

# Virtual Routing for Provider Edge Applications



**ip** infusion™  
white paper

## Introduction

Service Providers (SP) are continually adding new and more complex services for their most critical customers. One of the most cost-effective services that can be provided by an SP is a Virtual Private Network (VPN) service. By providing VPN services at the SP site rather than using hardware or software at the end user location, significant economies of scale can be leveraged to reduce the overall cost to the end user, while significantly adding revenue to the SP. In order to provision this VPN service in Provider Edge (PE) equipment, it would be necessary to setup a complex access control and route filtering mechanism to ensure that routing information did not propagate from one user to another when it was not desired. This is difficult, if not impossible, to manage.

There are also many complex SP users, i.e. enterprise customers, which need control of their routing configuration but still want to employ the SP to provide Internet access and other services, such as VPN. In order to provide this additional control, the SP needs to provision a separate router for this customer, which can get expensive and difficult to manage.

Virtual routing solves both of these problems by allowing a single router to be configured into multiple virtual routers (VR), with each VR instance acting like its own separate router, both for routing and configuration control. By using virtual routing, it is possible for service providers to provision VPN services and offer additional control to enterprise and advanced user using a single PE platform. This lowers costs for the SP and increases revenue opportunities. Coupled with additional services, such as traffic engineering, a complete PE solution can be provided to the SP.

This paper provides an introduction to virtual routing, discusses virtual routing requirements and design considerations, and provides a system level overview of the architecture and management of a virtual router implementation. Lastly, we will discuss several of markets that will drive the use of virtual routing.

## Defining Virtual Routing

Although there are a number of different definitions of virtual routing, we consider a virtual router to be an emulation of a physical router at the software layer. Essentially, many instances of router and protocol code may be running on a single unit, each operating independently from one another, for potentially different purposes. Figure 1 shows the typical situation when virtual routing is not deployed.

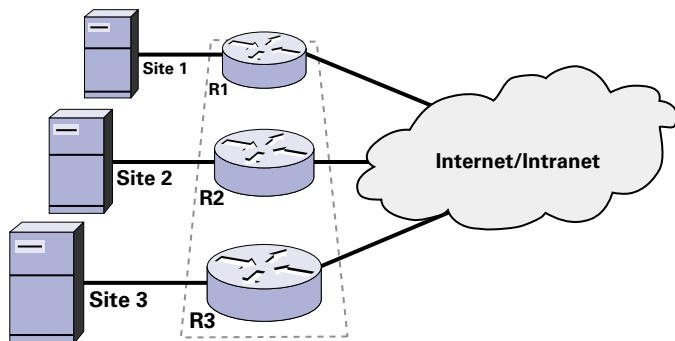


Figure 1 No Virtual Routing Deployed

In this case, each site must have a separate router installed to ensure the security, privacy, and manageability of the site. Compare this with the Figure 2 where virtual routing is deployed. In this case a single network unit can provide all of the functionality that is required for each of the sites, while still certifying the security and privacy of this customer. If the virtual routing implementation is scalable, then this approach can be used to support hundreds or thousands of customers by perhaps each using a different application.

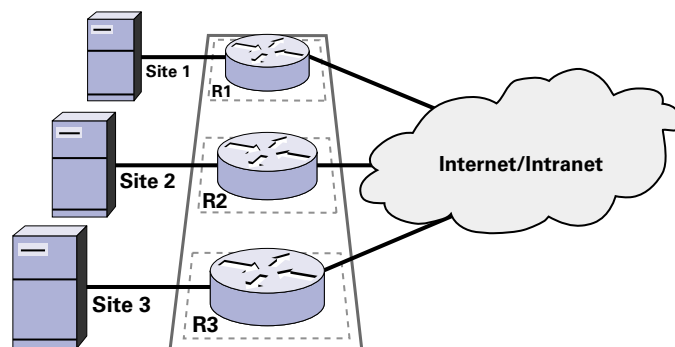


Figure 2 Virtual Routing Deployed

## Virtual Routing Requirements

In order to further define virtual routing, it is imperative that we define the requirements of a virtual router. This includes individual VR requirements, system-wide requirements, and management requirements. There are also security considerations of the VR system.

