Microsoft

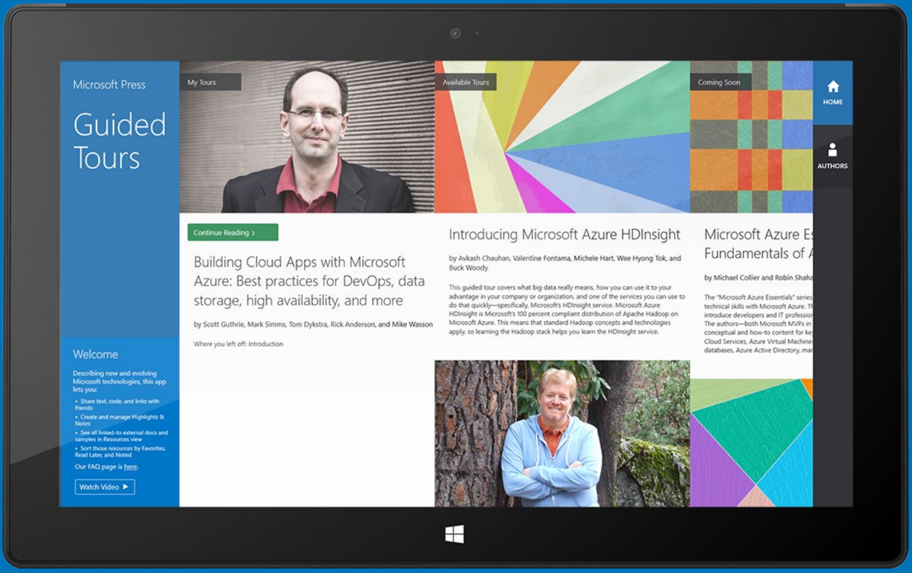
Microsoft System Center

Network Virtualization and Cloud Computing

Nader Benmessaoud ▪ CJ Williams ▪ Uma Mahesh Mudigonda
Mitch Tulloch, Series Editor

Hear about
it first.

Wait, there's more...



Welcome

Describing new and evolving Microsoft technologies, this app lets you:

- Share text, code, and links with friends
- Create and manage Highlights & Notes
- See all linked-to external docs and samples in Resources view
- Sort those resources by Favorites, Read Later, and Noted

Our FAQ page is here.

Building Cloud Apps with Microsoft Azure: Best practices for DevOps, data storage, high availability, and more

by Scott Guthrie, Mark Simms, Tom Dykstra, Rick Anderson, and Mike Wasson

Where you left off: Introduction

Introducing Microsoft Azure HDInsight

by Avkash Chauhan, Valentine Fontama, Michele Hart, Wee Hyong Tok, and Buck Woody

This guided tour covers what big data really means, how you can use it to your advantage in your company or organization, and one of the services you can use to do that quickly—specifically, Microsoft's HDInsight service. Microsoft Azure HDInsight is Microsoft's 100 percent compliant distribution of Apache Hadoop on Microsoft Azure. This means that standard Hadoop concepts and technologies apply, so learning the Hadoop stack helps you learn the HDInsight service.

Microsoft Azure Es
Fundamentals of A

by Michael Collier and Robin Shaha

The "Microsoft Azure Essentials" series
technical skills with Microsoft Azure. Th
introduce developers and IT profession
The authors—both Microsoft MVPs in
conceptual and how-to content for ke
Cloud Services, Azure Virtual Machines
databases, Azure Active Directory, ma

Find more great content and resources in the **Microsoft Press Guided Tours** app.

The Microsoft Press Guided Tours app provides insightful tours by Microsoft Press authors of new and evolving Microsoft technologies.

- Share text, code, illustrations, videos, and links with peers and friends
- Create and manage highlights and notes
- View resources and download code samples
- Tag resources as favorites or to read later
- Watch explanatory videos
- Copy complete code listings and scripts

Download from
Windows Store

PUBLISHED BY
Microsoft Press
A Division of Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399

Library of Congress Control Number: 2013952566
ISBN: 978-0-7356-8306-8

Printed and bound in the United States of America.

First Printing

Microsoft Press books are available through booksellers and distributors worldwide. If you need support related to this book, email Microsoft Press Book Support at mspinput@microsoft.com. Please tell us what you think of this book at http://www.microsoft.com/learning/booksurvey.

The example companies, organizations, products, domain names, email addresses, logos, people, places, and events depicted herein are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

This book expresses the author's views and opinions. The information contained in this book is provided without any express, statutory, or implied warranties. Neither the authors, Microsoft Corporation, nor its resellers, or distributors will be held liable for any damages caused or alleged to be caused either directly or indirectly by this book.

Acquisitions Editor: Anne Hamilton
Developmental Editor: Karen Szall
Editorial Production: Megan Smith-Creed
Copyeditor: Megan Smith-Creed
Cover Illustration: Twist Creative, Seattle

Contents

Introduction

As businesses move more toward cloud computing, one important factor for success is adopting multi-tenant software-defined networking (SDN) solutions in data centers. Hyper-V Network Virtualization (HNV) is a key enabler for a multi-tenant SDN solution and is essential for implementing a hybrid cloud environment where tenants can bring not only their own IPs, but their entire network topology since the virtualized networks are abstracted from the underlying fabric network. Network virtualization in general and Hyper-V Network Virtualization in particular are relatively new concepts. Unlike server virtualization, which is a mature, widely-understood technology, network virtualization still lacks this kind of broad familiarity.

This brief book identifies some key usage and deployment scenarios for cloud computing to provide some deep technical background on the Microsoft SDN solution, enabling IT professionals to quickly learn the internals of HNV, how it works from end to end, and where and how it should be used.

Acknowledgments

The authors would like to thank the following individuals for their assistance during our work on this title:

- Amit Kumar, Senior SDET, Windows Azure Networking
- Charley Wen, Program Manager, Windows Core Networking
- Luis Martinez Castillo, Senior SDET, Windows Core Networking
- Praveen Balasubramanian, Senior SDE, Windows Core Networking
- Ramandeep Singh Dhillon, Program Manager Windows Server Networking

Errata & book support

We’ve made every effort to ensure the accuracy of this content and its companion content. Any errors that have been reported since this book was published are listed at:

http://aka.ms/SCvirt/errata

If you find an error that is not already listed, you can report it to us through the same page.

If you need additional support, email Microsoft Press Book Support at *mspinput@microsoft.com*.

Please note that product support for Microsoft software is not offered through the addresses above.

We want to hear from you

At Microsoft Press, your satisfaction is our top priority, and your feedback our most valuable asset. Please tell us what you think of this book at:

http://aka.ms/tellpress

The survey is short, and we read every one of your comments and ideas. Thanks in advance for your input

Stay in touch

Let's keep the conversation going! We're on Twitter: *http://twitter.com/MicrosoftPress*.

CHAPTER 1

Hyper-V Network Virtualization internals

Network virtualization in general and Hyper-V Network Virtualization specifically are relatively new concepts. Unlike server virtualization, which is a mature technology that is widely understood, network virtualization lacks this same broad understanding. The first section of this chapter walks through key concepts in Hyper-V Network Virtualization and the benefits it provides. The later section of this chapter covers how to set up a basic virtual network and connects the key concepts to the implementation.

Overview

Server virtualization is a well-known concept by which many virtual servers can run on a single physical server with the appearance of running on a dedicated physical server. Typically, a hypervisor provides an abstraction of physical resources (CPU, memory, storage, and local networking) allowing for this illusion. The benefits of server virtualization are also well known and, among others, include:

- Isolation (performance and security) between virtual servers
- More efficient use of physical resources
- Easier movement of workloads across physical servers

Network virtualization, from a high level, has the same goals when it comes to the network fabric that connects virtual servers. Network virtualization should allow a virtual network, including all of its IP addresses, routes, network appliances, and so on, to appear to be running directly on the physical network. This allows the servers connected to that virtual network to continue to operate as if they were running directly on the physical network even as multiple virtual networks share the physical network. This concept of virtual networks allows the network to gain many of the same benefits that server virtualization provided to servers. Figure 1-1 shows conceptually how network virtualization and server virtualization are the same.

