1. In this paper, we demonstrate that these additional advertisements can cause global convergence problems, unless certain basic guidelines are followed.

Previous research has demonstrated that 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 [8]. 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 [9] and would require ASes to reveal their (often proprietary) routing policies. In practice, these routing policies depend on the commercial relationships with the neighboring ASes. Focusing on customer-provider and peer-peer relationships, the work in [10] demonstrates that local policies that favor routes via customers over routes via peers and providers ensure that the global BGP system converges. However, backup routes introduce two new challenges. First, under both the multi-homed and peer-peer backup relationships, an AS may prefer a path via a provider or peer rather than choosing a customer route that traverses one or more backup links. Second, under the peer-peer backup relationship, an AS violates the traditional routing policies by advertising its peer's routes to other peers or upstream providers.

In this paper, we extend the results of [10] to backup routing. We present routing policy guidelines that guarantee that the BGP system is *inherently safe* — 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 it propagates to subsequent ASes. An AS should always prefer a primary route over a backup route to the same destination. Then, we investigate policies for selecting among a set of backup routes to the same destination. We show by example that preferring backup routes via a customer over backup routes via a peer or provider can result in an unsafe system. Instead, we introduce the notion of an *avoidance level* that increases as a path traverses additional backup edges. This model allows a generalization of [10] where each AS can now prefer customers to peers and providers within an avoidance level.

Proving properties about a dynamic, distributed protocol is very difficult in practice. This is especially true for BGP due to the complex interaction of local routing policies. Instead, we investigate backup routing in the context of the *stable paths problem* (SPP), a static formalism that captures the semantics of the interdomain routing policies [11], [12], [13]. This allows us to define a high-level, global model of backup routing that is independent of the low-level implementation of local policies. In addition, we employ the results of [11], [12], [13] in the proofs that our model is inherently safe. Section II-A reviews the stable paths problem (SPP) while Section II-B reviews the main result of [10]. Section III discusses how to support peer-peer backup relationships without sacrificing the inherent safety of the underlying system. Section IV presents a general model of backup routing that also incorporates multi-homed backup relationships and load balancing. In Section V, we propose a new BGP attribute for propagating the avoidance level in route advertisements and describe how to implement our scheme without modifying BGP by using the BGP community attribute. Section VI concludes the paper with a summary of future research directions. In Appendix VII we prove that BGP realizations of our SPP model would not suffer from protocol divergence, even under link and router failures.

II. Routing Policy Model

This section reviews the *Stable Paths Problem* (SPP) formalism of [11], [12], [13] and revisits the key result of [10] in the context of SPP.

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. 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 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,

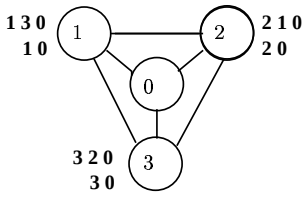


Fig. 2. BAD GADGET

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 2 presents the SPP called BAD GADGET, which has no solution.

A Simple Path Vector Protocol (SPVP) was defined in [12], [13] 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 Appendix VII, we present a sufficient condition for an SPP to be inherently safe.

As outlined in Section V, a given set of BGP routing policies gives rise to a separate instance of SPP for each IP address block announced. The nodes of the graph G model either individual routers or entire autonomous systems. Since the presentation in this paper will ignore the details of BGP configuration within an autonomous system, we will usually identify nodes with ASes.

B. Neighbor Relationships

Motivated by the commercial relationships between autonomous systems in the Internet, we consider the possibility that adjacent nodes have either a *customer-provider* or *peer-peer* relationship. Although a customer-provider or peer-peer relationship may apply across all blocks of IP addresses, our analysis allows for the possibility of ASes having different relationships for each address block. 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. 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 3 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 [10] identified several combinations of restrictions that 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 use the provider-customer relationship to impose directionality on some of the edges in the graph G. The *provider-customer digraph* refers to the directed graph resulting from the provider-customer edges of G. We assume that provider-customer relationships are hierarchical; in other words, the provider-customer digraph is acyclic. Informally, if u is a customer of v and v is a customer of w, then w is not a customer of u. Figure 3 has such a hierarchical structure. However, adding a provider-customer edge $\{0, 5\}$ would introduce a cycle involving nodes 5, 3, and 0. This assumption is consistent with the commercial relationships between ASes. For example, a nation-wide AS may be a provider for a regional AS, which in turn may be a provider for a particular university campus, but this university campus would not be a provider for the nation-wide AS.

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, since this would cause a customer to transit traffic between two of its upstream providers [4], [5]. For example, 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 [14].