The following is a list of requirements that pertain to individual VRs:

- Each VR must be fully independent from any other VR.
- A VR must contain its own instance of the applicable routing protocols.
- A VR must contain its own Routing Information Base (RIB) for each supported technology (IPv4, IPv6, MPLS, etc...).
- A VR must be able to be managed by authorized administrators of that VR. This should be done using standard management protocols and utilities, which in fact should be extensions of existing utilities.
- A VR must be able to be managed by authorized global (system) administrators. This should also be done using standard management protocols and utilities.
- A VR must contain its own Forwarding Information Base (FIB) for each supported technology (this has implications for the TCP/IP stack, which will be discussed later).

Besides the individual VR requirements, a set of system-wide requirements must also be met by any VR implementation. These include:

- The FIB may exist as one physical entity, but must be kept logically separated such that each VR's data is kept separated.
- Software and hardware forwarding may be done either on a per VR basis (effectively within the VR) or globally. If done globally, then isolation of frames, routes, ARP, and other applicable data between VRs must be kept logically separate.

- There must be a way for authorized global (system) administrators to configure meta-aspects of the system. These include: creating/deleting new VRs, assigning physical interfaces to existing VRs, adding services (protocols) to existing VRs, and adding authorized users to VRs.

Equally as important as the individual and system level VR requirements are the management requirements for the VR implementation. There are two users that are defined in IP Infusion's virtual routing implementation, which are Global Administrators (GA) and VR Administrators (VRA). The GA can login by connecting to an interface (via telnet for example) that is allowed accept global context connections; to an interface that is not assigned to any VR; or via a console port. VRAs can login by connecting to any interface that belongs to their VR.

Some commands of the management system will only be available to the GA, and some will only be available to the VRA. However, most will be available to both. VRAs must not be allowed to execute any commands that are specific to the GA, whereas, GAs should be allowed to execute all commands. They may be required to login to a VR prior to doing so.

In addition, there are a number of security considerations that must not be compromised. These include the following:

- If a GA wishes to configure a VR, they must first jump to the context of correct instance. If not, a critical section could be compromised whereby both the GA and a VRA are configuring the same information.
- It must be possible for the GA to jump back to the global context once it has finished configuring a VR.
- It must not be possible to jump from one VR context to another. The GA must pass through the global context first.

- It must not be possible for a VRA to configure or view any information that is not owned by its VR. The VRA must not even be aware of the existence of resources beyond those assigned to its VR.
- It must be possible to connect to the global context using an IP address assigned to any VR. Furthermore, only users that have previously logged into the global context can move from a VR context to the global context.

## VPN Specific Virtual Router Requirements

There are VPN specific virtual router requirements that are mandatory:

- **Membership:** All VRs that are members of a specific VPN MUST share the same VPN-ID. The format is defined in RFC2685.
- **Scalability:** The backbone internal nodes (ISP 'P' nodes) must not be required to be VPN aware.
- **Routing:**
  - o **PE↔CE:** Any existing routing protocol could be used. Initially, IP Infusion focuses on BGP, static routing, or RIP.
  - o **Routing in the backbone:** This should not be affected by the VPNs.
    - **PE↔PE:** Any existing routing protocol can be used.
- **Security:** The architecture must accommodate different levels of data and routing security. The architecture must provide authentication and encryption services for VPNs where necessary.
- **Topology:** All existing VPN topologies must be supported.
- **Architecture:**
  - o Must accommodate different sizes of VPNs.
  - o Must support overlapping VPN address spaces in separate VPNs.

## Virtual Router Design Considerations

Design considerations that must be addressed in a VR system include assumptions and dependencies, general constraints, and an enumeration of the goals of the VR system.

Because the VR software depends on the TCP/IP stack, there are specific assumptions, which must be made in order for the VR software to function correctly. Firstly and most logically, the TCP/IP stack must support multiple FIBs. It is also important that the TCP/IP stack be able to provide APIs for assigning interfaces to FIBs and updating individual FIBs. When sending packets, the TCP/IP stack should allow applications to send them using the FIB that is appropriate for their VR context. When forwarding packets, the TCP/IP stack must also determine the correct FIB to use based on the received interface.