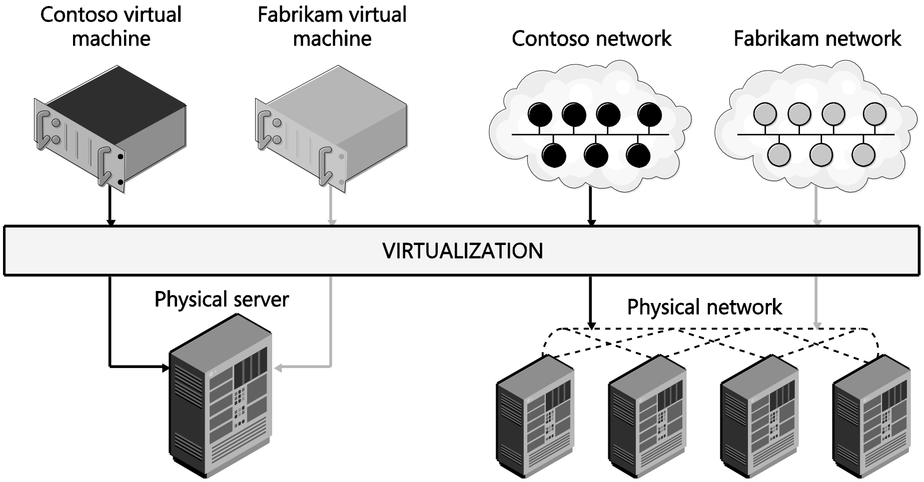


FIGURE 1-1 Network virtualization is conceptually the same as server virtualization.

In many ways, without network virtualization, the full range of benefits of server virtualization cannot be realized. Consider for example a virtualized SQL server, made possible by great strides in virtualizing high performance workloads. A virtualized SQL server should provide all the benefits of server virtualization, such as VM migration, but a physical network reduces the flexibility you actually get. This SQL server is assigned an IP address, which means that it has to stay in that IP address physical subnet. This limits any migration to only hosts that are attached to the same physical subnet (maybe only a rack or two out of a whole data center). Also, if the SQL server is on a VLAN, you must make sure that the VLAN has been properly configured across the physical network. With network virtualization you can decouple the network that the SQL server is attached to from the physical network and take full advantage of the potential of server virtualization. So without network virtualization, a key feature of server virtualization is much less flexible (i.e., you can move VMs only to hosts on the same physical subnet) and less automated (i.e., you might need to reconfigure the network before a VM can be migrated). This is just one such example of how network virtualization can allow you to gain the full potential of server virtualization.

Before diving into the details of how Hyper-V Network Virtualization works, consider the following summary of a few key benefits of network virtualization that help solve major problems you may face:

- The ability to run multiple virtual networks securely isolated from each other all with the illusion that they are each alone on the physical network.
- The ability to move VMs around in the physical network without having to reconfigure the physical network, including the IP address and VLANs.
- The ability to abstract the virtual network away from the underlying physical network.

Network virtualization provides value to three main groups: enterprises, workload owners, and service providers.

For enterprises, the biggest benefit of network virtualization is the ability to consolidate resources using a private cloud. For several years, enterprises have been implementing server virtualization to help consolidate workloads, but this approach has limitations. This is especially true when workloads expect a specific network topology, one that the private cloud's physical network can't accommodate. For enterprises that have grown through acquisitions and mergers, this can potentially be a major issue since each acquisition will have an existing IT infrastructure including network topologies that might have been in place for years. Network virtualization allows these existing network topologies to be decoupled from the underlying physical infrastructure so that even overlapping IP addresses can easily run on the same infrastructure. Also, enterprises can leverage the hybrid IT model where they only partially move their workloads to the cloud. Network virtualization helps reduce the pain of partially migrating resources to the cloud because the virtual network is not tied to the physical network.

For workload owners (whether on-premises, in a hosted environment, or in the cloud), the big benefit is that they do not have to change the configuration of the workload regardless of whether the workload needs to be moved around. Line of business applications in particular are sometimes designed to run with a particular network configuration, even with some components having well-defined IP addresses. As a result, to move an application to the cloud or to a service provider, a workload owner must either change the configuration of the application or figure out how the service provider can allow policies, VM settings, and IP addresses to be preserved. With network virtualization, this is no longer an issue because the workload owner can now move an application into the cloud while preserving all network settings, including IP addresses, even if they overlap with those belonging to another customer in the cloud or at the service provider.

For service providers, network virtualization provides some clear benefits. Most importantly, it allows them to offer their customers the ability to bring their own networks including any network settings (such as IP addresses, network topologies, and network services) that the customer wants to preserve. Network virtualization thus gives service providers a scalable, multi-tenant solution that provides them with flexibility concerning where they place workloads. For large service providers this is particularly important as they can now utilize their resources more efficiently and not have their resources usage dictated by customer requirements.

Network virtualization in some form has already been happening for some time, most prominently using VLANs. Virtualization using VLANs has recently run into issues, however, such as:

- **Scalability** Limit of 4,095 VLANs and specific switches and routers support only 1,000 VLANs.
- **Size** VLANs are limited to a single L2 network. This means that an individual L2

network must be very large (which has its own challenges) for a large number of VMs to participate in a specific VLAN. This is becoming even more of an issue because current data center trends are moving to smaller L2 domains (typically a rack or less).

- **Deployment** Often when VMs are migrated, the configuration of many switches and routers must be updated. In addition, VLAN configuration has to be coordinated with the Hyper-V hosts because the virtual switch must have matching VLAN configuration. Finally, where VMs can migrate is limited because they must stay in the same physical L2 domain to retain their existing IP address.

Due to these challenges, the industry has been moving to different models of virtual networks, including OpenFlow-based virtual networks and overlay networks. IBM, NEC, and Big Switch have commercially available OpenFlow-based virtual network solutions. Cisco's VXLAN based Network Virtualization, VMWare NSX Network Virtualization, and Microsoft's Hyper-V Network Virtualization are examples of the overlay network–based solution for network virtualization. The rest of this chapter will detail how Hyper-V Network Virtualization works.

Architecture and key concepts

Hyper-V Network Virtualization (HNV) provides a complete end-to-end solution for network virtualization that uses a network overlay technology paired with a control plane and gateway to complete the solution. These three pieces are embodied in the following:

- The Hyper-V virtual switch (with a virtual network adapter attached to a virtual network)
- Microsoft System Center 2012 Virtual Machine Manager (VMM) as the control plane
- The in-box HNV Gateway in Windows Server 2012 R2

At the core of HNV is a network overlay technology that allows separation between the virtual network and the underlying physical network. Network overlays are a well-known technique for layering a new network on top of an existing network. This is often done using a network tunnel. Typically, this tunnel is provided by packet encapsulation, essentially putting the packet for the virtual network inside a packet that the physical infrastructure can route (see Figure 1-2).

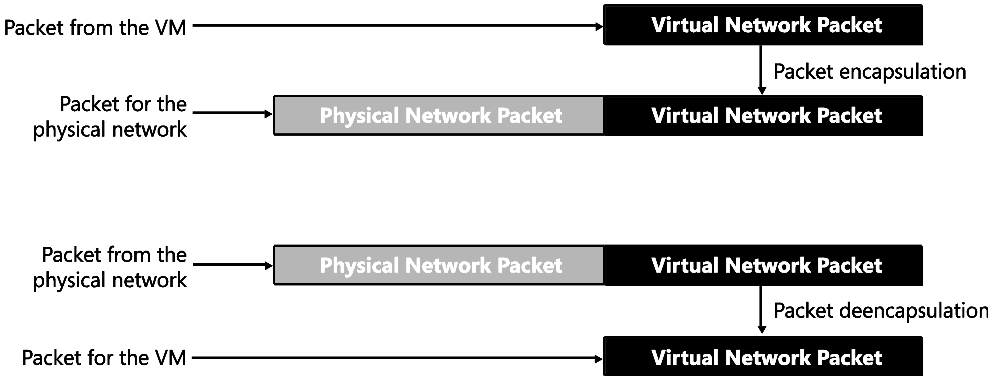


FIGURE 1-2 Network tunnel through packet encapsulation/de-encapsulation.

Network overlays are widely used for a number of scenarios, including VPN connections over wide area network (WAN) connections and Multiprotocol Label Switching (MPLS) connections over a variety of telecommunication networks. The endpoints in the overlay network have the intelligence needed to begin or terminate the tunnel by either encapsulating or de-encapsulating the packet. As mentioned earlier, the implementation of the overlay network is done as part of the Hyper-V virtual switch through the HNV filter, which encapsulates and de-encapsulates the packets as they are entering and exiting the virtual machines. This is discussed in detail in the "HNV architecture in the Hyper-V virtual switch" section.

In addition to an overlay network, HNV also provides a control plane that manages the overlay network independently from the physical network. There are two main types of control planes, centralized and distributed, each with its own strengths. For HNV, a centralized control plane is used to distribute policies to the endpoints needed to properly encapsulate and de-encapsulate the packets. This allows for a centralized policy with a global view of the virtual network while the actual encapsulation and de-encapsulation based on this policy happens at each end host. This makes for a very scalable solution since the policy updates are relatively infrequent while the actual encapsulation and de-encapsulation is very frequent (every packet). Windows provides PowerShell APIs to program the policies down to the Hyper-V virtual switch, which means anyone can build the central policy store. System Center 2012 Virtual Machine Manager implements the necessary functionality to be the central policy store and is the recommended solution, especially when System Center VMM is managing your virtual machines. (This text assumes that VMM is being used as the centralized policy store for HNV.)