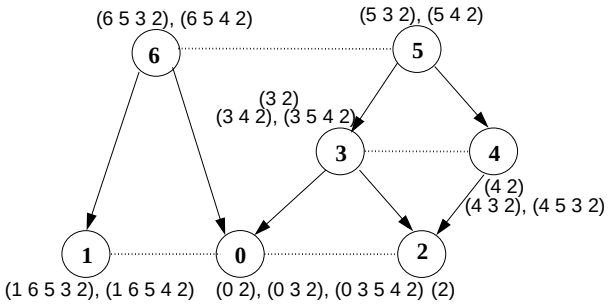


Fig. 3. AS graph with provider-customer and peer-peer relationships, with permissible paths to node 2

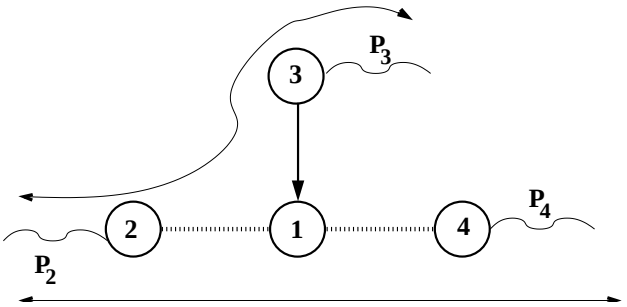


Fig. 4. Node 1 providing backup service for node 2

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* since this would violate the restriction that peers only transit traffic between their respective customers. Formally, a peer-peer edge $\{u, v\}$ is a *step* in path $P_1(u\ v)P_2$ if either the last edge of P_1 is a peer-peer or provider-customer edge, or 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 $(5\ 6\ 0)$. This is consistent with the commercial relationships between ASes. An AS has a financial incentive to limit the amount of traffic that traverses peer and provider links, and may receive payment based on the amount of traffic that travels via customer links. 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 provider-customer digraph) 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 [10], expressed in SPP terminology:

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

The theorem is a special case of Theorem IV.1, presented in the next section.

III. Inherently Safe Routing With Steps

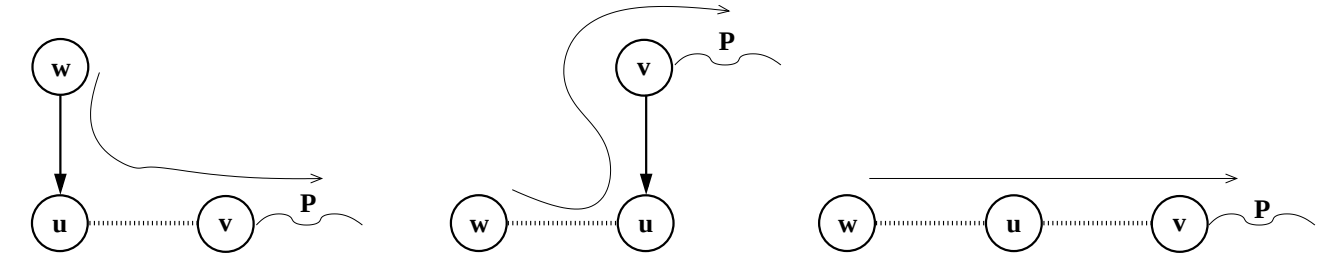
Peer-peer backup relationships violate the restrictions introduced in Section II-B. After introducing the notion of a backup path, we explain why backup paths should always have a lower ranking than other paths. Then, we explore policies for ranking between backup paths and show that preferring customer backup paths over peer and provider backup paths can result in an unsafe system. Finally, we propose a policy that prefers paths with fewer steps over paths with more steps, with preference for customer paths among paths with the same number of steps.

A. Backup Paths With Steps

Normally, the selection and export of routes is constrained by the customer-provider and peer-peer relationships. Backup relationships introduce additional paths that provide connectivity in the event of a failure and change how nodes select among paths to the same destination. Consider the example in Figure 3 after removing the edge $\{5, 3\}$. The new solution to the stable paths problem may select different paths for some node pairs. 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 node 2, even though the graph is still connected. Ensuring that these nodes have a path to node 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 of permitted paths can include paths that have steps. Figure 4 shows an example where node 1 provides backup service for node 2. This backup relationship allows paths like $(2\,1\,3)P_3$ and $(2\,1\,4)P_4$ to ensure that node 2 can reach destinations in the rest of the Internet. In addition, path like $(4\,1\,2)P_2$ and $(3\,1\,2)P_2$ ensure that the rest of the Internet can reach node 2 and its downstream customers.

These paths fall into the three categories shown in Figure 5. In Figure 5(a), node v has a backup relationship with node u to allow backup paths $(w\,u\,v)P$, corresponding to the path $(3\,1\,2)P_2$



(a) Provider−customer, peer−peer (b) Peer−peer, customer−provider (c) Peer−peer, peer−peer

Fig. 5. Allowing backup paths $(w\,u\,v)P$ with a step

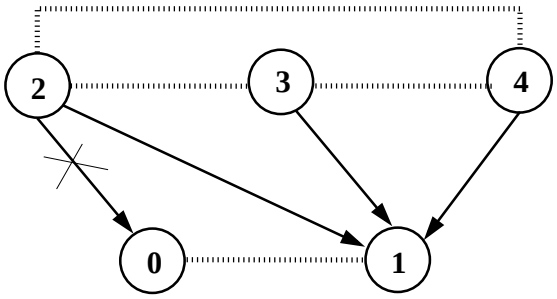


Fig. 6. Backup paths should be global

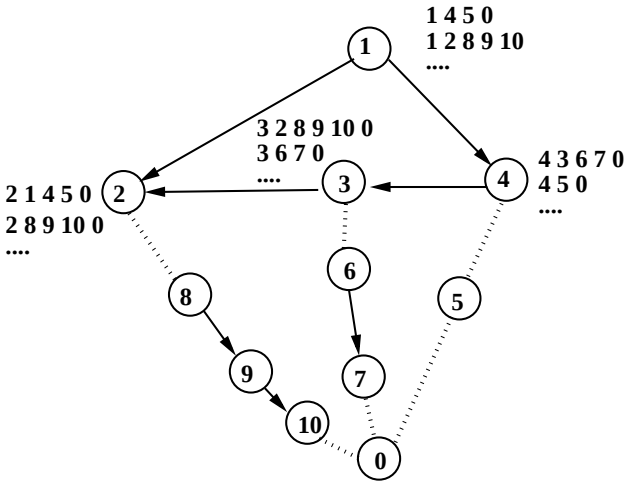


Fig. 7. Unsafe system when customer backup paths are preferred over peer backup paths

