Integrity for Virtual Private Routed Networks

Randy Bush
Internet Initiative Japan (IIJ)
randy@psg.com

Timothy G. Griffin
AT&T Labs - Research
griffin@research.att.com.

***Abstract*— The term Virtual Private Network (VPN) encompasses a wide array of diverse technologies and network architectures. All VPNs should provide users with the isolation and security associated with private networks, but at lower costs made possible by implementing these networks over some type of shared infrastructure. *Provider provisioned* VPN allow enterprises to outsource their private backbone networks to service providers. For this reason, we will also refer to them as *Virtual Private Routed Networks* (VPRNs). This contrasts with other VPN technologies that require customers to manage their own point-to-point connections (leased lines or tunnels) and associated network administration.**

One type of VPRN currently being deployed is described in RFC 2547, which uses BGP to propagate routing information for all VPNs implemented within a provider's backbone, and a tunneling technology, such as MPLS, to isolate traffic. This technology requires fairly complex configurations within the backbone, and so poses challenges to providers supporting a large number of VPN customers.

We present the a formal analysis of several configuration and implementation concerns for VPRNs of the RFC 2547 variety. In particular, we focus on *integrity constraints* that must be maintained by providers in order to ensure that intra-VPRN connectivity is achieved, and that disjoint VPRNs are isolated from each other.

I. Introduction

The term *Virtual Private Network* (VPN) has a many possible interpretations (see for example [1], [2]). All VPNs should provide users with the isolation and security associated with private networks, but at lower costs made possible by implementing these networks over some type of shared infrastructure.

Most often VPN refers to virtual circuits at layer 2 (frame relay, ATM), or to various types of tunneling and encryption technologies at layer 3 (IPSec, firewalls) [3], [4]. As the pricing for internet connectivity goes far below that of classic Frame Relay and ATM circuits, there has been increasing market desire to replace leased circuits by the Internet as the transport for corporate and inter-corporate private networks. These "Internet VPNs" have been the subject of considerable work within the Internet Engineering Task Force. The major dimensions along which these Internet VPNs seem to differentiate are

- implementation using layer two (L2VPN) (example, using MPLS as in RFC 2547 [5]), or layer three (L3VPN), (example, using IPsec),
- providing virtual LAN (layer two healing) connectivity or normal IP transport only,
- providing encryption of payload or relying on some sort of network isolation,
- configuration set-up done by the enterprise or by the internet provider (also called "provider provisioned VPNs").

The promise of provider provisioned VPNs [6] is now attracting a great deal of industry attention. A provider provisioned VPN allows each site in a corporate or campus network to "plug into the cloud" for connectivity. That is, a provider provisioned VPN allows enterprises to outsource their private backbone networks to service providers. For this reason, we will also refer to them as *Virtual Private Routed Networks* (VPRNs). This contrasts with other VPN technologies that require customers to manage their own point-to-point connections (leased lines or tunnels) and associated network administration.

RFC 2547 [5] describes VPRNs implemented with BGP [7], [8], [9] and MPLS [10]. Many Internet Service Providers (ISPs) are already offering VPRN services based on this RFC. The full VPRN model promised by RFC 2547 includes (1) VPRNs that can overlap, thus implementing "intranets" and "extranets," (2) the ability for a site to selectively expose address ranges to the various VPRNs to which it is a member, and (3) the ability for VPRNs to span multiple providers. We believe it quite likely that, with the increasing emphasis being placed on security, VPNs will be extremely widely deployed, and that user sites will have very complex interrelationships. A single site will likely participate in many VPNs. For example, a site may connected to (1) the corporate intranet, (2) extranets of critical suppliers, (3) extranets of important customers, and (4) extranets of financial providers.

For providers, RFC 2547 adds complexity to the configuration and management of their networks. This complexity increases when customer sites participate in multiple VPNs, when customer sites contribute different address spaces to the VPNs to which they belong, and when the VPN backbone service is shared among multiple providers. Therefore we believe that formal models will help us better understand, and have increased confidence in, the correctness of VPN implementation and configuration. This is particularly important in the design of software for automating the process of VPN configuration.

This paper presents a formal analysis some aspects of the implementation of VPRNs in the style of RFC 2547. In particular, we are interested in the *integrity* of a set of VPRNs. For us, integrity means both the the VPRN specification is well-formed and that the implementation is correct. A VPRN specification is well-formed when it does not violate certain disjointness conditions that guarantee routing will not be ambiguous within any given VPRN (even when it overlaps

with other VPRNs). Implementation correctness defines when a collection of forwarding tables provides *full connectivity* within every VPRN, and *isolates* sites from VPRNs to which they do not belong.

RFC 2547 introduces many technical details, and vendors have provided many additional features to configure these VPNs (see for example [11], [12]). However, the approach of this paper is to model VPRNs in a manner that is independent of many low-level or vendor-specific implementation details. That is, we attempt to separate *what* is being implemented from *how* it is being implemented. In addition, this approach allows us to identify ambiguities in the definition of VPRNs and to explore alternative solutions to some of the problems raised.

Section II provides an informal overview of RFC 2547. Section III presents the basic definitions of our formal model, defines the fundamental integrity constraints, and explores some of the logical relationships between these constraints. Section IV shows how several implementation techniques can be used to enforce the integrity constraints. Section V provides a summary and a recommendation for improving scalability of backbone configurations. In particular, we recommend the use of source/destination forwarding tables for directing traffic in provider-edge routers. Section VI concludes with suggestions for future work in this area.

II. An overview of RFC 2547

The model of physical connectivity of RFC 2547 is illustrated in Figure 1. There are a number of *customer sites*, which are assumed to have internat connectivity that does not use the backbone. Each site has a number of *customer edge* devices, CEs, connected to *provider edge* devices, PEs. The backbone provides transit between PEs, possibly using internal provider routers, or P routers. There routers do not require any VPN related state. It is assumed that the backbone is provided by one or more network providers and that the sites are owned and managed by customers. The figure illustrates three overlapping VPNs. For example site 1 belongs to both VPN 1 and VPN 3.

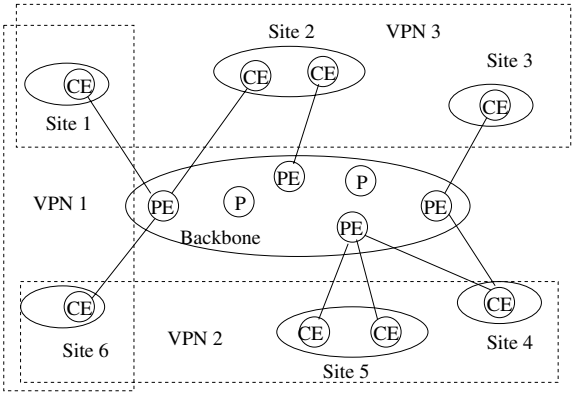


Fig. 1. Basic components of RFC 2547.