Finally, because a virtual network that cannot communicate with the outside world is of little value, gateways are required to bridge the virtual network and either the physical network or other virtual networks. Windows Server 2012 R2 provides an in-box gateway and several third parties, including F5, Iron Networks, and Huawei, have gateways that can provide the bridge needed for virtual networks.

Figure 1-3 shows how the three pieces (VMM, the HNV Gateway, and the Hyper-V virtual switch) combine to provide a complete network virtualization solution. In this example the in-box Windows HNV Gateway provides VPN capabilities to connect customers over the Internet to data center resources being hosted at a service provider.

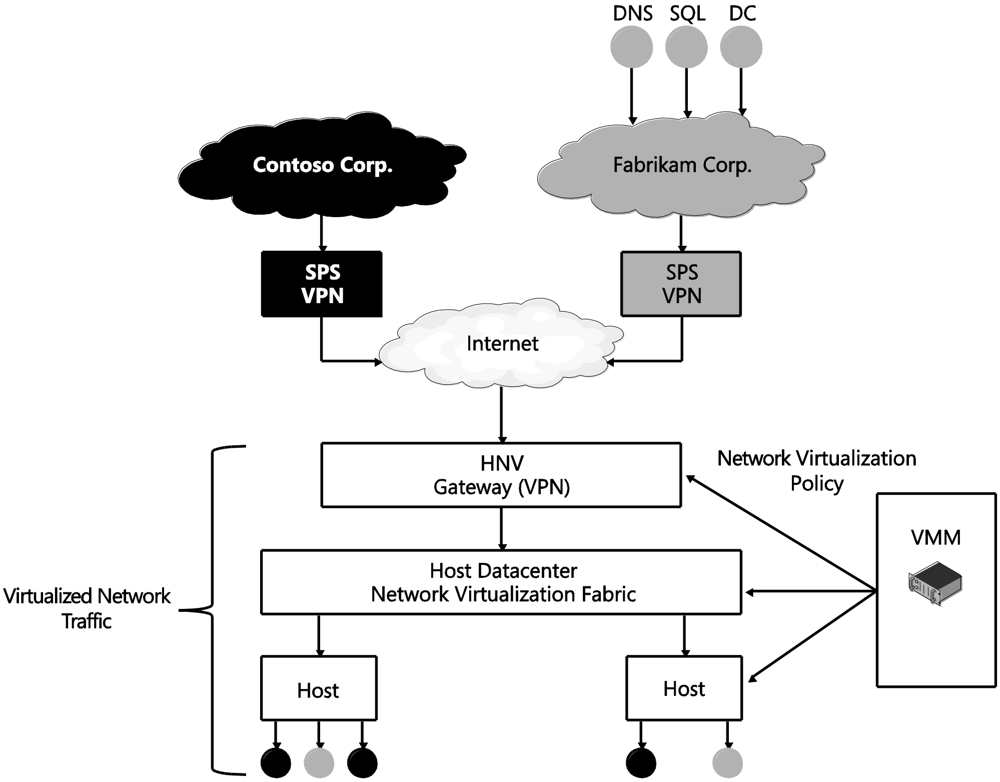

DNS
SQL
DC
Contoso Corp.
Fabrikam Corp.
SPS VPN
SPS VPN
Internet
HNV Gateway (VPN)
Network Virtualization Policy
VMM
Host Datacenter Network Virtualization Fabric
Virtualized Network Traffic
Host
Host

FIGURE 1-3 The Microsoft network virtualization solution.

Virtual machine network

The virtual machine network is a core concept in network virtualization. Much like a virtual server is a representation of a physical server including physical resources and operating system services, a virtual network is a representation of a physical network including IP, routing policies, and so on. Just like a physical network forms an isolation boundary where there needs to be explicit access to go outside the physical network, the virtual machine network also forms an isolation boundary for the virtual network.

In addition to being an isolation boundary, a VM network has most of the characteristics of a physical network, but several features are unique to VM networks:

- First, there can be many VM networks on a single physical network. This a major advantage for virtual networks, particularly in data centers that contain multiple tenants, such as what a service provider or cloud provider might have. These VM networks are isolated from each other even though their traffic is flowing across the same physical network and even in the same hosts. Specifically, the Hyper-V virtual switch is responsible for this isolation.

- Second, it is good to understand how IP and MAC addresses work in VM networks. There are two important cases. Within a single VM network, IP and MAC addresses cannot overlap, just like in a physical network. On the other hand, across multiple VM networks, each VM network can contain the same IP and MAC address, even when those VM networks are on the same physical network. Also, HNV supports both IPv4 and IPv6 addresses. Currently, HNV does not support a mixture of IPv4 and IPv6 customer addresses in a particular VM network. Each VM network must be configured to use either IPv6 or IPv4 for the customer addresses. On a single host there can be a mixture of IPv4 and IPv6 customer addresses if they are in different VM networks.
- Third, only VMs can be joined to a virtual network. Windows does allow the host operating system to run through the Hyper-V virtual switch and can be attached to a VM network but VMM, in System Center 2012 R2, won’t configure the host operating system to be attached to a virtual network.
- Fourth, currently a single instance of VMM manages a particular VM network. This limits the size of the VM network to the number of VMs supported by a single instance of VMM. In the R2 release, VMM allows a maximum of 8,000 VMs and 4,000 VM networks.

In VMM, the virtual machine network is called “VM network” and has a workflow that allows for the creation and deletion of VM networks and management of the properties associated with a VM network. In the HNV Windows PowerShell APIs, the VM network is identified by a Routing Domain ID (RDID) property. This RDID property must be unique within the physical network and set automatically by VMM.

Virtual subnet

Within a VM network, there must be at least one virtual subnet. The concept of a virtual subnet is identical to a subnet in a physical network in that it provides a broadcast domain and is a single LAN segment. In HNV, the virtual subnet is encoded in each virtualized packet in the Virtual Subnet ID (VSID) property and is a 24-bit field discoverable on the wire. Because of the close approximation to VLANs, valid VSIDs range from 4096 to 16,777,215 beginning after the valid VLAN range. The Virtual Subnet ID must also be unique within a particular physical network, typically defined as the network being managed by VMM.

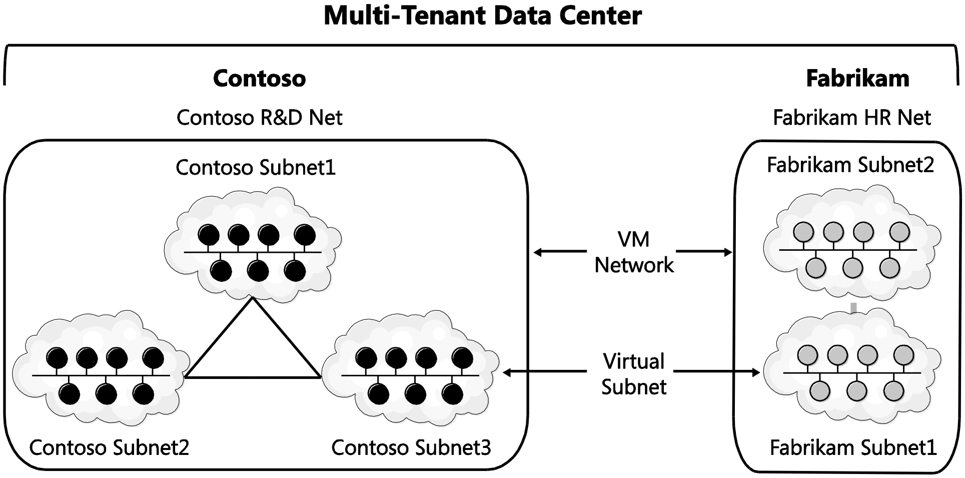


FIGURE 1-4 Example of how VM networks and virtual subnets are related.

To understand how VM networks and virtual subnets relate to each other, Figure 1-4 shows an example of multi-tenant data center network virtualization turned on. In this example, there are two tenants representing different companies, potentially competitors. They want their traffic to be securely isolated from each other so they form two VM networks. Inside each of these VM networks they are free to create one or more virtual subnets and attach VMs to particular subnets, creating the particular network topology that suits their needs.