Because very few stacks support the multiple FIB capability, there are limitations that are operating system dependent. With Linux, multiple FIBs (up to 255) are supported, but each network interface must have a unique IP address. This limits the ability to support private IP addresses using RFC1597. There are TCP/IP stacks, which not only support multiple FIBs but also support conflicting IP addresses, i.e. using RFC1597 private addressing rules. One example is the Interpeak TCP/IP stack. This stack, together with IP Infusion's virtual routing implementation, can support thousands of FIBs on both VxWorks™ and OSE (and should be portable onto other real-time operating systems) and supports RFC1597 by allowing multiple TCP/IP stacks to run concurrently. This also provides higher performance to the total system.

The bottom line is that depending upon the operating system, some work may still be needed to fully support all of the features and principals outlined in this paper for virtual routing.

## General Virtual Routing Goals & Guidelines

There is a general set of goals that IP Infusion felt were important for our specific implementation of virtual routing. We believe that these apply to all systems supporting the virtual routing concept.

First, the implementation must be scalable up to hundreds or thousands of virtual routers. Threading design must be chosen carefully in order to avoid wasteful operations, such as unnecessary context switches. Bottlenecks must be avoided. For example, running one instance of the TCP/IP stack to handle 400 VRs is probably not feasible. Data structures must be chosen carefully. Speed of retrieval will be generally preferred to memory consumption.

Next, IP Infusion's design goal was to make minimal modifications to the current routing and route table management software. Our current product, the Advanced Routing Suite, was originally designed to support the MPLS-VPN solution (and IPv4/IPv6 routing). Some internal structures were leveraged for the VR project. Also, our product supported running multiple instances of the routing protocols, and this was leveraged to extend the product to support virtual routing.

Subsequently, it was very important that the "look and feel" of a virtual router be identical to the current product. We achieved this goal by extending commands rather than adding complete new ones whenever this was feasible. The only exception is the Global Management Authority (GMA), which is entirely new.

Lastly, platform portability was targeted as an essential goal of the VR project. In order to achieve this goal, we developed an API that sits between our Network Services Module (which provides route table management and other centralized services) and the IP stack. This will allow developers to quickly integrate virtual routing into a variety of operating system and TCP/IP stack environments.

## Virtual Routing Development Methodology

There were a number of enhancements to the Advanced Routing Suite that were undertaken to support virtual routing. First, as we have discussed, it was necessary to choose the TCP/IP stack first, as all other components depend on this stack. We chose the Linux TCP/IP stack and are currently working with Interpeak to support VxWorks™ and OSE. For Linux, we recommend that you refer to the IPRROUTE2 utility package from Alexey Kuznetsov (<ftp://ftp.inr.ac.ru/ip-routing/>).

In addition, modifications were made to our Network Services Module (NSM) to support multiple contexts/instances; one per VR. The design leveraged all existing virtual routing forwarding (VRF) capabilities from our MPLS-VPN solution. Each VR now has its own Route Information Base (RIB). Each RIB will be associated with a particular FIB in the forwarding plane.

For our Inter-Process Communication (IPC) we modified and added to the existing IPC facilities. The protocols now convey the virtual router instance to the central route management entity.

Our initial release of virtual routing supports OSPFv2 for IPv4. Modifications were made to the existing multi-instance feature. These changes will also be made to BGP and RIP in order to support virtual routing for these protocols in future releases of the product.

Perhaps the most important changes made to the existing Advanced Routing Suite product for the support of virtual routing were in the command line interface. IP Infusion developed components to handle all of the added management requirements for virtual routing. As was discussed earlier, a GMA was created to enable administrators to create and manipulate new VRs. The global manager can peer within and manipulate individual VRs. Furthermore, service provider customers can connect directly to and manage their own VR by telnetting to a particular interface on the "box."