RFC 2547 states that constraints must be enforced to ensure that overlapping VPNs do no have overlapping address spaces:

> We assume that any two non-intersecting VPNs (i.e., VPNs with no sites in common) may have overlapping address spaces; the same address may be reused, for different systems, in different VPNs. As long as a given endsystem has an address which is unique within the scope of the VPNs that it belongs to, the endsystem itself does not need to know anything about VPNs.

A site can also selectively expose some of its address space to one of its VPNs, yet conceal it from another. As RFC 2547 states,

> While the basic unit of interconnection is the site, the architecture described herein allows a finer degree of granularity in the control of interconnectivity. For example, certain systems at a site may be members of an intranet as well as members of one or more extranets, while other systems at the same site may be restricted to being members of the intranet only.

Finally, the VPN implementation must ensure that there is intra-VRPN reachability and some notion of privacy. That is, the provider backbone must guarantee that distinct VPNs are *isolated*. RFC 2547 states this as

> Two sites have IP connectivity [...] only if there is some VPN which contains them both. Two sites which have no VPN in common have no connectivity over that backbone.

PE routes do not have a single forwarding table, but require several to implement connectivity and isolation. Figure 2 illustrates three sites that make up two VPNs, connected by the simplest possible backbone — one with a single PE. Each site shows a representative prefix from that site, p_1 from site s_1, p_2 from site s_2, and p_3 from site s_3.

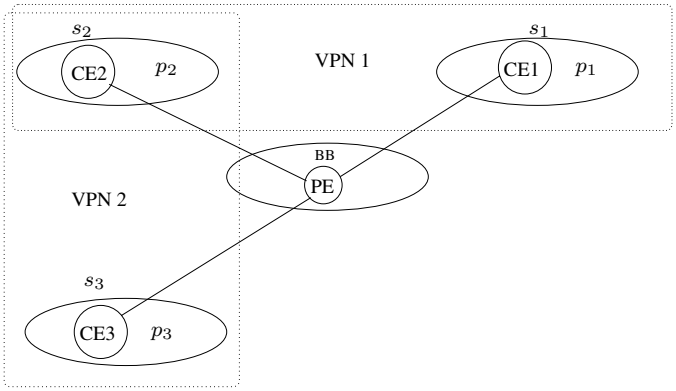


Fig. 2. Per-site forwarding example

A standard next-hop forwarding table at PE,

dest	nxt-hop
p_1	CE1
p_2	CE2
p_3	CE3

ensures connectivity. However, even isolation will be violated

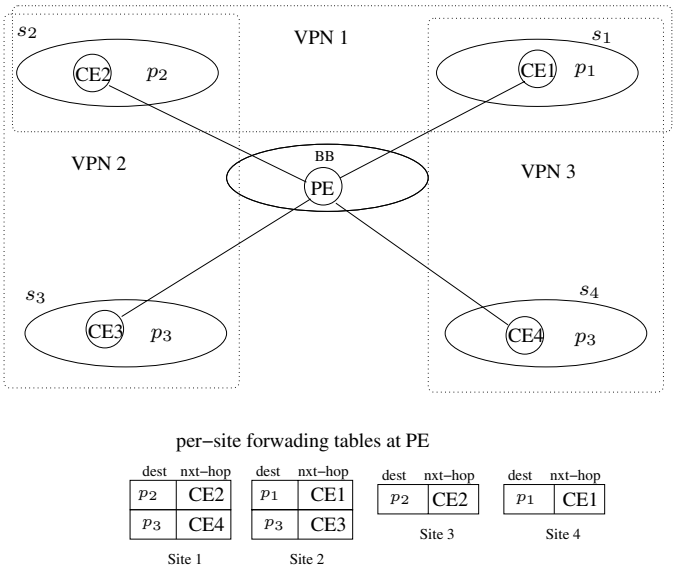


Fig. 3. Overlapping address space also requires per-site forwarding tables.

since sites s_1 and s_3 would have connectivity without belonging to a mutual VPN.

The solution described in RFC 2547 involves *per-site forwarding tables* at each PE. That is, each PE has a distinct forwarding table for each site to which it is connected. Traffic originating from Site i is forwarded using only the forwarding table associated with Site i. For example, the per-site forwarding tables for the PE in Figure 2 are

dest	nxt-hop
p_2	CE2

Site 1

dest	nxt-hop
p_1	CE1
p_3	CE3

Site 2

dest	nxt-hop
p_2	CE2

Site 3

For this example, this implementation ensures both connectivity and isolation. It is interesting to note that per-site forwarding tables are also required to support the fact that distinct VPNs can use the same address space. For example, Figure 3 presents a case where VPN v_3 contains site s_4 with destination p_3, which is the same as a destination in v_2 at site s_3. If the PE employed a vanilla forwarding table, it would not implement these VPNs correctly.

We briefly note the need for tunneling across the provider backbone. Figure 4 illustrates a situation in which PE1 is connected to sites 1 and 2 and PE2 is connected to sites 3 and 4. The connectivity between PEs is across the backbone, so VPN traffic between these two PEs must be *tunneled*. There several reasons for this :

- addresses from different VPNs can conflict,
- addresses of a VPN can conflict with addresses used in the backbone,
- even if the possibility of address conflict were somehow eliminated, tunneling would address *scalability* in that backbone routers/switches would not have to know all VPN address.

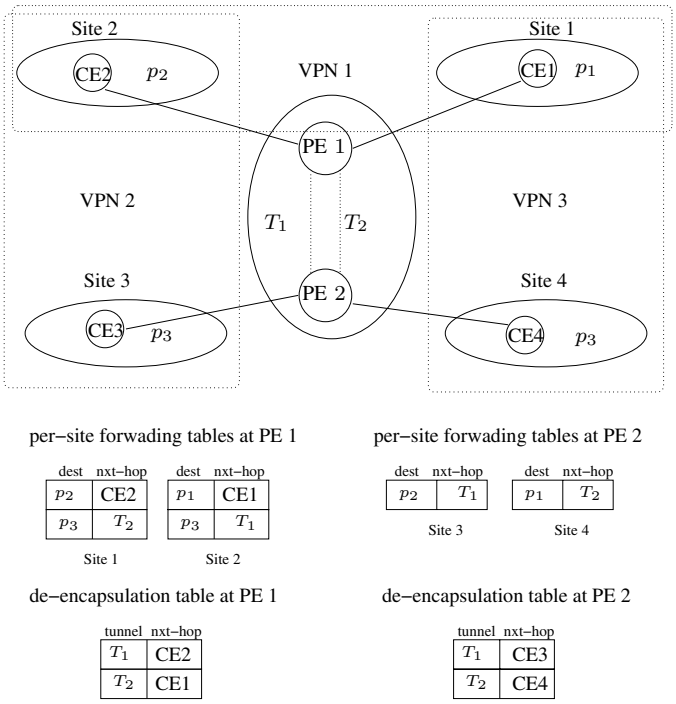


Fig. 4. Tunnels are required in the backbone.