in Figure 4. In Figure 5(b), node w has a backup relationship with node u to allow backup paths $(w\,u\,v)P$ through u's provider node v, corresponding to the path $(2\,1\,3)P_3$. In Figure 5(c), node w has a backup relationship with node u to include backup paths $(w\,u\,v)P$, corresponding to both paths $(2\,1\,4)P_4$ and $(4\,1\,2)P_2$.

B. Lower Ranking for Paths With Steps

A backup path should have *global* significance. That is, if a path P is a backup path, then $(u\,v)P$ is also a backup path. A backup path should not be used unless all primary paths are unavailable. More formally, if path P_1 has no steps and path P_2 has one or more steps, then $\lambda(P_1) > \lambda(P_2)$. Having a lower ranking for backup paths is important for the safety of the system. In the example in Figure 6, nodes 2, 3, and 4 each have a provider-customer relationship 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 edge 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 $(3\ 1\ 0)$, following the approach in Figure 5(a). Nodes 2, 3, and 4 should still prefer the primary path that uses the edge from 2 to 0. If the edge between 0 and 2 fails, nodes 2, 3, and 4 select paths to 0 that include node 1. Each of these paths has a step. 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 6 could devolve to the BAD GADGET scenario in Figure 2.

C. Ranking Among Paths With Steps

In addition to preferring primary paths over backup paths, each node needs to rank amongst the backup paths. The simplest policy *ranks the backup paths based on the path length, favoring shorter backup paths over longer backup paths*. This approach has the desirable property of minimizing the length of backup paths. In addition, always selecting the shortest backup path ensures that the system is inherently safe. 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 in Section II-B addresses the subproblem concerning primary paths, and the shortest-path requirement addresses the subproblem concerning backup paths. However, the shortest-path policy is very restrictive, since it does not incorporate information about the commercial relationships between nodes or the number of steps in the backup paths. The shortest-path policy would prefer a provider path with two steps over a longer customer path with one step.

Extending the prefer-customer policy to backup paths is an appealing first approach to allowing a more liberal ranking of backup paths. In this scheme, a node would assign a *higher rank to a backup path through a customer over a backup path through a peer or provider*. However, this policy can result in an unsafe system. Consider the example in Figure 7, where

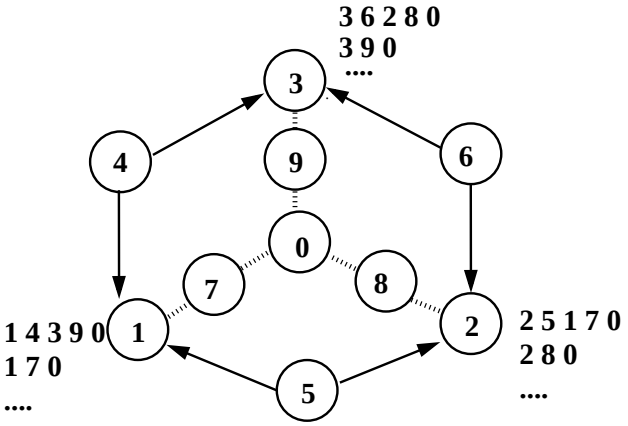


Fig. 8. Unsafe system when provider backup paths are preferred over peer backup paths

each node prefers customer backup paths over peer and provider backup paths. Also, suppose that each node ranks among customer paths (or among peer and provider paths) based on the length of the path. Nodes 1, 2, 3, and 4 do not have any step-free paths to node 0. The prefer-customer property would result in node 4 preferring a path via node 3 and node 3 preferring a path via node 2. Node 1 prefers the shorter customer path via node 4 over the path via node 2. In addition, node 2 prefers the shorter provider path via node 1 over the peer path via node 8. This system is unsafe. Similar examples can be shown for preferring peers over providers, preferring providers over peers, and for preferring customers over peers. Figure 8 shows a an example of an unsafe system where provider paths are preferred over peer paths at nodes 1, 2 and 3.

Ranking backup paths based on path length would have avoided these problems, at the expense of imposing significant restrictions on backup policies. The shortest-path policy would force a node to prefer a path with a large number of steps with a small number of steps, whenever the path with more steps has a shorter overall length. Instead, we consider another approach that ranks paths based on the number of steps. Among paths with the same number of steps, customer paths are ranked higher than peer and provider paths, consistent with the approach in Section II-B. This policy is consistent with the commercial relationships between nodes and also ensures that the system is inherently safe. Informally, ranking paths with fewer steps higher than paths with more steps reduces the problem to a collection of subproblems, each corresponding to paths with a fixed number of steps. The safety of each of these subproblems is assured by the prefer-customer requirement, as shown in the next section.