To support a VPN-virtual routing environment, the VPN will be handled using a tunneling approach with both encryption and authentication support. The tunnel endpoints are logical interfaces and are treated exactly the same as physical interfaces.

## Virtual Router System Overview

A virtual router comprises a full implementation of a physical router at the software level. Figure 3 below, shows a single VR and illustrates the components that exist within a VR as well as the connections between components.

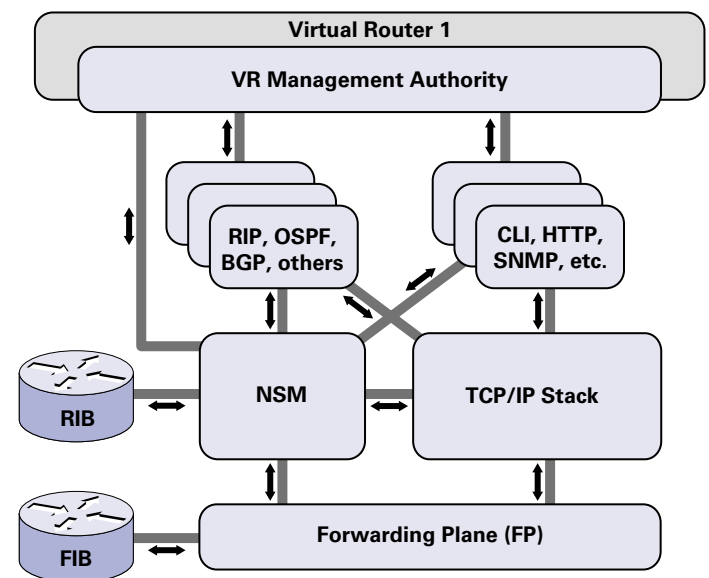


Figure 3

As Figure 3 shows, a virtual router contains all of the components you would expect to find on a physical router. The Management Authority (MA) maintains configuration and management information that pertains to the VR itself. It also controls access to this information and communicates with the box-level MA. There is a set of protocols that can be used with the MA to configure and retrieve information about the VR.

There is also a set of routing protocols, which is used for their normal functions of discovering neighbors, establishing relationships with neighbors, and learning routes to parts of the network to which they are not directly connected.

Additionally, there is a Route Table Manager (RTM), in this case the NSM daemon, that accepts route table updates from the routing protocols, prioritizes them, and then propagates the applicable information to different parts of the system.

Finally, there is a TCP/IP stack that connects the virtual router to the Forwarding Plane (FP). The TCP/IP stack includes many features of a VR now shown in this diagram, such as software forwarding, ARP processing, management of the FIBs, and ICMP functionality.

The entire system configuration with multiple VRs is shown in Figure 4 below. Aside from the individual VRs, there exists the GMA, which the box administrator will use to configure any system-wide information and to configure individual VRs.

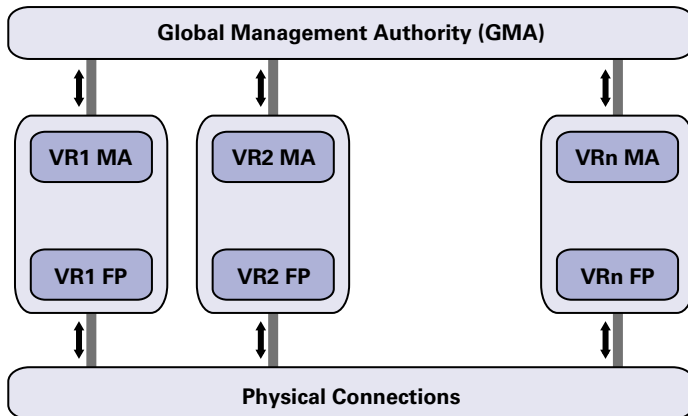


Figure 4

## Virtual Routing Markets

There are a number of applications for the use of virtual routing. In general, whenever it is desired to deliver secure communications from one point to another over the Internet/Intranet, virtual routing is applicable. VPN service using virtual routing is one such example. Figure 5 to the right shows an example setup of virtual routing with a service provider that provides VPN services for IP Infusion and another customer (the mysterious Customer X!). Note the prefix/interface assignments and their mapping and that an additional VR-C is used to establish OSPF-OSPF communication for the service provider.