In the example, (at least) two tunnels are needed to distinguish between VPN 2 and VPN 3. To see this, imagine only one tunnel existed between the PEs. Then PE 2 could not distinguish between traffic from PE 1 that is destined for p_3 at site 3 from that destined for p_3 at site 4. In general, tunnels terminate at PEs, but there are two distinct ways to decapsulate traffic. First, traffic could be directed to a particular per-site forwarding table. Second, traffic could be directed to a particular CE interface. The figure also shows the decapsulation mappings required at each PE, using the second type of mapping. Although tunnel provisioning and maintenance is an important component in VPNs, we do not consider it further in this paper.

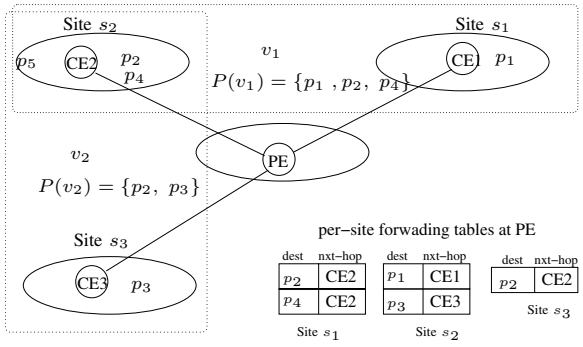


Fig. 5. Per-site forwarding tables and source address assurance cannot implement isolation.

Even with per-site forwarding tables, overlapping VPN that

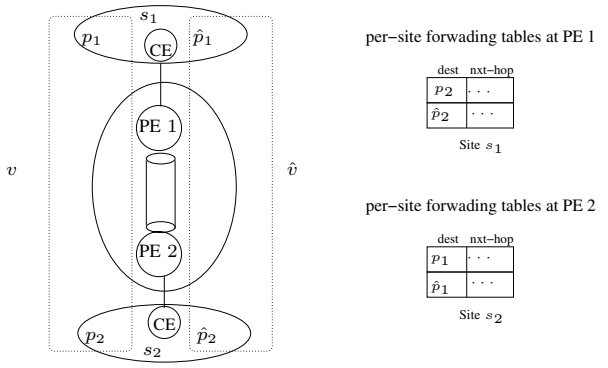


Fig. 6. Per-site forwarding tables alone cannot implement isolation.

can contribute different address spaces to different VPNs can cause violations of isolation. Figure 5 presents an example that is derived from a small modification to the VPNs presented earlier in Figure 2. Site s_2 has an additional prefixes p_4 and p_5. It contributes p_4 to VPN v_1 but not to VPN v_2, and does not contribute p_5 to either v_1 or v_2. Without some kind of address assurance at site s_2, this example violates isolation since p_5 can send traffic to both sites s_1 and s_3). But even if traffic with source address in p_5 can be prevented from sending traffic to site s_3, there is still another problem. Note that a system with source address in p_4 at site s_2 can send traffic to destinations in p_3 at site s_3, yet p_4 is not in the prefixes exposed to VPN v_2. This uni-directional traffic violates isolation.

Figure 6 presents an example that shows that using per site forwarding tables and source address assurance, that there can exist *bi-directional traffic* that violates isolation. Suppose that v and $\hat{v}$ are two select VPNs and that $s_1, s_2 \in S(v) \cap S(\hat{v})$. In addition, suppose that $E(s_1,\ v) = \{p_1\}$, $E(s_1,\ \hat{v}) = \{\hat{p}_1\}$, $E(s_2,\ v) = \{p_2\}$, and $E(s_2,\ \hat{v}) = \{\hat{p}_2\}$. Note that the per-site forwarding table for s_1 must contain entries for both p_2 and $\hat{p}_2$. Similarly, the per-site forwarding table for s_2 must contain entries for both p_1 and $\hat{p}_2$. This means that end systems in p_1 could have bi-directional communication with end systems in $\hat{p}_2$, which again violates isolation.

Many solutions to these problems are possible. Customers could have multiple interfaces to the backbone. Or per-VPN forwarding tables could be employed. Or perhaps a stronger form of source address assurance could be employed. Finally, some combination of these techniques might be used to avoid these problems. But it is clear that these issues are not straightforward, and some care has to be taken to make sure that isolation is assured. The following sections attempt to provide a rigorous basis for providing isolation guarantees.

III. Constraints for VPRNs

This section defines the basic components of and the backbone integrity constraints of our formal model of VPRNs. We then explore some of the logical relationships between these constraints.

A. Sites, interface classes, and addresses spaces

The set $S = \{s_1, \cdots, s_n\}$ represents a set of n *sites*, and $V = \{v_1, \cdots, v_k\}$ represents a set k VPRNs. For each site $s \in S$, there is a finite non-empty set C_s of *interface classes* for site s. An *abstract interface* is a pair $\langle s,\ c\rangle$, where s is a site and $c \in C_s$ is an interface class. Informally, abstract interfaces will be implemented with one or more logical interfaces at CEs, and will be linked to interfaces on PEs. Traffic from any logical interface of the same abstract interface will be treated the same way (filtered or forwarded) by PE routers. The set of abstract interface at site s is

$$I(s) = \{\langle s,\ c\rangle \mid c \in C_s\}.$$

A VPRN configuration must specify the VPRN membership of each each abstract interface $i = \langle s,\ c\rangle$. This set is denoted $V(i) \subseteq V$. That is, if $v \in V(\langle s,\ c\rangle)$, then the abstract interface $\langle s,\ c\rangle$ is a member of the VPRN v. Each VPRN $v \in V$ is associated with a set of abstract interfaces,

$$I(v) = \{\langle s,\ c\rangle \mid s \in S,\ c \in C_s,\ v \in V(\langle s,\ c\rangle)\},$$

and a set of sites

$$S(v) = \{s \in S\ \mid \langle s,\ c\rangle \in I(v)\}.$$

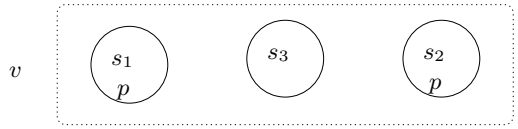
If $s \in S(v)$, then we say that site s is in VPRN v. Note that there must exist some abstract interface $\langle s,\ c\rangle \in I(v)$. For any site $s \in S$ the set of all VPRNs to which it belongs is denoted $V(s)$. That is, $V(s) = \{v \in V \mid s \in S(v)\}$.

For any IP prefix (CIDR block) p, we let $A(p)$ denotes the set of all addresses covered by p. If P is any nonempty set of prefixes, then $A(P)$ denotes the union of all sets $A(p)$, over $p \in P$. For each site s, we associate the set $P(s)$ of all prefixes that s can use to participate in in VPRNs, and the *address space of* s, denoted $A(s)$, is defined to be $A(P(s))$.