IV. Generalized Backup Routing

This section proposes a general technique for dividing the permissible paths into *avoidance levels*. Paths with a lower avoidance level are preferred over paths with a higher avoidance level; within an avoidance level, customer paths are preferred over peer and provider paths. In addition to capturing the peer-peer backup relationship (i.e., steps), this model incorporates multi-homed backup relationships and load balancing while ensuring that the system remains inherently safe.

The idea of counting the number of steps in a path suggests a general routing model based on *avoidance levels*, where each node assigns a higher rank to paths with a lower avoidance level. The assignment of an avoidance level to each path should preserve the maximum flexibility for each node to apply local policies for backup routing and load balancing. The notion of an avoidance level is more general than counting steps in a path in two ways. First, the avoidance level does not have to increase by unit increments. For example, a path with two steps could have a higher avoidance level than another path with three steps. Second, the avoidance level can increase for any edge in the path. For example, a provider-customer edge can increase the avoidance level of a path. Conceptually, a *backup edge* with respect to a path is any edge that increases the avoidance level.

The model does not prescribe how much to increase the avoidance level for each backup edge; in practice, this may depend on local policies. However, some basic restrictions on avoidance levels are necessary to ensure that the global system is inherently safe. An edge $(w\,u)$ that contributes to a step is a must be a backup edge; that is, the path $(w\,u)P$ must have a higher avoidance level than the subpath P. Any other edge may also be a backup edge with respect to a path.

We formalize these notions using a non negative function $\kappa(P)$, called an *avoidance classifier*, that obeys the following conditions:
(a) If P and Q are paths permitted at node u and $\kappa(P) < \kappa(Q)$, then $\lambda(Q) < \lambda(P)$.
(b) If P, Q, and QP are permitted paths, then $\kappa(P) \leq \kappa(QP)$.
An avoidance level is any value in the range of κ. Condition (a) states that a path P with a lower avoidance level is preferred over a path Q with a higher avoidance level. Condition (b) states that the avoidance level cannot decrease as a path traverses additional edges.

The *weight of an edge with respect to a path* P is equal to $\kappa((w\,u)P) - \kappa(P)$. An edge $\{w, u\}$ is a *backup edge with respect to path* P if the weight of edge $\{w, u\}$ with respect to P is non-zero. Note that this generalizes the notion of edge weights in two ways. First, a given edge can have different weights with respect to different paths. Second, an edge may have a weight of zero with respect to certain paths. An edge $\{w, u\}$ is a *mandatory backup edge with respect to path* $(u\,v)P$ if it matches any of the three conditions outlined in Figure 5. In particular, this means that
(a) $\{w, u\}$ is a provider-customer edge and $\{u, v\}$ is a peer-peer edge, or
(b) $\{w, u\}$ is a peer-peer edge and $\{u, v\}$ is a customer-provider edge, or
(c) $\{w, u\}$ and $\{u, v\}$ are peer-peer edges.
An avoidance classifier κ is *step aware* if for any P permitted at v and $(w\,u\,v)P$ permitted at w we have $\kappa((u\,v)P) < \kappa((w\,u\,v)P)$ whenever $\{w, u\}$ is a mandatory backup edge with respect to path $(u\,v)P$.

If κ is an avoidance classifier for SPP S, then ranking function λ is said to be *prefer-customer with respect to* κ if within a set of paths of the same avoidance level customers paths are preferred. More formally, if for every P and Q permitted at node u with $\kappa(P) = \kappa(U)$, if P is a path to one of u's customers and Q is not, then $\lambda(Q) < \lambda(P)$. Based on these definitions, we have the following theorem:

Theorem IV.1: Consider an SPP S with the acyclic and no-valley properties with a step-aware avoidance classifier κ. If the ranking function λ prefers customers with respect to κ, then S is inherently safe.
The proof appears in Appendix VII.

V. Think Globally, Act Locally

We now consider how safe backup routing might be implemented in BGP with locally-implementable routing policies that do not require global knowledge. The goal is to define a set of recommendations that, if followed by all ASes, would allow for a very flexible framework for implementing backup routes and load balancing, while at the same time provide a guarantee of the inherent safety of interdomain routing.

Our recommendations will ensure that all conforming interdomain BGP routing policies could be translated into inherently safe instances of the Stable Paths Problem for every network destination. In particular, we require that
(1) valleys are not permitted,
(2) we can define an avoidance classifier κ on the associated SPP that is step aware,
(3) the ranking of paths prefers customers with respect to the avoidance levels of κ.
We do not explicitly enforce the condition that the provider-customer digraph is acyclic, depending instead on natural market considerations. Note that we are not advocating the the translation of BGP policies into instances of the SPP. The SPP formalism is serving only as a simple semantic framework in which we can more easily prove inherent safety.

Section V-A first describes how routing policies are implemented with BGP and how BGP policies can be translated to an instance of the Stable Paths Problem. Section V-B then outlines how the rules of [10] can be implemented in BGP. Section V-C proposes a new attribute for BGP, called the *avoidance level*, and describes a very natural extension to the implementation of [10] that covers the inherently safe backup routing defined the current paper. Since the avoidance level attribute does not yet exist in BGP, Section V-D describes how it can be simulated using the BGP communities attribute, at the cost of introducing some complexity into the definition of routing policies.