VM network routing

After VM networks and virtual subnets, the next concept to understand is how routing is handled in VM networks, specifically, routing between virtual subnets and routing beyond the VM network. For more detail on how routing works and the packet flow related to routing in a VM network, see the section titled "Packet flows."

ROUTING BETWEEN VIRTUAL SUBNETS

In a physical network, a subnet is the L2 domain where machines (virtual and physical) can directly communicate with each other without having to be routed. In Windows Server, if you statically configure a network adapter, you must set a default gateway, which is the IP address to send all traffic that is going out of the particular subnet so that it can be routed appropriately. This is typically the router for the physical network. HNV uses a built-in router that is part of every host to form a distributed router for the virtual network. This means that every host, in particular the Hyper-V virtual switch, acts as the default gateway for all traffic that is going between virtual subnets that are part of the same VM network. In Windows Server 2012 and Windows Server 2012 R2, the address used as the default gateway is the ".1" address for the subnet. This .1 address is reserved in each virtual subnet for the default gateway and cannot be used by VMs in the virtual subnet.

HNV acting as a distributed router allows for a very efficient way for all traffic inside a VM network to be routed appropriately because each host can directly route the traffic to the appropriate host without needing an intermediary. This is particularly true when two VMs in the same VM network but different virtual subnets are on the same physical host. As you will see later in this section, when the packet flows are described with the distributed router the packet never has to leave the particular host.

ROUTING BEYOND A VM NETWORK

Sometimes a packet needs to go beyond the VM network. As explained earlier, the VM network is an isolation boundary, but that does not mean that no traffic should go outside of the VM network. In fact, you could easily argue that if there was no way to communicate outside the VM network then network virtualization wouldn't be of much use. So much like physical networks have a network edge that controls what traffic can come in and out, virtual networks also have a network edge in the form of an HNV gateway. The role of the HNV Gateway is to provide a bridge between a particular VM network and either the physical network or other VM networks.

An HNV gateway has several different capabilities, including:

- **Forwarding** Forwarding is the most basic function of the gateway and simply encapsulates or de-encapsulates packets between the VM network and the physical network the forwarding gateway is bridging to. This means that the IP address in the VM network must be routable on the physical network. This type of gateway would typically be used from a VM in a VM network to a shared resource like storage or a backup service that is on the physical network. Forwarding can also be used to connect a VM network to the edge of the physical network so that the VM network can use the same edge services (firewall, intrusion detection) as the physical network.
- **VPN** There are two types of VPNs:
    - **Site-to-Site** The Site-To-Site function of the gateway allows direct bridging between a VM network and a network (physical or another VM network) in a different data center. This is typically used in hybrid scenarios where a part of a tenant's data center's network is on-premises and part of the tenant's network is hosted virtually in the cloud. To use the Site-To-Site function, the VM network must be routable in the network at the other site and the other site's network must be routable in the VM network. Also, there must be a site-to-site gateway on each side of the connection (for example, one gateway on-premises in the enterprise and one gateway at the service provider).
    - **Remote Access (Point-to-Site)** The Remote Access function of the gateway allows a user on a single computer to bridge in the virtual network. This is similar to the Site-To-Site function but doesn't require a gateway on each side, only on one side. For example, with Remote Access an administrator can use a laptop to connect to the virtual network from the corporate network instead of an on-premises data center network.

- **NAT/Load Balancing** The final function that the gateway can provide is NAT/Load Balancing. As expected, NAT/Load Balancing allows connectivity to an external network like the Internet without having the internal virtual subnets and IP addresses of the VM network routable external to the VM network. The NAT capability allows for a single externally routable IP address for all connections external to the VM network or can provide a one-to-one mapping of a VM that needs to be accessed from the outside where the address internal to the virtual network is mapped to an address that is accessible from the physical network. Load Balancer provides the standard load balancing capabilities with the primary difference being that the virtual IP (VIP) is on the physical network while the dedicated IPs (DIPs) are in the VM network.

In Windows Server 2012 R2 the in-box gateway provides Forwarding, Site-to-Site, and NAT functionality. The gateway is designed to be run in a virtual machine and takes advantage of the host and guest clustering capabilities in Windows and Hyper-V to be highly available. A second major feature of the gateway is that a single gateway VM can be the gateway for multiple VM networks. This is enabled by the Windows networking stack becoming multi-tenant aware with the ability to compartmentalize multiple routing domains from each other. This allows multiple VM networks to terminate in the same gateway even if there are overlapping IP addresses.

In addition to the in-box HNV Gateway, there are a growing number of third-party gateways that provide one or more of these functions. These gateways integrate with VMM just as the in-box HNV Gateway does and acts as the bridge between the VM network and the physical network.

A few other requirements of VMM support of gateways should be noted:

- There can be only one gateway IP address per VM network.
- The gateway must be in its own virtual subnet.
- There can be multiple gateway VMs on the same host, but there cannot be other VMs on a VM network on the same host as the gateway VMs.

Packet encapsulation

Packet encapsulation is the core of network virtualization. In particular, in overlay networking technologies like HNV, packet encapsulation is the way in which the virtual network is separated from the physical network. Basically, in packet encapsulation, the packet for the virtual network is put inside (encapsulated) a packet that is understood by the physical network. Before the packet is delivered to the VM, the packet that is understood by the physical network is stripped off (de-encapsulated), leaving only the packet for virtual network. As mentioned previously, in HNV the VM switch provides the packet encapsulation functionality.

There are many different encapsulation formats, including recent ones like Virtual eXtensible Local Area Network (VXLAN), Stateless Transport Tunneling Protocol for Network Virtualization (STT), and Generic Routing Encapsulation (GRE). HNV uses a particular format of GRE, called Network Virtualization using Generic Routing Encapsulation (NVGRE), for the encapsulation protocol. GRE was chosen as the encapsulation protocol for HNV because it is an industry standard mechanism for packet encapsulation protocol. NVGRE is a specific format of GRE that is provided jointly by Microsoft, Arista, Intel, Dell, HP, Broadcom, Emulex, and Mellanox as an Internet draft at the IETF. A full version of the specification can be found at *http://tools.ietf.org/html/draft-sridharan-virtualization-nvgre-00*.

The NVGRE wire format has an outer header with source and destination MAC and IP addresses and an inner header with source and destination MAC and IP addresses. In addition there is the standard GRE header between the outer and inner headers. In the GRE header, the Key field is a 24-bit field where the virtual subnet ID (VSID) is put in the packet. As mentioned previously, this allows the VSID to be explicitly set in each packet going across the virtual network. To get hands on with the NVGRE packet format you can set up a simple HNV network (see the section titled "Hyper-V Network Virtualization: Simple setup") and use Message Analyzer to decode the packets and see NVGRE packets on the wire.

Customer Address (CA)

When looking at the NVGRE format it is important to understand where the address space for the inner packet comes from. It is called the Customer Address (CA). The CA is the IP address of a network adapter that is attached to the VM network. This address is only routable in the VM network and does not necessarily route anywhere else. In VMM, this CA comes from the IP pool assigned to a particular virtual subnet in a VM network.

Provider Address (PA)

The outer packet is similar in that the IP address is called the Provider Address (PA). The PA must be routable on the physical network but should not be the IP address of the physical network adapter or a network team. In VMM, the PA comes from the IP pool of the logical network.

Figure 1-5 shows how NVGRE, CAs, and PAs relate to each other and the VMs on the VM networks.

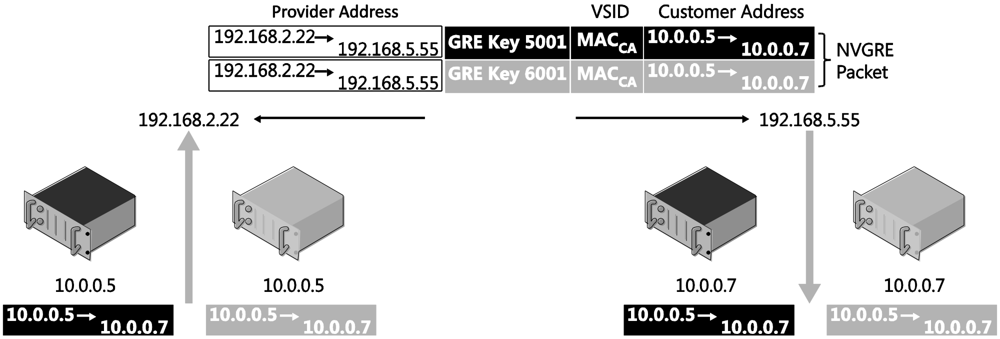


FIGURE 1-5 NVRGE, CA, and PA.