If v is a VPRN with site $s \in S(v)$, then s can selectively expose some prefixes in $P(s)$ to v while concealing others. If $i = \langle s,\ c\rangle \in I(v)$, the $E(i,\ v)$ denote the set of prefixes that site s exposes to VPRN v at abstract interface i. The corresponding address space is denoted $A(i,\ v)$. That is, if $a \in A(\langle s,\ c\rangle,\ v)$, then a is in the address space that site s exposes to VPRN v at abstract interface $i = \langle s,\ c\rangle$.

B. Connectivity Constraints

In formalizing the constraint that address spaces should not overlap there are actually two distinct cases that should be disallowed. The first is illustrated here,

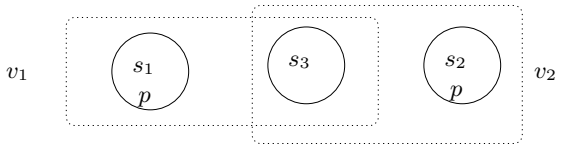


where VPRN v contains sites s_1, s_2, and s_3, and both sites s_1 and s_2 expose the prefix p to v. The problem is that there is ambiguity when s_3 attempts to exchange traffic to prefix p or when s_1 attempts to exchange traffic with prefix p in s_2. The same problems are encountered if site s_1 exposes p_1, s_2 exposes p_2, and p_1 is a subnet of of p_2.

S	=	set of sites
s	=	a site
V	=	set of VPRN names
v	=	a VPRN name
C_s	=	set of interface classes for site s
c	=	an interface class
$\langle s,\ c\rangle$ or i	=	an abstract interface
$V(i)$	=	names of all VPRNs to which i belongs
$I(s)$	=	set of all abstract interfaces at site s
$I(v)$	=	set of all abstract interfaces belonging to VPRN v
$S(v)$	=	set of all sites belonging to VPRN v
$V(s)$	=	names of all VPRNs to which site s belongs
$A(i,\ v)$	=	address space exposed to VPRN v at abstract interface i

TABLE I
SUMMARY OF NOTATION.

The second situation arises when s_1 is in VPRN v_1, s_2 is in VPRN v_2, and s_3 belongs to both VPRNs:



Again there is ambiguity when s_3 sends traffic to prefix p. Note that in this case it may be rather difficult to detect and correct since v_1 and v_2 may represent distinct administrative domains. And as with the first example, the same problems are encountered if site s_1 exposes p_1, s_2 exposes p_2, and p_1 is a subnet of of p_2. Indeed, if s_1 could announce a subnet of a prefix announced by s_3, then this raises security issues since it could be used as an attack by s_1 against v_2 (a VPRN that does not even include s_1 as a member).

The condition DISJOINT means that for all v, v_1, v_2, s_1, and s_2, if $s_1 \neq s_2$ $i_1 \in I(s_1)$ and $i_2 \in I(s_2)$, then the following two conditions hold:

(a) If $i_1, i_2 \in I(v)$, then $A(i_1,\ v) \cap A(i_2,\ v) = \phi$,
(b) If $v_1 \neq v_2$, $i_1 \in I(v_1)$, $i_2 \in I(v_2)$, and $A(i_1,\ v_1) \cap A(i_2,\ v_2) \neq \phi$, then $I(v_1) \cap I(v_2) = \phi$.

Note that when VPRNs share no sites they are completely unrestricted and are free to use overlapping address spaces.

C. Connectivity Constraints

We should ensure that an implementation provides intra-VPRN connectivity. Suppose that $i_1 = \langle s_1,\ c_1\rangle$ and $i_2 = \langle s_1,\ c_2\rangle$. We define $(a_1,\ i_1) \rightarrow (a_2,\ i_2)$ to mean that the underlying connectivity allows data traffic with source address a_1 and destination address a_2 to exit site s_1 via abstract interface i_1 and be delivered to the abstract interface i_2 at site s_2. If $(a_1,\ i_1) \rightarrow (a_2,\ i_2)$ and $(a_2,\ i_2) \rightarrow (a_1,\ i_1)$, then we write $(a_1,\ i_1) \leftrightarrow (a_2,\ i_2)$.

The constraint CONNECTIVITY is defined to mean that for all v, s_1, s_2, $i_1 \in I(s_1)$, $i_2 \in I(s_2)$, $a_1 \in A(i_1,\ v)$, and $a_2 \in A(i_2,\ v)$, if $i_1, i_2 \in S(v)$, then $(a_1,\ i_1) \leftrightarrow (a_2,\ i_2)$. Informally this constraint says that if two abstract interfaces share a common VPRN, then the underlying data connectivity allows them to exchange data traffic.

1) Isolation Constraints: The constraint CONNECTIVITY simply asserts that some underlying data connectivity is required. However, it does not disallow connectivity that should be prohibited. Therefore, we need additional constraints to enforce VPRN *isolation*. One possible formulation of isolation, which we call WEAK-ISOLATION, is as follows: for all i_1, i_2, a_1, and a_2, if $(a_1,\ i_1) \leftrightarrow (a_2,\ i_2)$, then there is some VPRN v such that $i_1,\ i_2 \in I(v)$, $a_1 \in A(i_1,\ v)$, and $a_2 \in A(i_2,\ v)$. In other words, if two abstract interfaces can exchange bidirectional traffic, then they must belong to a common VPRN, and the if two abstract interfaces must contribute the covering address spaces associated with the source and destination addresses of this traffic.

Note that weak isolation does not rule out unidirectional connectivity between abstract interfaces that do no share a VPRN. For this reason we introduce a stronger notion of isolation, call STRONG-ISOLATION: for all i_1, i_2, a_1, and a_2, if $(a_1,\ i_1) \rightarrow (a_2,\ i_2)$, then there is some VPRN v such that $i_1,\ i_2 \in I(v)$, $a_1 \in A(i_1,\ v)$, and $a_2 \in A(i_2,\ v)$. Note that STRONG-ISOLATION implies that WEAK-ISOLATION holds.

Ideally, we would like to achieve STRONG-ISOLATION with some combination of forwarding tables and traffic filtering. To study this, we identify the following constraints for source address assurance and for the scope of VPRN traffic.