A. BGP Routing Policies and Translation to SPP

BGP exchanges route announcements between BGP speaking routers. These announcements are records containing a destination network (also called an address block, CIDR block, or IP prefix) and attributes associated with this destination. Route announcements include the following attributes.

nlri	:	IP prefix
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*. These policies can delete, insert, and modify some attribute values in route announcements. They can also *filter out* announcements. As a BGP announcement moves from AS w to AS u it undergoes three transformations. First, w applies its export policies. If the route announcement is not filtered out, then a BGP-specific transformation occurs, which adds w to the **as_path** of the announcement, and filters out the announcement if the **as_path** contains u. Finally, the *import policies* of u are applied to the announcement. This is the point that a local preference value is assigned.

The BGP route selection procedure selects best route by first preferring highest values of local preference, then shortest AS paths, and so on. For a complete description of this process, see [2], [3], [1].

Informally, the process of transforming a set of BGP policies into an instance of SPP could proceed as follows. Designate any AS as the origin o. A distinct instance of the SPP would be constructed for each prefix p originated by an o. For each AS u we then enumerate every possible simple path in the AS graph from AS o to AS u. For each such path P, we start at o and compose the export policies, BGP transformation, and import policies, marching along the path in a hop-by-hop fashion toward u. If we do not make it to the end of the path (that is, the route was filtered out somewhere along the path), then the path is not in the set of permitted paths at u. Otherwise, the path P is in the set of permitted paths at u. For each permitted path P this procedure will produce a BGP route announcement $r(P)$. We then define λ^u to be any ranking function that the obeys the rules (1) $\lambda^u(P_1) < \lambda^u(P_2)$ if and only if the BGP route selection process will prefer $r(P_2)$ over $r(P_1)$, and (2) $\lambda^u(P_1) = \lambda^u(P_2)$ if and only if the BGP route selection process does not prefer one of $r(P_2)$ over $r(P_1)$ the other. In case (2), the details of the BGP selection process dictate that both $r(P_2)$ and $r(P_1)$ must come from the same next-hop router. Since a BGP speaking router only announces at most one route to any prefix (its best route), there is no possibility of a router simultaneously having two routes of the same rank. (Note that this accounts for the *strictness* condition in the definition of a ranking function for Stable Paths Problems, given in Section II-A.)

Observe that many distinct sets of BGP routing policies could translate to the same instance of SPP. Put another way, the set of SPP permitted paths at node u and the ranking of those paths via λ^u does not simply represent the routing policy of AS u, but rather the composition of all routing policies along paths from the origin to u.

B. Implementing Simple Routing

We now describe how the simple routing of [10] and Section II-B can be implemented in BGP.

The following table indicates whether or not an AS announces a route to its neighbor depending on its relationship to the AS that sent the route. A **Y** indicates that the route may be announced, while a **N** indicates that it may not be. For example, an announcement from a peer may be passed along to a customer but not to another peer.

		To		
		customer	peer	provider
From	customer	**Y**	**Y**	**Y**
	peer	**Y**	**N**	**N**
	provider	**Y**	**N**	**N**

These export rules ensure that no permitted path will have a step (**N** in peer-peer, peer-provider, and provider-peer entries) or a valley (**N** in provider-provider entry). In addition to these rules, each AS must ensure that all values of local preference assigned to customer announcements are greater than all local preference values assigned to peers and providers.

C. Safe Backup Routing with an Avoidance Level Attribute

The easiest way to describe inherently safe backup routing is to imagine that BGP is extended with a new attribute, the *avoidance level* of a route. The idea is that the higher this value, the less preferable a route. The avoidance level attribute must be considered before any other attribute in the BGP decision process. Other approaches that affect later stages of the decision process are not sufficient. One example is the commonly-used technique of AS prepending (inflating AS path length by repeating an AS number multiple times) that affects the decision process after the local preference attribute.

We can now extend the export rules to allow the three cases covered by Figure 5:

		To		
		customer	peer	provider
From	customer	**Y**	**Y**	**Y**
	peer	**Y**	**Y**	**Y**
	provider	**Y**	**Y**	**N**

In order to make sure that are avoidance classifier is step aware, we must impose rules concerning the use of the *avoidance level* attribute. For each **Y** of the export table, the following table indicates when the avoidance level attribute must be increased. The entry **O** indicates that its increase is optional, while a **R** indicates that its increase is required

		To		
		customer	peer	provider
From	customer	**O**	**O**	**O**
	peer	**O**	**R**	**R**
	provider	**O**	**R**	

Note that the optional entries in this table allow for very flexible backup routing and load balancing.

The more liberal export rules allow the possibility of forming a valley involving one or more steps. To avoid this, an AS must never send to a provider an upstream route that was received from a peer. An AS must mark routes that it receives from a provider before it sends them to a peer. An AS should never export such a marked route to a provider. A special community value could be employed between peers to signal that a route was received from an upstream provider.