Hyper-V virtual switch

The Hyper-V virtual switch is the component that provides the network virtualization features on the end hosts. Specifically it provides all the capabilities pertaining to NVGRE encapsulation/de-encapsulation, policy enforcement (i.e., ensuring VMs on different VM networks can't communicate with each other), routing of packets between virtual subnets in the same VM network, and managing the local host's network virtualization policy as configured by VMM.

A design change between Windows Server 2012 and Windows Server 2012 R2 allows more compatibility between HNV and Hyper-V virtual switch extensions. In Windows Server 2012, HNV was an NDIS LWF, which meant that Hyper-V virtual switch extensions worked only on the customer address space. This also meant that capture and filter extensions were not aware of the underlying physical networking being used for HNV packets and that forwarding switch extensions could not co-exist with HNV, so customers had to choose a forwarding switch using HNV or a particular forwarding extension. Windows Server 2012 R2 introduced the ability for switch extensions to work on both the original customer address packet and the encapsulated provider address packet (see Figure 1-6). In addition, forwarding switch extensions can co-exist with HNV, allowing multiple network virtualization solutions (one provided by HNV and another provided by the forwarding switch extension) to co-exist on the same Hyper-V host.

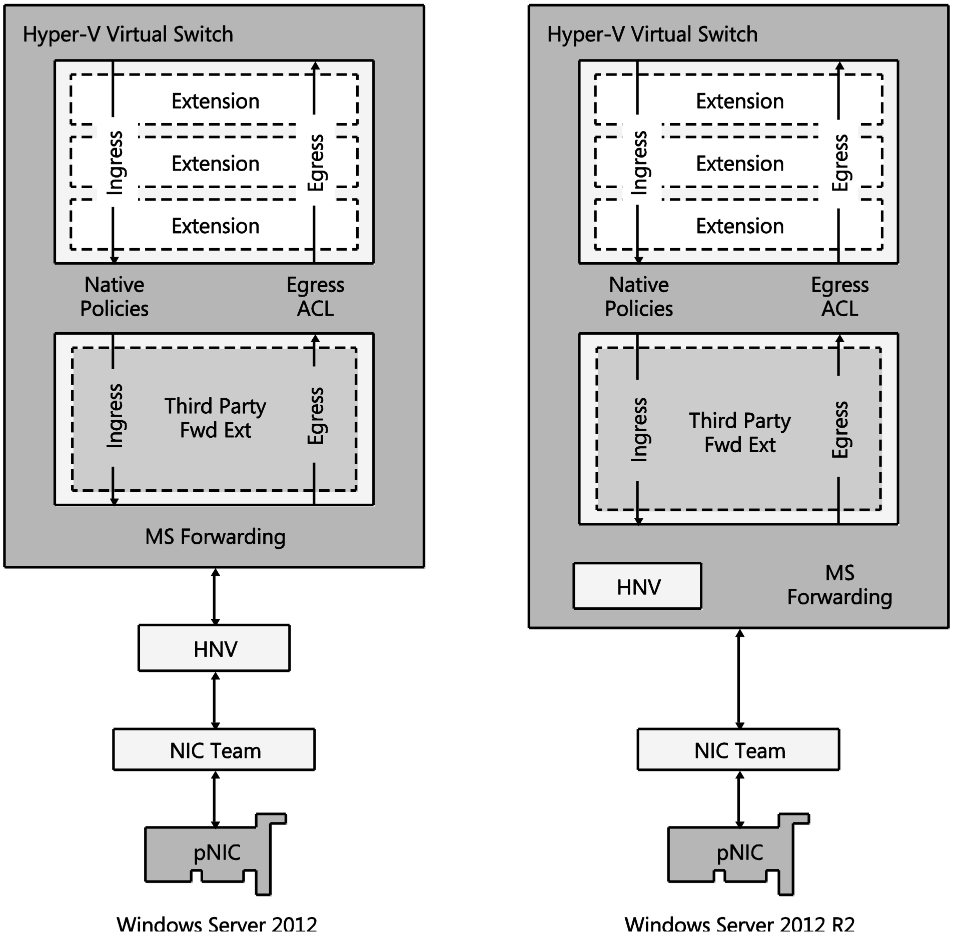


FIGURE 1-6 HNV architectural update in Windows Server 2012 R2.

Improved interoperability with switch extensions was the primary reason for the change, but a nice side effect is that the HNV NDIS LWF does not have to be bound to network adapters anymore. After you attach a network adapter to the virtual switch you can enable HNV simply by assigning a virtual subnet ID to a particular virtual network adapter. For those using VMM to manage VM networks this is transparent, but for anyone using PowerShell this will save an often-missed step.

Control plane

The control plane is comprised of two major pieces: HNV policy records and the central policy store. The control plane can be characterized from a high level as a centralized control plane that uses policy records to drive a distributed router on each host.

Policy records

Policy records drive the distributed router running on each host. The best way to understand the policy records is to go through the PowerShell APIs used to set the policy records. There are four APIs to look at. Each API has a New, Get, Set, and Remove command, but for this review, the New command is most interesting.

New-NetVirtualizationCustomerRoute

The New-NetVirtualizationCustomerRoute cmdlet creates a virtual network route in a VM network. HNV uses customer routes to manage network traffic on a virtual network.

To create a VM network route, specify the following values:

- **DestinationPrefix** A range of IP addresses as an IP prefix.
- **NextHop** A next hop gateway for the specified destination addresses.
- **RoutingDomainID** An ID for a virtual network that can include multiple virtual subnets.
- **VirtualSubnetID** An ID for a virtual subnet.

The full command line looks like this:

```
New-NetVirtualizationCustomerRoute
   -DestinationPrefix <String>
   -NextHop <String>
   -RoutingDomainID <String>
   -VirtualSubnetID <UInt32>
```

New-NetVirtualizationLookupRecord

The New-NetVirtualizationLookupRecord cmdlet creates a lookup record policy entry for an IP address that belongs to a VM network. Computers can exchange network traffic with a virtual machine by using a customer address within the virtual network. Network Virtualization manages the provider addresses that are the physical network addresses. This cmdlet creates a record that maps a customer address to a provider address.

To create a lookup record, specify the following values:

- **CustomerAddress** Specifies the IP address for a VM. You can use either an IPv4 or IPv6 address.
- **MACAddress** Specifies a MAC address that corresponds to the customer address.
- **ProviderAddress** Specifies an IP address, either IPv4 or IPv6, for a physical address that corresponds to the customer address.
- **Rule** Specifies which type of virtualization mechanism the policy entry uses. The acceptable values for this parameter are:
  - TranslationMethodEncap. Network Virtualization Generic Routing Encapsulation (NVGRE).
  - TranslationMethodNone. None.

- **Type** Specifies the type of the look up record. This is a return field only and can't be set by the user.
  - Dynamic
  - Static
  - GatewayWildcard
  - L2Only
- **VirtualSubnetID** Specifies an ID for the virtual subnet that the customer address belongs to. The acceptable values for this parameter are integers from 4096 through 16777214.

The full command line looks like this:

```
New-NetVirtualizationLookupRecord
   -CustomerAddress <String>
   -MACAddress <String>
   -ProviderAddress <String>
   -Rule <RuleType>
   -VirtualSubnetID <UInt32>

   [-Type <Type>]
```

New-NetVirtualizationProviderAddress

The New-NetVirtualizationProviderAddress cmdlet assigns a provider address to a network interface for use with HNV. A provider address is an IPv4 or IPv6 address that HNV uses for multiple virtual customer addresses. To assign a provider address, specify the IP address, an interface, and the IP prefix length for the subnet. You can also specify a virtual local area network (VLAN) ID.

To create a provider address, specify the following values:

- **InterfaceIndex** Specifies the index for a network interface that has HNV enabled.
- **PrefixLength** Specifies the length of the IP prefix.
- **ProviderAddress** Specifies an IP address configured for the network interface. You can use IPv4 or IPv6 addresses.
- **VlanID** Specifies an ID for a LAN for the provider address.

The full command line looks like this:

```
New-NetVirtualizationProviderAddress
   -InterfaceIndex <UInt32>
   -PrefixLength <Byte>
   -ProviderAddress <String>
   [-VlanID <UInt16> ]
```

New-NetVirtualizationProviderRoute

The New-NetVirtualizationProviderRoute cmdlet creates a network route for HNV. HNV uses provider routes to direct network traffic on the physical network. To create a provider route, specify the subnet as an IP prefix, the interface, and the address for the next hop gateway.

To create a provider address route, specify the following values:

- **DestinationPrefix** Specifies an IP prefix, as a string, for the destination network. You can specify an IPv4 or IPv6 address. Use prefix notation: 0.0.0.0/0.
- **InterfaceIndex** Specifies the index for a network interface that has HNV enabled.
- **NextHop** Specifies an IP address for the next hop gateway for this route.