By *source address assurance at site s* we mean that site s or the PEs connecting s to the backbone have some means of ensuring that any traffic originating from site s must have a "legal" source address. We will capture this in two constraints. First, *weak source address assurance*, denoted WSAA, is the constraint that for all i_1, i_2, a_1, and a_2, if $(a_1,\ i_1) \rightarrow (a_2,\ i_2)$, then there exists a v such that $i_1 \in I(v)$ and $a_1 \in A(i_1,\ v)$. That is, WSAA states that traffic leaving an abstract interface must have a source address associated in the space of some VPRN supported on that interface. Second, *strong source*

DISJOINT	=	all sites have unambiguous addressing
$(a_1,\ i_1) \rightarrow (a_2,\ i_2)$	=	traffic with source address a_1 and destination address a_2 can leave interface i_1 and arrive at interface i_2
CONNECTIVITY	=	intra VPRN connectivity holds
WEAK-ISOLATION	=	if two abstract interfaces can exchange bidirectional traffic, then they must belong to a common VPRN
STRONG-ISOLATION	=	if two abstract interfaces can exchange unidirectional traffic, then they must belong to a common VPRN
SCOPE	=	traffic leaving an abstract interface must have a destination address in the address space of some VPRN of the target abstract interface
WSAA	=	traffic leaving an abstract interface of a site must have a source address belonging to the space contributed by that site to some VPRN to which the interface belongs
SSAA	=	traffic leaving an abstract interface of a site must have a source address belonging to the space contributed by that site to some VPRN to which the interface belongs, and furthermore the destination interface must belong to the same VPRN

TABLE II

INFORMAL SUMMARY OF BACKBONE CONSTRAINTS.

address assurance, denoted SSAA, is the constraint that for all i_1, i_2, a_1, and a_2, if $(a_1,\ i_1) \rightarrow (a_2,\ i_2)$, then there exists a v such that $i_1,\ i_2 \in I(v)$ and $a_1 \in A(i_1,\ v)$. That is, SSAA states that traffic leaving an abstract interface must have a source address associated in the space of some VPRN supported on that interface and that the destination interface belongs to the same VPRN. It is easy to see that SSAA implies WSAA.

We define the constraint SCOPE to mean that for all i_1, i_2, a_1, and a_2, if $(a_1,\ i_1) \rightarrow (a_2,\ i_2)$, then there exists a v such that $i_1,\ i_2 \in I(v)$ and $a_2 \in A(i_2,\ v)$. This means that if there is unidirectional connectivity from abstract interface i_1 to abstract interface i_2, then the destination address of this traffic must be contained in the address space contributed by i_2's site to some VPRN to which both abstract interfaces belong.

Intuitively, source address assurance could be achieved by traffic management either at the customer site (for example, using firewalls) or at the PEs (for example, using source/destination forwarding tables), while SCOPE could be achieved by the use of forwarding tables.

D. Logical Constraints

A *simple VPRN* is one in which every interface belongs to a unique VPRN. The constraint SIMPLE means that for each i and for each v_1 and v_2 if $i \in I(v_2)$ and $i \in I(v_2)$, then $v_1 = v_2$.

A *union VPRN* is one in which every interface contributes the same address space to every VPRN to which it belongs. The constraint UNION means that for each i and for each v_1 and v_2 such that $i \in I(v_2)$ and $i \in I(v_2)$, we have $A(i,\ v_1) = A(i,\ v_2)$.

Theorem 3.1: If SIMPLE and SCOPE hold, then STRONG-ISOLATION holds.

Proof: Suppose SIMPLE and scope hold, and for some a_1, a_2, i_1 and i_2, we have that $(a_1,\ i_1) \leftrightarrow (a_2,\ i_2)$. In other words, we have (1) $(a_2,\ i_2) \rightarrow (a_1,\ i_1)$ and (2) $(a_1,\ i_1) \rightarrow (a_2,\ i_2)$. From (1) and SCOPE we know there exists a v_1 such that $i_1,\ i_2 \in I(v_1)$ and $a_1 \in A(i_1,\ v_1)$, and from (2) and SCOPE we know there exists a v_2 such that $i_1,\ i_2 \in I(v_2)$ and $a_2 \in A(i_2,\ v_2)$. Since SIMPLE holds, it must be the case that $v_1 = v_2$. Therefore STRONG-ISOLATION holds. ■

Theorem 3.2: If UNION and SCOPE hold, then WEAK-ISOLATION holds.

Proof: Suppose UNION and SCOPE hold, and for some a_1, a_2, i_1 and i_2, we have that $(a_1,\ i_1) \leftrightarrow (a_2,\ i_2)$. In other words, we have (1) $(a_2,\ i_2) \rightarrow (a_1,\ i_1)$ and (2) $(a_1,\ i_1) \rightarrow (a_2,\ i_2)$. From (1) and SCOPE we know there exists a v_1 such that $a_1 \in A(i_1,\ v_1)$, and from (2) and SCOPE we know there exists a v_2 such that $a_2 \in A(i_2,\ v_2)$. If $v_1 = v_2$, then we are done. Otherwise, take v_1 (or v_2) to be the witness for WEAK-ISOLATION. If v_1 is taken, then note that $a_2 \in A(i_2,\ v_2) = A(s_1,\ v_1)$, since UNION holds. Therefore WEAK-ISOLATION holds. ■

Theorem 3.3: If UNION, SCOPE, and WSAA hold, then STRONG-ISOLATION holds.

Proof: Suppose that UNION, SCOPE, and WSAA hold. Furthermore, suppose that for some a_1, a_2, s_1 and s_2 we have $(a_1,\ i_1) \rightarrow (a_2,\ i_2)$. By WSAA, there must exists a v_1 such that $i_1 \in I(v_1)$ and $a_1 \in A(i_1,\ v_1)$. By SCOPE, there must exists a v_2 such that $i_2 \in I(v_2)$ and $a_2 \in A(i_2,\ v_2)$. If $v_1 = v_2$, then we are done. Otherwise, take v_1 (or v_2) to be the witness for STRONG-ISOLATION. If v_1 is taken, then note that $a_2 \in A(i_2,\ v_2) = A(i_2,\ v_1)$, since UNION holds. Therefore STRONG-ISOLATION holds. ■

Of course the constraints STRONG-ISOLATION, SSAA, and

SCOPE are related. But how? The following easy lemma shows one direction.

Lemma 3.4: If STRONG-ISOLATION holds, then SSAA and SCOPE hold.

Proof: The proof is trivial. ■

However, this is not the direction we are most interested in. This would allow is to simply implement SSAA (perhaps with filters) and SCOPE (perhaps with forwarding tables) and have STRONG-ISOLATION follow automatically. That is we would like SSAA and SCOPE together to imply that STRONG-ISOLATION holds. Unfortunately, this is not always the case. We need another constraint to make this true, which we will call SYMMETRIC: for all i_1, i_2, a_1, and a_2, if $a_1 \in A(i_1,\ v_1)$, $a_2 \in A(i_2,\ v_2)$, and $i_1, i_2 \in I(v_1) \cap I(v_2)$, then $a_1 \in A(i_1,\ v_2)$ and $a_2 \in A(i_2,\ v_1)$. We will be the first to admit that this constraint is rather unintuitive, but it is what is required to make the following theorem work.

Theorem 3.5: Suppose that SYMMETRIC holds. Then if SSAA and SCOPE together hold, then STRONG-ISOLATION holds.