Local preference must be assigned in a way that ensures that no customer route can get a value less than or equal to a peer or provider route. This ensures that customers are preferred within an avoidance level since the selection process will consider local preference only after the avoidance level. Together these simple rules ensure that the conditions of Theorem IV.1 are met.

D. Simulating Avoidance Levels with Communities Values

RFC 1998 [7] suggests using BGP communities as a way to influence a neighbor's routing policies. It applies this technique to the implementation of peer-peer backup. We extend this approach to simulate inherently safe backup routing described in Section V-C. Since we will use communities to carry several bits of information, it is perhaps best to think of them as extended community values [16].

We assume that each AS w has defined the following community values, and that the semantics of these labels is shared with w's neighbors.
$(w : bu : l)$ tag for backup route of avoidance level l,
$(w : up)$ tag for upstream routes (used between peers).
Routes that have not been tagged with an avoidance level community value are treated as being in avoidance level 0. We also assume that for any neighbor u, and any avoidance level l, that $lpa(w, u, l)$ represents the local preference assigned to routes from u received at w. We assume that for each w this function conforms to the following rules:

- If $u \in customers(w)$, $v \in peers(w) \cup providers(w)$, then for each level l we have $lpa(w, v, l) < lpa(w, u, l)$.
- For each l_1, l_2, u, v, if $l_1 < l_2$, then $lpa(w, u, l_2) < lpa(w, v, l_1)$.

That is, within each avoidance level customer routes are preferred over peer and provider routes, and routes of level lower avoidance levels are always preferred over routes of higher avoidance levels.

In practice, this scheme requires imposing some limit m on the number of avoidance levels. If an AS uses n values of local preference to differentiate between routes of the same avoidance level, then the function lpa must take on $m \times n$ distinct values. This complicates configuration of routing policies, particularly given the limitations of today's router configuration languages.

VI. 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 network is connected. In addition, conflicting local policies amongst a collection of ASes can cause the protocol to oscillate. This paper has has presented a model 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. It should be noted that we are ignoring the configuration of BGP within each AS, and so our recommendations will not exclude all types of BGP divergence. This follows from the fact that some types of internal BGP configurations can lead to BGP divergence within a single AS. For example, the interaction of BGP route reflectors, the processing of BGP routes using multi-exit discriminator values, and internal routing metrics, is not guaranteed to converge (see for example Scudder [17]). 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," RFC 1771 (BGP version 4), March 1995.
[2] Bassam Halabi, *Internet Routing Architectures*, Cisco Press, 1997.
[3] John W. Stewart, *BGP4: Inter-Domain Routing in the Internet*, Addison-Wesley, 1998.
[4] Geoff 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] C. Labovitz, R. Wattenhofer, S. Venkatachary, and A. Ahuja, "The impact of Internet policy and topology on delayed routing convergence," in *Proc. IEEE INFOCOM*, April 2001.
[7] E. Chen and T. Bates, "An application of the BGP community attribute in multi-home routing," Request for Comments 1998, August 1996.
[8] Kannan Varadhan, Ramesh Govindan, and Deborah Estrin, "Persistent route oscillations in inter-domain routing," Tech. Rep. 96-631, USC/ISI, February 1996.
[9] T. G. Griffin and G. Wilfong, "An analysis of BGP convergence properties," in *Proc. ACM SIGCOMM*, September 1999.
[10] Lixin Gao and Jennifer Rexford, "Stable Internet routing without global coordination," in *Proc. ACM SIGMETRICS*, June 2000.
[11] T. G. Griffin, F. B. Shepherd, and G. Wilfong, "Policy disputes in path-vector protocols," in *Proc. Inter. Conf. on Network Protocols*, November 1999.
[12] Timothy Griffin and Gordon Wilfong, "A safe path vector protocol," in *Proc. IEEE INFOCOM*, March 2000.
[13] T. G. Griffin, F. B. Shepherd, and G. Wilfong, "The stable paths problem and interdomain routing," Submitted for publication.
[14] Lixin Gao, "On inferring autonomous system relationships in the Internet," in *Proc. IEEE Global Internet Symposium*, November 2000.
[15] R. Chandra, P. Traina, and T. Li, "BGP communities attribute," RFC 1997, August 1996.
[16] Srihari Ramachandra and Daniel Tappan, "BGP extended communities attribute," Internet Draft draft-ramachandra-bgp-ext-communities-07.txt, Work in progress, expires June 2001.
[17] John Scudder, "BGP update," Talk at NANOG, October 24, 2000. See www.nanog.org.

VII. PROOFS

This appendix presents the proofs of all of results stated in the body of the paper. Some definitions are repeated here for ease of reading.

A. Dispute Wheels

We first introduce a sufficient condition that guarantees that an instance of the Stable Paths Problem is inherently safe. This definition is taken from [11], [12].

A *dispute wheel* W *of size* k is a triple $(\vec{U}, \vec{\mathcal{Q}}, \vec{\mathcal{R}})$, where $\vec{U}$ is a sequence of k nodes $u_0, u_1, \cdots, u_{k-1}$, $\vec{\mathcal{Q}}$ is a sequence of k non-empty paths $Q_0, Q_1, \cdots, Q_{k-1}$, and $\vec{\mathcal{R}}$ is a sequence of k non-empty paths $R_0, R_1, \cdots, R_{k-1}$. This triple is such that for each $0 \leq i \leq k-1$ we have

(1) R_i is a path from u_i to u_{i+1},
(2) Q_i and R_iQ_{i+1} are permitted at u_i,
(3) $\lambda^{u_i}(Q_i) \leq \lambda^{u_i}(R_iQ_{i+1})$.