The full command line looks like this:

```
New-NetVirtualizationProviderRoute
   -DestinationPrefix <String>
   -InterfaceIndex <UInt32>
   -NextHop <String>
```

These four PowerShell APIs provide all the policy needed by the Hyper-V virtual switch to act as the distributed router and properly encapsulate and de-encapsulate packets.

Central policy store

There are other central policy store implementations, but for this text, VMM is assumed as the central policy store. The central policy store plays a critical role in HNV, ensuring that the policy records are up to date and validating that the policy matches the physical network. As the central policy store, VMM provides several key pieces of functionality:

- Ensures that as VMs are migrated, the policy across the hosts with VMs on the VM network has the latest policy including the correct CA-to-PA mapping.
- Ensures that the HNV Gateway is properly configured including forwarding, VPN, and NAT configurations to the external networks.
- Ensures that IP and MAC addresses are unique within a virtual network.
- Ensures that VSIDs and RDIDs are unique within a single data center.
- Ensures that the PAs are routable across the physical network that the hosts are on.

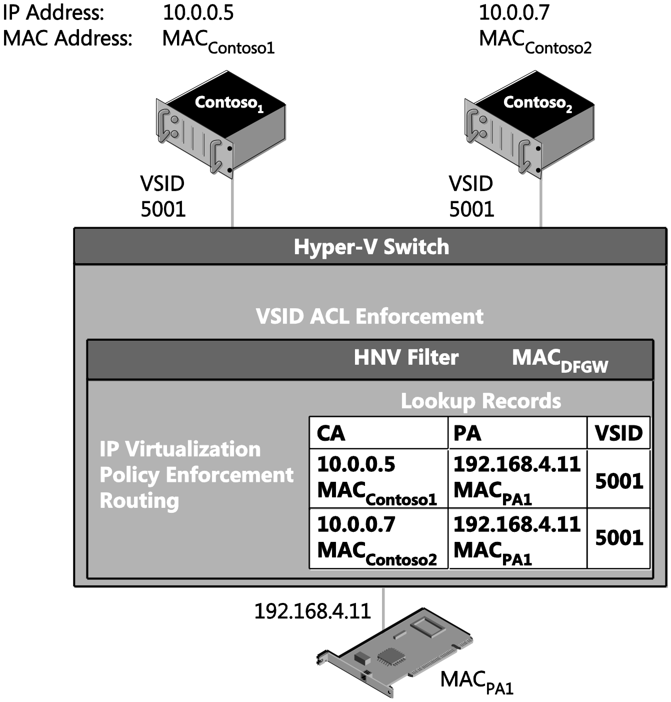
The policies that are pushed out will regularly be updated so there is a need to constantly refresh policies across all hosts whenever changes occur.

Packet flows

The next step in understanding how HNV works is to walk through various packet flows, from simple cases where the packet does not leave the Hyper-V host to a scenario where a packet goes through a gateway. After going through this section, you should understand what traffic goes over the wire and what traffic the HNV filter handles. In addition, you should understand what the NVGRE packets look like on the wire in different scenarios.

Two VMs on same virtual subnet, same host

The simplest packet flow to understand is when two VMs are on the same virtual subnet and on the same host. Figure 1-7 shows the setup in this scenario showing that both VMs are on the same Hyper-V host and the same VSID. In the HNV filter, the configured lookup records are shown.



CA	PA	VSID
10.0.0.5 $MAC_{Contoso1}$	192.168.4.11 MAC_{PA1}	5001
10.0.0.7 $MAC_{Contoso2}$	192.168.4.11 MAC_{PA1}	5001

FIGURE 1-7 Packet flow for two VMs on the same virtual subnet and the same host.

When Contoso1 communicates with Contoso2, the packet flow is as follows:

1. Contoso1 sends ARP messages for 10.0.0.7.
2. The Hyper-V switch broadcasts the ARP to:

   - All the local VMs on VSID 5001
   - The HNV filter
3. Contoso2 responds to the ARP for IP address 10.0.0.7 on VSID 5001 with $MAC_{Contoso2}$.
4. Contoso1 learns to use $MAC_{Contoso2}$ for 10.0.0.7.

$MAC_{Contoso1}$ → $MAC_{Contoso2}$	**10.0.0.5 → 10.0.0.7**

5. Contoso1 sends an IP packet destined for Contoso2.
6. This packet is delivered to the Hyper-V switch and gets the VSID (5001) associated with the sender's VM network adapter as out-of-band (OOB) data.

> ***NOTE*** **The Hyper-V switch is the component that does all VSID ACL'ing as the VSID is configured on a particular VM network adapter, so all packets coming from that VM network adapter are tagged with the OOB data specifying the configured VSID. VSID ACL'ing ensures that the packet's destination VM network adapter is in the same VM network as the VSID that the packet originated on.**

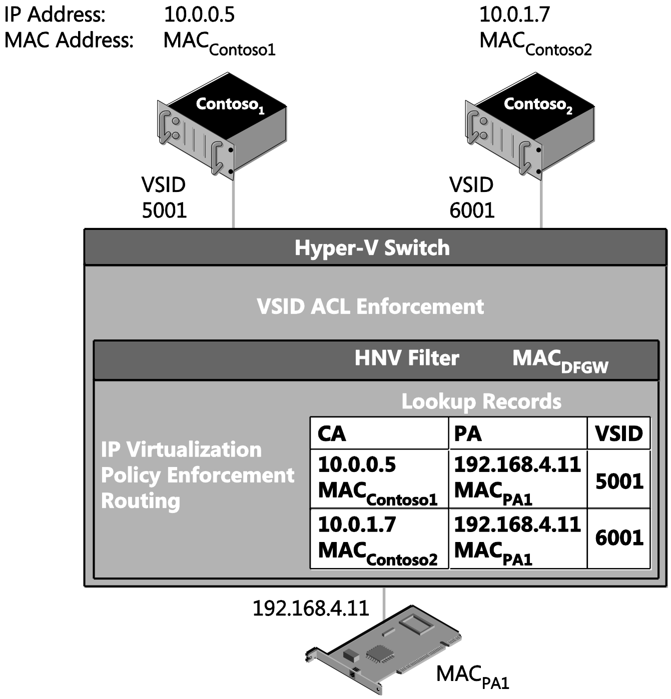
7. Since the Hyper-V switch is an L2 switch, it knows all the MAC addresses of VMs attached to it. It also ACLs the VSID so that VMs can only see packets destined for VSIDs it is configured for. The switch sees that the packet it being sent to $MAC_{Contoso2}$ on VSID 5001 and matches that with the VM network adapter for Contoso2.
8. The Hyper-V switch then delivers the packet to Contoso2.

This is the simplest packet flow related to HNV. Two things to emphasize about this scenario are:

- When the two VMs are on the same host, there is no NVGRE encapsulation and the HNV filter never sees the packet.
- The VSID ACL'ing happens in the Hyper-V switch itself based on the VSID provided in the OOB data.

Two VMs on different virtual subnets, same host

When VMs are on the same host but on different virtual subnets, in the virtual network they need to be routed (in the virtual network) not switched. Figure 1-8 shows the setup in this scenario showing that both VMs are on the same Hyper-V host but on different VSIDs. The configured lookup records are shown in the HNV filter.

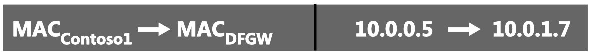


CA	PA	VSID
10.0.0.5 $MAC_{Contoso1}$	192.168.4.11 MAC_{PA1}	5001
10.0.1.7 $MAC_{Contoso2}$	192.168.4.11 MAC_{PA1}	6001

FIGURE 1-8 Packet flow for two VMs on different virtual subnets but the same host.

When Contoso1 communicates with Contoso2, the packet flow is as follows:

1. The IP addresses are on different subnets so Contoso1 sends ARP messages for the default gateway, not the IP address directly.
2. The Hyper-V switch broadcasts the ARP to the HNV filter.
3. The HNV filter responds to the ARP with MACDFGW. MACDFGW is associated with the HNV filter itself. As noted previously, the IP address associated with this MAC address is the .1 address in the virtual subnet, 10.0.0.1 in this case.
4. Contoso1 learns to use MACDFGW for the default gateway (10.0.0.1) on VSID 5001 (included in the OOB data for this MAC address).
5. Contoso1 sends an IP packet destined for Contoso2 with the MAC address of the default gateway so that the packet is delivered to the default gateway to be routed appropriately.



6. This packet is delivered to the Hyper-V switch and gets the VSID (5001) associated with the sender's VM network adapter as out-of-band (OOB) data.