Proof: Assume that SYMMETRY, SSAA, and SCOPE hold. Now assume that STRONG-ISOLATION does not hold. Then we know that there exist i_1, i_2, a_1, and a_2 such that $(a_1,\ i_1) \rightarrow (a_2,\ i_2)$ and for all v and all such that $i_1,\ i_2 \in I(v)$ we have $a_1 \notin A(i_1,\ v)$ or $a_2 \notin A(i_2,\ v)$. We know by SSAS that there is some v_1 such that $i_1,\ i_1 \in I(v_1)$ and $a_1 \in A(i_1,\ v_1)$. Therefore, by the negation of STRONG-ISOLATION, we know that $a_2 \notin A(i_2, v_1)$. In a similar way, we know by SCOPE that there is some v_2 such that $i_1,\ i_1 \in I(v_2)$ and $a_2 \in A(i_2,\ v_2)$. Therefore, by the negation of STRONG-ISOLATION, we know that $a_1 \notin A(i_1, v_2)$. But together these two conditions violate SYMMETRIC, which is a contradiction. Therefore, STRONG-ISOLATION does hold. ■

IV. Implementation Invariants

In this section we explore which integrity constraints can be enforced with simple implementations techniques such as traffic filtering and forwarding tables.

A. Forwarding

Each interface i will have an *outbound* and *inbound* table of forwarding entries associated with it. An *entry* in an abstract destination-based forwarding is a tuple of the form $(p,\ i)$, where p is a prefix and i, the *next hop abstract interface*. In an actual implementation, if the destination address matches prefix p, then it will be forwarded to a logical interface associated with i (the exact interface will depend routing decisions). If the logical interface is not directly connected to the PE containing the entry $(p,\ i)$, then i can be taken to represent the head end of a tunnel that terminates at an inbound forwarding table for i.

Suppose $s \in S$, $i \in I(s)$, $v \in V$, and $i \in I(v)$. Then the *outbound forwarding table for v at i* is defined to be the set

$$F_v^O(i) = \{(p,\ i') \mid i' \notin I(s),\ p \in E(i',\ v),\ i' \in I(v)\}.$$

Note that if $v \notin V(i)$, then $F_v^O(i) = \phi$. The *inbound forwarding table for v at i* is defined to be the set

$$F_v^I(i) = \{(p,\ i) \mid p \in E(i,\ v)\}.$$

Note that if $v \notin V(i)$, then $F_v^I(i) = \phi$.

Again, this formalization abstracts away the issues of mapping logical interfaces implementing an abstract interface i to PEs, and of configuring tunnels between PEs. It is important to note that this forwarding scheme *does not* require any type of per-VPRN labeling of traffic, and requires no VPRN state on non-PE routers (P routers) in the backbone.

If $\hat{V} \subseteq V$ is a subset of VPRNs, then we define

$$F_{\hat{V}}^O(i) = \bigcup_{v \in \hat{V}} F_v^O(i)$$

In particular, we define $F_{V(i)}^O(i)$ to be the *per-interface forwarding table for i*. If a site has a single interface class, then this table is called the *per-site forwarding table for s*. On the other extreme, if each interface class at site s represents a distinct VPRN, then each table is called a *per-VPRN forwarding table for v*.

We first observe that the constraint DISJOINT ensures that there are no address clashes in forwarding tables.

Lemma 4.1: Suppose that DISJOINT holds. Then for any address prefix p there is at most one tuple $(p,\ i') \in F_{V(i)}(i)$.

Proof: Suppose $(p,\ i_1) \in F_{V(i)}(i)$ and $(p,\ i_2) \in F_{V(i)}(i)$, where i_2 is not associated with i_1's site. There are two cases to consider. In the first case, there is a v such that both $(p,\ i_1)$ and $(p,\ i_2)$ are in $F_v(i)$. This would violate condition (a) of DISJOINTNESS. In the second case, there are v_1 and v_2 such that $(p,\ i_1) \in F_{v_1}(i)$ and $(p,\ i_2) \in F_{v_2}(i)$. This would violate condition (b) of DISJOINTNESS. ■

Note that we are not implying that in an actual implementation violating disjointness that the forwarding tables would contain ambiguity. Often a routing protocol will install at most one entry. However, this may lead to unexpected loss of connectivity in some VPRNs.

Theorem 4.2: Suppose that DISJOINT holds. Suppose that traffic outbound from each interface i is forwarded using the per-interface forwarding table for i, $F_{V(i)}^O(i)$. Then CONNECTIVITY holds.

Proof: Suppose for some v, $i_1,\ i_2 \in I(v)$, $a_1 \in A(i_1,\ v)$, and $a_2 \in A(i_2,\ v)$. Then there exist prefixes $p_1 \in E(i_1,\ v)$ and $p_2 \in E(i_2,\ v)$ such that p_1 is the best match among $E(i_1,\ v)$ for address a_1 and p_2 is the best match among $E(i_2,\ v)$ for address a_2. By Lemma 4.1 we know that there is a unique tuple $(p_2,\ i_2) \in F_v(i_1)$ and a unique tuple $(p_1,\ i_1) \in F_v(i_2)$. Therefore $(a_1,\ i_1) \leftrightarrow (a_2,\ i_2)$, and so CONNECTIVITY holds. ■

One way of reading this result is that if the backbone is providing the correct connectivity and a customer does not have correct VPRN connectivity, then there must be some violation of either DISJOINTNESS or SITE-CONNECTIVITY.

Theorem 4.3: Suppose that traffic outbound from each interface i is forwarded using the per-interface forwarding table for i, $F_{V(i)}^O(i)$. Then SCOPE holds.

Proof: Assume that $(a_1,\ i_1) \rightarrow (a_2,\ i_2)$. Then, from the definition of $F^O_{V(i_1)}(i_1)$, there must be a best match $(p,\ i_2) \in F^O_{V(i_1)}(i_1)$ such that for some $v \in V(i_1)$, $p \in E(i_2,\ v)$ and $i_2 \in I(v)$. This means that $a_2 \in A(i_2,\ v)$. Therefore SCOPE holds. ■

B. Source/Destination Forwarding

An *extended forwarding table* for abstract interface i will contain entries of the form $(p_1,\ p_2,\ i')$ which will forward traffic to destination p_2 at abstract interface i', but *only* when the source of the traffic is within the range p_1, filtering out the traffic otherwise. Define the set

$$EF^O_v(i,\ p) = \{(p,\ p',\ i') \mid i' \notin I(s),\ p' \in E(i',\ v),\ i' \in I(v)\}.$$

Then the extended forwarding table can be defined as

$$EF_{V(i)}(i) = \bigcup_{v \in V(i)} \bigcup_{p \in E(i,\ v)} EF_v(i,\ p).$$

Note that this is much more restrictive than destination based forwarding together with source address assurance. A naive implementation of an extended source/destination forwarding table would require a "cross product" of entries. That is, the efficient implementation of such forwarding tables will present a challenge. However, much of the research on fast and efficient packet classification may be applicable in this context (see for example [13], [14]).

Theorem 4.4: Suppose that traffic outbound from each interface i is forwarded using the extended forwarding table for i, $EF^O_{V(i)}(i)$. Then STRONG-ISOLATION holds.