In our implementation, it is also possible for multiple interfaces to correspond to a single VR implementation, which is ideal for Enterprise customers that need alternate routing at the access point.

There may be larger customers of a service provider that need to control or have their own routing table support. Virtual routing is a way for a service provider to allow the end customer to support its own routing configuration without having to have an entirely new router installed at the service provider site. A completely secure method is provided by IP Infusion's virtual routing implementation to ensure that their routing setup is completely separate and secured from the other interface routing configurations. This saves the service provider valuable capital, but still earns them the additional revenue from this customer.

Additional applications for virtual routing can be found in the broadband connectivity and multi-tenant unit markets. There are certainly other applications that will be discovered over the next few years.

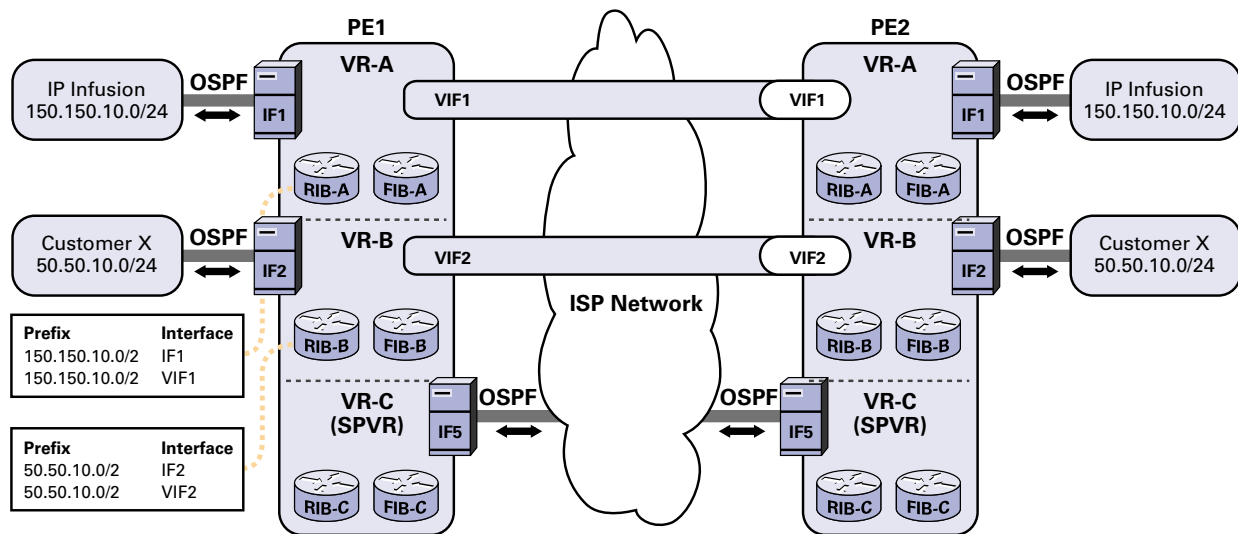


Figure 5

## Conclusion

This paper discusses the design and implementation of virtual routing, both from an architectural and system setup standpoint. We believe that our specific virtual routing implementation has achieved most, if not all, of the initial design goals. IP Infusion has initially implemented this for OSPF and intends to provide support for our entire line of routing protocols. We expect many novel applications of our virtual routing feature and welcome any feedback on its implementation and use by customers and service providers.



IP Infusion Inc.  
111 W. St. John Street  
Suite 910  
San Jose, CA 95113  
tel: 408-794-1500  
fax: 408-278-0521  
marketing@ipinfusion.com  
www.ipinfusion.com

© Copyright 2002 IP Infusion Inc. All Rights Reserved.  
IP Infusion, ipinfusion and ZebOS are trademarks of  
IP Infusion Inc. All other brands or product names are  
trademarks or registered trademarks of their respective  
holders. All specifications within this document are  
subject to change without notice.

Part No. 01920001