All subscripts are to be interpreted modulo k.

See Figure 9 for an illustration of a dispute wheel. An SPP S is *wheel-free* if it does not contain a dispute wheel. It follows from the results of [11], [12] that if S is wheel-free, then it has a unique solution and is inherently safe.

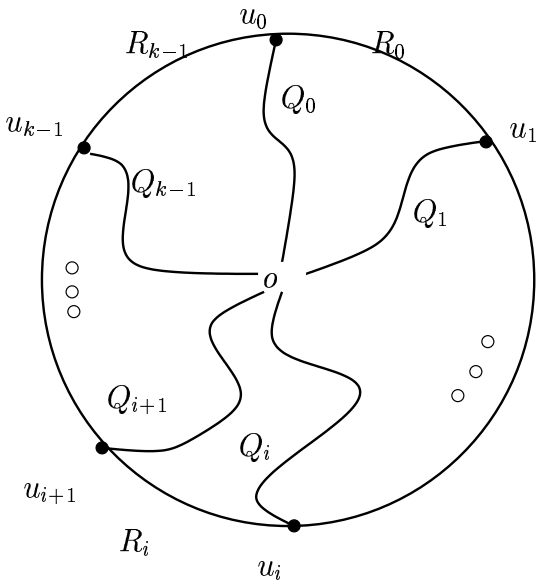


Fig. 9. A dispute wheel of size k.

B. Avoidance Classifiers

A function $\kappa(P)$ on permitted paths P is called an *avoidance classifier*, if the following conditions hold.
(a) If P and Q are paths permitted at node u and $\kappa(P) < \kappa(Q)$, then $\lambda(Q) < \lambda(P)$.
(b) If P, Q, and QP are permitted paths, then $\kappa(P) \leq \kappa(QP)$.

An *avoidance level* is any value in the range of κ. If l is an avoidance level and S is an SPP, A dispute wheel W is said to be *of level* l if for each i and j, $\kappa(Q_i) = \kappa(R_jQ_{j+1}) = l$.

Lemma VII.1: Suppose W is a dispute wheel for SPP S and κ is an avoidance classifier for S. Then there must exist an avoidance level l such that W is of level l.

Proof: Let W be a dispute wheel. For any i we have $\lambda^{u_i}(Q_i) \leq \lambda^{u_i}(R_iQ_{i+1})$. By property (a) of an avoidance classifier, this implies that $\kappa(R_iQ_{i+1}) \leq \kappa(Q_i)$. From property (b) we derive $\kappa(Q_{i+1}) \leq \kappa(R_iQ_{i+1})$. Together, these give us

$$\kappa(Q_{i+1}) \leq \kappa(R_iQ_{i+1}) \leq \kappa(Q_i).$$

for each i. By composing these equations we obtain (again, all subscripts are taken to be mod k),

$$\begin{array}{lllllll} \kappa(Q_i) & \leq & \kappa(R_{i-1}Q_i) & \leq & \kappa(Q_{i-1}) & \leq \\ & & \kappa(R_{i-2}Q_{i-1}) & \leq & \kappa(Q_{i-2}) & \leq \\ & & \cdots & \cdots & \cdots & \cdots \\ & & \kappa(R_{i+1}Q_{i+2}) & \leq & \kappa(Q_{i+1}) & \leq \\ & & \kappa(R_iQ_{i+1}) & \leq & \kappa(Q_i) & \end{array}$$

This proves the equalities claimed. ■

Corollary VII.2: Suppose S is an SPP and κ is an avoidance classifier for S. If for every avoidance level l there is no dispute wheel of level l, then S is wheel-free.

C. Backup Edges

Suppose $e = \{u, v\}$ is an edge in the SPP graph. The provider-customer digraph can impose an orientation on this edge. If the edge is oriented, we use the notation (u, v). We say that (u, v) is a customer-provider edge if $u \in customers(v)$ or a provider-customer edge if $u \in providers(v)$. If e is an edge

and eP is a permitted path, then e is called a *backup edge* if $\kappa(P) < \kappa(eP)$. Let $(w\,u\,v)P$ be a permitted path at w. If any of the following conditions hold, then edge $\{w,u\}$ is called a *mandatory backup edge*:
(a) (w,u) is a provider-customer edge and $\{u,v\}$ is a peer-peer edge,
(b) $\{w,u\}$ is a peer-peer edge and (u,v) is a customer-provider edge,
(c) $\{w,u\}$ and $\{u,v\}$ are peer-peer edges.
An avoidance classifier κ is *step aware* if it treats any mandatory backup edge as a backup edge. That is, $\kappa((u\,v)P) < \kappa((w\,u\,v)P)$ for each of the cases (a)-(c).

Lemma VII.3: Let S be an SPP with a step aware avoidance classifier κ. Suppose S has a dispute wheel W of avoidance level l. Then no R_i contains a mandatory backup edge.