7. The HNV filter verifies Contoso1 and Contoso2 are in same VM network, otherwise the packet is dropped.
8. The HNV filter uses its lookup records to determine the PA of the destination VM. If there is no lookup record for the IP address, the packet is dropped.

$\text{MAC}_{\text{Contoso1}}$ → *$\text{MAC}_{\text{Contoso2}}$*	**10.0.0.5 → 10.0.1.7**

In this scenario, the PA for the destination VM is the PA for the local HNV filter, so the HNV filter rewrites the packet to change the destination MAC address to $\text{MAC}_{\text{Contoso2}}$.

9. The HNV filter updates the OOB data with the VSID to the destination VSID (6001).

> ***NOTE*** **Since the VSID is carried in the GRE key there is space in the packet for only one VSID. The destination VSID is put into the packet such that it can go with the packet over the wire.**

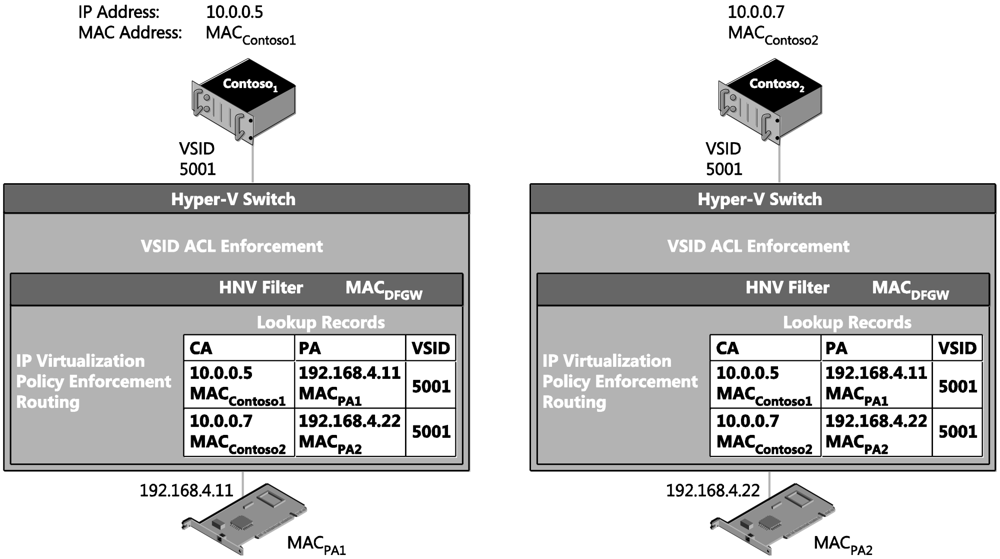
10. Since the Hyper-V switch is an L2 switch, it knows all the MAC addresses of VMs attached to it. It also does the VSID ACL'ing. The switch sees that the packet is being sent to $\text{MAC}_{\text{Contoso2}}$ on VSID 6001 and matches that with the VM network adapter for Contoso2.
11. The Hyper-V switch then delivers the packet to Contoso2.

This scenario shows how HNV acts as a router for the virtual network. A few things to emphasize are:

- When the two VMs are on the same host, there is no NVGRE encapsulation.
- In contrast with VMs on the same virtual subnet, in this scenario the HNV filter receives and processes the packet because it is acting as the default gateway. This a good example of how HNV acts as a distributed router.
- When acting as the default gateway, the HNV filter updates the VSID and the destination MAC address of the packet to match the receiver's VSID and MAC address.

Two VMs on the same virtual subnet, different hosts, dynamic IP address learning not enabled

The two previous packet flow examples showed what happens in HNV when VMs are on the same host. More commonly, however, VMs are on different Hyper-V hosts. The following example walks through the packet flow where two VMs are on the same virtual subnet but on different hosts. Figure 1-9 shows the setup in this scenario where the VMs are on different Hyper-V hosts but the same VSID. In the HNV filter, the configured lookup records are shown.


IP Address: 10.0.0.5
MAC Address: $\text{MAC}_{\text{Contoso1}}$
Contoso_1
VSID 5001
10.0.0.7
$\text{MAC}_{\text{Contoso2}}$
Contoso_2
VSID 5001
Hyper-V Switch
VSID ACL Enforcement
HNV Filter
MAC_{DFGW}
Lookup Records
IP Virtualization Policy Enforcement Routing
Hyper-V Switch
VSID ACL Enforcement
HNV Filter
MAC_{DFGW}
Lookup Records
IP Virtualization Policy Enforcement Routing
192.168.4.11
MAC_{PA1}
192.168.4.22
MAC_{PA2}

CA	PA	VSID
10.0.0.5 $\text{MAC}_{\text{Contoso1}}$	192.168.4.11 MAC_{PA1}	5001
10.0.0.7 $\text{MAC}_{\text{Contoso2}}$	192.168.4.22 MAC_{PA2}	5001

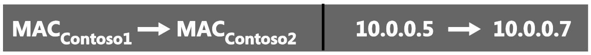
FIGURE 1-9 Packet flow for two VMs on the same virtual subnet, but on different hosts and with dynamic IP address learning not enabled.

When Contoso1 communicates with Contoso2, the packet flow is as follows:

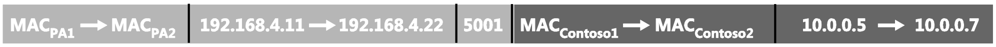
1. Contoso1 sends ARP messages for 10.0.0.7.
2. The Hyper-V switch broadcasts the ARP to:
   - All the local VMs on VSID 5001
   - The HNV filter
3. The HNV filter responds to the ARP for IP address 10.0.0.7 on VSID 5001 on behalf of Contoso2 with $\text{MAC}_{\text{Contoso2}}$.

NOTE **The ARP is not broadcast to the network.**

4. Contoso1 learns to use $\text{MAC}_{\text{Contoso2}}$ for 10.0.0.7.


$\text{MAC}_{\text{Contoso1}}$ → $\text{MAC}_{\text{Contoso2}}$
10.0.0.5 → 10.0.0.7

5. Contoso1 sends an IP packet destined for Contoso2.
6. This packet is delivered to the Hyper-V switch and gets the VSID associated with the packet as out-of-band (OOB) data.
7. The Hyper-V switch sees that $\text{MAC}_{\text{Contoso2}}$ is not on the local Hyper-V switch and sends it to the HNV filter.


MAC_{PA1} → MAC_{PA2}
192.168.4.11 → 192.168.4.22
5001
$\text{MAC}_{\text{Contoso1}}$ → $\text{MAC}_{\text{Contoso2}}$
10.0.0.5 → 10.0.0.7

8. The HNV filter finds the lookup record associated, on the 5001 VSID, with the CA (10.0.0.7) and the MAC address ($\text{MAC}_{\text{Contoso2}}$). It finds the PA (192.168.4.22) associated with this lookup record. With all the required information, it now encapsulates the original IP packet with an NVGRE packet that will be delivered on the wire.

NOTE **The VSID (5001) is explicitly put into the packet and can be seen on the wire. The lighter shaded part of the packet is called the outer packet and the darker shaded part of the packet is called the inner packet.**

9. The NVGRE encapsulated packet gets passed through the networking stack and on to the physical network adapter to the physical network infrastructure.
10. The physical network infrastructure uses the outer packet (destination MAC and IP address) to route the packet to the specified Hyper-V host.
11. The Hyper-V host corresponding to IP address 192.168.4.22 and MAC_{PA2} receives the packet and delivers it to the HNV filter.