Proof: Assume that $(a_1,\ i_1) \rightarrow (a_2,\ i_2)$. Then, from the definition of $EF^O_{V(i_1)}(i_1)$, there must be a best match $(p_1,\ p_2,\ i_2) \in EF^O_{V(i_1)}(i_1)$ such that for some $v \in V(i_1)$, $p_1 \in E(i_1,\ v)$ and $p_2 \in E(i_2,\ v)$ and $i_2 \in I(v)$. This tells us that $a_1, \in A(i_1,\ v)$, and $a_2 \in A(i_2,\ v)$. Therefore STRONG-ISOLATION holds. ■

V. Summary and Recommendations

Table III provides a summary of the logical implications explicitly or implicitly proved in Section III. We note the proofs of isolation do not depend on the assumption of disjointness, and so these issues are in some sense orthogonal.

The results of the previous section on implementation can be combined with the logical implications to produce Table IV, which shows which isolations levels can be achieved with which implementation techniques. Here PS stands for implementation with per-site forwarding tables, PV for implementation with per-VPRN forwarding tables, and PS-SD for implementation with per-site source/destination forwarding tables (extended forwarding tables). As already noted, source address assurance could be achieved by traffic management either at the customer site (for example, using access lists or firewalls) or at a provider's PE. Since there are multiple low-level means of implementing this, we will not fully specify the implementation details. Instead, we will simply write WSAF for any filtering scheme that implements WSAA, and SSAF for any filtering scheme that implements SSAA. An X in means that there is an example that violates both weak and strong isolation.

Table IV indicates that there is a wide spectrum of choices available to providers of VPRNs. For example, enforcing STRONG-ISOLATION is easy for simple VPRNs and union VPRNs. However, no matter what choice is made, it is clear that VPN management cannot be completely outsourced to the providers. For example, customers must make sure that DISJOINT holds. In addition, customers must implement some type of per-site *strong-isolation*, which may or may not be easy depending on the complexity of the VPNs to which the site belongs. However, it is clear that from the provider's perspective, assuring *strong-isolation* would be greatly simplified if RFC 2547 insisted that VPNs be implemented with source/destination forwarding tables on all PEs.

VI. Remarks and Open Problems

We have explored some of the specification and implementation issues related to provider provisioned VPNs of the RFC-2547 variety. In particular, we have focused on the issues of maintaining connectivity and isolation in the context of intersecting VPNs with overlapping address spaces. We close by listing some related research problems that we think are are worth exploring. Scalability and robustness are the common and overriding concerns for all of these issues. The goal is to be able to support tens of thousands of VPN customers — some having complex interrelationships — over a shared infrastructure with low costs and high performance guarantees.

Verification of vendor implementations. Vendors implementing RFC 2547 each provide vendor-specific configuration commands and proprietary implementations. If remains to be tested which of the constraints (SCOPE, WSSA, SSAA, and so on), can actually be enforced with these implementations.

Routing configuration correctness. There is a distinction between forwarding tables and routing tables. Forwarding tables are low-level data structures, often implemented in special hardware, that simply direct traffic to the appropriate interface. On the other hand, routing tables are maintained by dynamic routing protocols, and generally contain much more information about the network state than can be found in a forwarding table. Routing protocols use their tables to maintain the forwarding tables in a state consistent with the state of the network and with the routing policies that have been configured. With a protocol such as BGP, these routing policies can can be quite complex.

With RFC 2547 VPNs, configuration of BGP is complicated by the use of *route distinguishers* that make overlapping addresses unique, extended community *route targets* that identify the VPNs to which a route belongs, and import and export route maps that use these extended communities to associate routes with specific forwarding tables (see for example [11], [12]).

A formalization of such BGP configurations may help in developing high level specification languages and compilers to generate low level vendor-specific commands. The designer of such a compiler could make use of the formalism presented

Implementation constraints

VPRN constraints	SCOPE	SCOPE + WSAA	SCOPE + SSAA
SIMPLE	STRONG-ISOLATION	STRONG-ISOLATION	STRONG-ISOLATION
UNION	WEAK-ISOLATION	STRONG-ISOLATION	STRONG-ISOLATION
SYMMETRIC			STRONG-ISOLATION

TABLE III
SUMMARY OF LOGICAL IMPLICATIONS.

Implementation

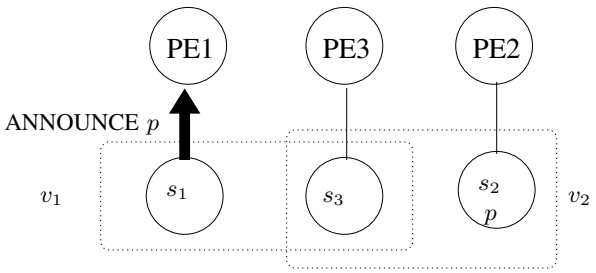
VPRN constraints	PS	PS + WSAF	PS + SSAF	PS-SD	PV
SIMPLE	strong	strong	strong	strong	strong
UNION	weak	strong	strong	strong	strong
SYMMETRIC	X	X	strong	strong	strong
any VPRN	X	X	X	strong	strong

TABLE IV
ISOLATION LEVELS ACHIEVED FOR VPRNS.

in this paper and attempt to prove that isolation invariants are maintained by the generated configurations.

Debugging routing problems. With provider provisioned VPNs, the customer and provider share the job of maintaining a network and debugging routing and connectivity problems. For example, the maintenance of disjointness is generally considered the customer's task (see for example [5], [15]). However, new debugging tools may be needed to coordinate debugging between customers and providers, and between multiple providers supporting the same VPN.

For example, consider a violation of condition (b) of disjointness that might arise when an administrator dynamically announces a new route. As illustrated here, site s_1 dynamically adds the prefix p to the set of prefixes it exposes to VPN v_1:



However, p is already a prefix of VPN v_2 (from site s_2), and v_2 shares site s_3 with VPRN v_1. Thus s_1's announcement potentially causes a violation of disjointness. Depending on the routing policies of the PE routers, this operation could either not result in the connectivity expected at s_1 (that is, PE3 may continue to route traffic from s_3 for p to s_2) or it could cut s_3's connectivity to p at s_2, replacing it with connectivity to p at s_1.

Of course this type of problem can occur whenever we have intranets and extranets, regardless of the VPN technologies used. What adds to the potential complexity here is that, in the worst case, PE1, PE2, and PE3 might each be managed by *a different ISP*. Furthermore, it may be that the prefix p does not appear in routing tables at site s_3, but only in the tables at PE3, which may or may not be visible to the customer at site s_3. In either case, these tables may very well not be visible to the operator at site s_1. Clearly, customers will require visibility into all forwarding tables that support their VPNs. In addition, service providers will have to share information concerning mutually supported VPNs.

A complementary approach might be to develop new protocols that aid in the negotiation and management of VPN address spaces (it does not seem possible to "piggy-back" a negotiation protocol on top of BGP, which is based purely on propagating announcements). In this example, perhaps s_1 or PE1 would signal an *intent* to announce prefix p to all other PE's supporting sites within VPRN v_1 (in this case only PE3), and only announce the prefix if the PE's reply with permission to do so.