Proof: Suppose that $R_i = P_1 e P_2$, where e is a mandatory backup edge. From the definition of an avoidance classifier we have $\kappa(eP_2Q_{i+1}) \leq \kappa(P_1eP_2Q_{i+1}) = \kappa(R_1Q_{i+1})$. Since κ is step aware, we have $\kappa(Q_{i+1}) < \kappa(eP_2Q_{i+1})$. Together these facts imply that $\kappa(Q_{i+1}) < \kappa(R_iQ_{i+1})$. This contradicts the assumption that W is of one avoidance level l. ■

Lemma VII.4: Let S be a valley-free SPP, κ a step aware avoidance classifier for S, and W is a dispute wheel for S.
(1) If there is some R_i in W where $R_i = (u_i\,v)P$ and (u_i,v) is a provider-customer edge, then every edge of R_i is a provider-customer edge.
(2) If there is some R_i in W where $R_i = P(v\,u_{i+1})$ and (v,u_{i+1}) is a customer-provider edge, then every edge of R_i is a customer-provider edge.

Proof: (1). Let e be any edge in $R_i = P_1eP_2$. Assume, without loss of generality, that all edges of P_1 are provider-customer edges. Suppose that e is a customer-provider edge. This is a contradiction, since this would mean that R_iQ_{i+1} contains a valley. On the other hand, suppose that e is a peer-peer edge. Let e' be the last edge in P_1 (a provider-customer edge). This is then a mandatory backup edge (condition (a)), which is prohibited by Lemma VII.3. Therefore, e can be only a provider-customer edge.

(2). Let $R_i = P(v\,u_{i+1})$ and e be any edge in $P = P'eP''$. Assume without loss of generality that all edges of P'' are customer-provider edges. Suppose that e is a provider-customer edge. This is a contradiction, since this would mean that the permitted path R_iQ_{i+1} contains a valley. On the other hand, suppose that e is a peer-peer edge. Then e is a mandatory backup edge (condition (b)), and is prohibited by Lemma VII.3. Therefore, e can be only a customer-provider edge. ■

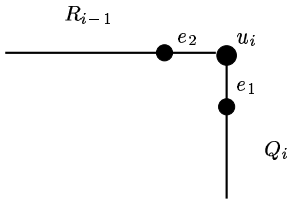
D. Preference for Customers

If κ is an avoidance classifier for SPP S, then ranking function λ is said to be *customer preferring with respect to* κ if within a set of paths of the same avoidance level customers paths are preferred, or put more formally, if for every P and Q permitted at node u with $\kappa(P) = \kappa(U)$, if P is a path to one of u's customers and Q is not, then $\lambda(Q) < \lambda(P)$.

Theorem VII.5: Let S be a valley-free SPP with an acyclic provider-customer digraph. Suppose that κ is a step aware avoidance classifier for S, and that S has a customer preferring ranking λ with respect to κ. Then S is inherently safe.

Proof: From Corollary VII.2 and the results of [11], [12] it is enough to show that for every avoidance level l there is no dispute wheel of level l.

Suppose $W = (\vec{U}, \vec{\mathcal{Q}}, \vec{\mathcal{R}})$ is a dispute wheel of level l. Pick any $u_i \in \vec{U}$. Let $Q_i = e_1Q$ and $R_{i-1} = Re_2$. We now perform a case analysis on all of the combinations of edges e_1 and e_2.



First, suppose that e_1 is a provider-customer edge. Since λ prefers customers, and $\lambda(Q_i) \leq \lambda(R_iQ_{i+1})$, it must be that R_i starts with a provider-customer edge. Lemma VII.4 (1) tells us that all the edges in R_i are provider-customer edges. We now consider the first edge of Q_{i+1}. This cannot be a customer-provider edge, since this would imply that R_iQ_{i+1} contains a valley. And this edge cannot be a peer-peer edge, since this would imply that the last edge of R_i is a mandatory backup edge (condition (a)), which is prohibited by Lemma VII.3. Therefore the first edge of Q_{i+1} must be a a provider-customer edge. We can now repeat this argument step-by-step clockwise around the dispute wheel and conclude that the entire rim is made up of provider-customer edges. This contradicts the assumption that the customer-provider digraph is acyclic. Therefore, we need not consider further the case where e_1 is a provider-customer edge. This leaves the following cases to consider.

Case 1: e_2 is a peer-peer edge. There are two subcases to consider.

Case 1.1: e_1 is a peer-peer edge. This means that e_2 is a mandatory backup edge (condition (c)), which is prohibited by Lemma VII.3.

Case 1.2: e_1 is a customer-provider edge. This means that e_2 is a mandatory backup edge (condition (b)), which is prohibited by Lemma VII.3.

Case 2: e_2 is a provider-customer edge. There are two subcases to consider.

Case 2.1: e_1 is a peer-peer edge. This means that e_2 is a mandatory backup edge (condition (a)), which is prohibited by Lemma VII.3.

Case 2.2: e_1 is a customer-provider edge. This contradicts the assumption that no permitted path contains a valley.

Case 3: e_2 is a customer-provider edge. Since this is the only remaining case, this must be true for the last edge in every path R_j in the dispute wheel. Then we know, by Lemma VII.4 (2), that all edges on the rim of the wheel are customer-provider edges. This contradicts the assumption that the customer-provider digraph is acyclic. ■