SLA assurance. VPN customers may expect strict adherence to Service Level Agreements (SLAs) that commit a provider to maintaining specified levels of performance such as uptime, available bandwidth, limits on packet loss and delay (for more examples, see [15]). Designing and deploying networks with predicatable performance guarantees is a very difficult problem. VPNs make this even more difficult due to the potentially diverse requirements imposed by a large heterogeneous customer base. For example, the management of intra PE tunnels is greatly complicated by the existence of bandwidth SLAs. Some very interesting work has been done in this area for MPLS networks [16], [17], [18], [19]. However, it remains to be seen if these techniques can be efficiently implemented for RFC 2547 VPNs, and if they can be extended to VPNs that span multiple providers.

Dynamic robustness. Related to SLA assurance, and network robustness in general, is the stability of network devices and protocols during transient periods of stress. For example,

it has been shown that large oscillations in BGP table size can result in cascading failures in IPv4 networks [20]. This type of problem could, in principle, allow large instabilities in one RFC-2547 VPN to spill over into other VPNs supported on the same shared routing infrastructure. Research needs to be done on implementation techniques for isolating this kind of instability.

Convergence time. Path vector protocols, such as BGP, are inherently slower in adapting to routing changes than link state protocols. Delayed convergence of BGP in the Internet has been extensively studied [21], [22], and it has been shown that the rate of convergence is impacted by factors such as network topology, routing policies, route damping techniques (both the minimum route advertisement interval and route flap damping of RFC 2439 [23]). RFC-2547 VPNs will inherit some of these convergence delays due to the use of BGP. In particular, it is not clear how the BGP convergence in a single VPN will be impacted by the number of other VPN routes being supported over the same infrastructure.

Policy interaction. BGP policies can interact in unexpected and counter-intuitive ways to produce routing anomalies [24], [25], and these can even occur within a single autonomous system [26], [27], [28], [29]. We need to investigate if anomalous policy conflicts can arise — either within a single service provider or between providers — in BGP/MPLS VPNs of RFC 2547.

ACKNOWLEDGMENTS

Formalizing a rapidly developing technology such as RFC 2547 proved to be something of a challenge. We would like to thank Chris Chase, Ruomei Gao, Albert Greenberg, Zhuoqing Morley Mao, and Yakov Rekhter for provided many helpful comments and corrections concerning earlier drafts of this paper. This acknowledgment in no way implies that these reviewers agree with the conclusions of our analysis. The authors take full responsibility for any remaining errors or misinterpretations.

REFERENCES

[1] Paul Ferguson and Geoff Huston, "What is a vpn: Part I," *Internet Protocol Journal*, vol. 1, no. 1, June 1998.

[2] Paul Ferguson and Geoff Huston, "What is a vpn: Part II," *Internet Protocol Journal*, vol. 1, no. 2, September 1998.

[3] Dave Kosiur, *Building and Managing Virtual Private Networks*, John Wiley & Sons, Inc., 1998.

[4] C. Scott, P. Wolfe, and M. Erwin, *Virtual Private Networks*, O'Reilly, 1998.

[5] E. Rosen and Y. Rekhter, "BGP/MPLS VPNs," RFC 2547, 1999.

[6] ppvpn, "Ietf provider provisioned vpn working group," http://www.ietf.org/html.charters/ppvpn-charter.html.

[7] Y. Rekhter and T. Li, "A border gateway protocol," RFC 1771 (BGP version 4), 1995.

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

[9] J. W. Stewart, *BGP4, Inter-Domain Routing in The Internet*, Addison-Wesley, 1998.

[10] Eric W. Gray, *MPLS — Implementing the Technology*, Addison Wesley, 2001.

[11] Jim Guichard and Ivan Pepelnjak, *MPLS and VPN Architectures*, Cisco Press, 2000.

[12] Vivek Alwayn, *Advanced MPLS Design and Implementation*, Cisco Press, 2002.

[13] Priyank Warkhede, Subhash Suri, and George Varghese, "Fast packet classification for two-dimensional conflict-free filters," in *INFOCOM*, 2001.

[14] Florin Baboescu and George Varghese, "Scalable packet classification," in *SIGCOMM*, 2001.

[15] M. Carugi and D. McDysan (Eds), "Service requirements for layer 3 provider provisioned virtual private networks," IETF draft, Work In Progress, 2002.

[16] K.G.Ramakrishnan, D.Mitra, and J.A.Morrison, "VPN DESIGNER: A tool for design of multiservice virtual private networks," in *Proc. 8th International Telecom. Network Planning Symposium, NETWORKS*, 1998, Sorrento, Italy.

[17] D. Mitra and K.G.Ramakrishnan, "A case study of multiservice, multipriority traffic engineering design for data networks," in *Proc. IEEE GLOBECOM*, 1999.

[18] E.Bouillet, D.Mitra, and K.G.Ramakrishnan, "Design-assisted, real time, measurement-based network controls for management of service level agreements," in *EURANDOM Workshop on Stochastics of Integrated-Services Communications Networks*, 1999, Eindhoven, The Netherlands.

[19] E. Bouillet, D.Mitra, and K.G.Ramakrishnan, "The structure and management of service level agreements in networks," in *JSAC special issue on Recent Advances in Fundamentals of Network Management*, 2001.

[20] Di-Fa Chang, Ramesh Govindan, and John Heidemann, "An empirical study of router response to large bgp routing table load," in *Internet Measurement Workshop (IMW)*, 2002.

[21] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian, "Delayed Internet Routing Convergence," in *Proc. ACM SIGCOMM*, 2000.

[22] C. Labovitz, R. Wattenhofer, S. Venkatachary, and A. Ahuja, "The Impact of Internet Policy and Topology on Delayed Routing Convergence," in *Proc. IEEE INFOCOM*, 2001.

[23] C. Villamizar, R. Chandra, and R. Govindan, "BGP route flap damping," RFC 2439, 1998.

[24] Kanan Varadhan, Ramesh Govindan, and Deborah Estrin, "Persistent route oscillations in inter-domain routing," *Computer Networks*, vol. 32, pp. 1–16, 2000.

[25] Timothy G. Griffin, F. Bruce Shepherd, and Gordon Wilfong, "The stable paths problem and interdomain routing," *IEEE/ACM Transactions on Networking*, 2002.

[26] D. McPherson, V. Gill, D. Walton, and A. Retana, "BGP persistent route oscillation condition," IETF draft, Work In Progress, 2002.

[27] Cisco, "Endless BGP Convergence Problem in Cisco IOS Software Releases," Field Note, October 10 2001, `http://www.cisco.com/warp/public/770/fn12942.html`.

[28] Timothy G. Griffin and Gordon Wilfong, "On the correctness of ibgp configuration," in *Proc. ACM SIGCOMM*, 2002.

[29] Timothy G. Griffin and Gordon Wilfong, "Analysis of the MED oscillation problem in BGP," in *Proceedings of the International Conference on Network Protocols (ICNP)*, 2002.