NOTE **The PA is associated with the HNV filter. This is why the PA should not be the same IP address as the underlying physical NIC or NIC team.**

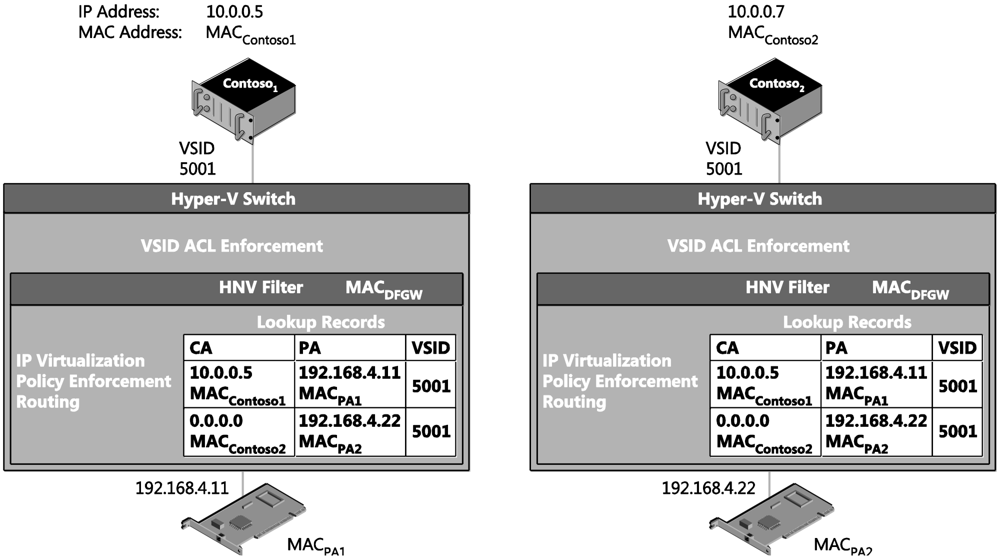
12. The HNV filter de-encapsulates the packet and now has the inner packet (the original IP packet), plus the OOB data contains the VSID (5001).
13. The HNV filter delivers the IP packet (the inner packet from the previous step) to the Hyper-V switch with the corresponding OOB data containing the VSID (5001).
14. The Hyper-V switch does any required VSID ACL'ing and then delivers the IP packet to the VM. In this scenario, the packet was NVGRE encapsulated but the encapsulation was completely transparent to either the sending or receiving VM.

This packet flow includes NVGRE encapsulation and provides a more complete picture of how HNV typically works. A few things to emphasize are:

- When dynamic IP address learning is not enabled, even if the destination VM is not on the on the local host the ARP is processed by the HNV filter and doesn't flow over the wire.
- On the wire, the physical network routes the packet based on the outer packet (PA IP address and MAC address) and is unaware of the inner packet (CA IP address and MAC address).
- Even when a packet is NVGRE encapsulated on the wire, this encapsulation is always transparent to the sending and receiving VMs.

Two VMs on the same virtual subnet, different hosts, dynamic IP address learning enabled

This scenario is a variant of the previous one only instead of a static lookup record being configured for a particular MAC address, an L2-only record is configured. Figure 1-10 shows the setup in this scenario where the VMs are on different Hyper-V hosts but the same VSID and an L2-only lookup record is configured. The figure shows the configured lookup records in the HNV filter, including the dynamic record indicated by the 0.0.0.0 IP address for Contoso2.



CA	PA	VSID
10.0.0.5 MAC$_{Contoso1}$	192.168.4.11 MAC$_{PA1}$	5001
0.0.0.0 MAC$_{Contoso2}$	192.168.4.22 MAC$_{PA2}$	5001

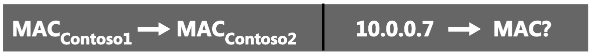
CA	PA	VSID
10.0.0.5 MAC$_{Contoso1}$	192.168.4.11 MAC$_{PA1}$	5001
0.0.0.0 MAC$_{Contoso2}$	192.168.4.22 MAC$_{PA2}$	5001

FIGURE 1-10 Packet flow when dynamic IP address learning is enabled for a MAC address.

When Contoso1 communicates with Contoso2 and dynamic IP address learning is enabled for a particular MAC address, the packet flow is as follows:

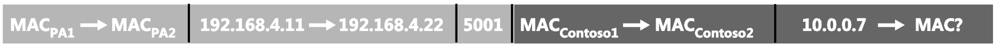
1. Contoso1 sends ARP messages for 10.0.0.7.
2. The Hyper-V switch broadcasts the ARP to:
   - All the local VMs on VSID 5001
   - The HNV filter
3. The HNV filter does not find a lookup record for the 10.0.0.7 IP address.

> ***NOTE*** **This will only happen the first time trying to communicate with an IP address. After this packet, flow happens once there are two lookup records for this MAC address. One is the L2-only record and the second is a newly created dynamic lookup record that matches the IP address with the MAC address. At this point the packet flow would be identical to the previous one.**



4. The HNV creates a unicast ARP packet for each PA that has an L2-only record associated with it on the 5001 VSID.

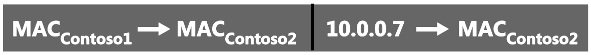
> ***NOTE*** **Just like a physical network, ARPs are not broadcast outside the current subnet because a subnet is a broadcast boundary.**


MAC_{PA1} → MAC_{PA2}
192.168.4.11 → 192.168.4.22
5001
$MAC_{Contoso1}$ → $MAC_{Contoso2}$
10.0.0.7 → MAC?

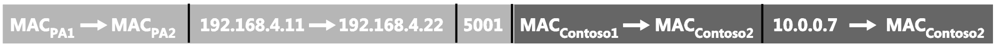
5. The HNV filter encapsulates the ARP request into an NVGRE packet.
6. The NVGRE encapsulated packet gets passed through the networking stack and on to the physical network adapter to the physical network infrastructure.
7. The physical network infrastructure uses the outer packet (destination MAC and IP address) to route the packet to the specified Hyper-V host.
8. The Hyper-V host corresponding to IP address 192.168.4.22 and MAC_{PA2} receives the packet and delivers it to the HNV filter.

> ***NOTE*** **The PA is associated with the HNV filter. This is why the PA should not be the same IP address as the underlying physical NIC or NIC team.**

9. The HNV filter de-encapsulates the packet and now has the inner packet (the original ARP packet), plus the OOB data contains the VSID (5001).
10. The HNV filter delivers the ARP packet (the inner packet from the previous step) to the Hyper-V switch with the corresponding OOB data containing the VSID (5001).
11. The Hyper-V switch does any required VSID ACL'ing and then delivers the ARP packet to the VM.


$MAC_{Contoso1}$ → $MAC_{Contoso2}$
10.0.0.7 → $MAC_{Contoso2}$

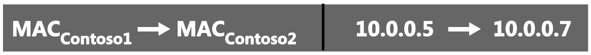
12. If the VM's network adapter matches the MAC address in the ARP request it reply's with an ARP reply packet and it is sent to the Hyper-V switch.
13. The Hyper-V switch attaches the OOB data with the VSID (5001).
14. The Hyper-V switch passes the ARP reply packet to the HNV filter since the packet is not destined for a local VM.
15. The HNV filter sees this is an ARP packet, and if there is no dynamic IP address record for this CA MAC/IP address and PA MAC/IP address pair for the local HNV filter, it will be added. Then a notification of a new dynamic record is sent to VMM so that the updated policy can be sent to other Hyper-V hosts.


MAC_{PA1} → MAC_{PA2}
192.168.4.11 → 192.168.4.22
5001
$\text{MAC}_{\text{Contoso1}}$ → $\text{MAC}_{\text{Contoso2}}$
10.0.0.7 → $\text{MAC}_{\text{Contoso2}}$

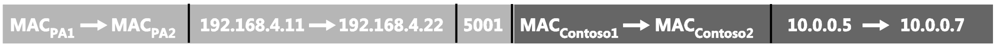
16. The HNV filter finds a lookup record for the destination MAC address ($\text{MAC}_{\text{Contoso1}}$) and encapsulates the packet for transportation on the physical network.
17. The NVGRE encapsulated packet is passed through the networking stack, to the physical network adapter, and then to the physical network infrastructure.
18. The physical network infrastructure uses the outer packet (destination MAC and IP Address) to route the packet to the specified Hyper-V host.
19. The Hyper-V host corresponding to IP address 192.168.4.11 and MAC_{PA1} receives the packet and delivers it to the HNV filter.

NOTE **The PA is associated with the HNV filter. This is why the PA should not be the same IP address as the underlying physical NIC or NIC team.**

20. The HNV filter de-encapsulates the packet, and since it is an ARP packet, adds a dynamic record for the CA MAC/IP address and PA MAC/IP address pair. Notification of a new dynamic record is sent to VMM so that the updated policy can be sent to other Hyper-V hosts. The Hyper-V extensible switch then delivers the ARP packet to Contoso1.
21. Contoso1 learns to use $\text{MAC}_{\text{Contoso2}}$ for 10.0.0.7.


$\text{MAC}_{\text{Contoso1}}$ → $\text{MAC}_{\text{Contoso2}}$
10.0.0.5 → 10.0.0.7

22. Contoso1 sends an IP packet destined for Contoso2.
23. This packet is delivered to the Hyper-V switch and gets the VSID (5001) associated with the packet as out-of-band (OOB) data.
24. The Hyper-V switch sees that $\text{MAC}_{\text{Contoso2}}$ is not on the local Hyper-V switch and sends it to the HNV filter.


MAC_{PA1} → MAC_{PA2}
192.168.4.11 → 192.168.4.22
5001
$\text{MAC}_{\text{Contoso1}}$ → $\text{MAC}_{\text{Contoso2}}$
10.0.0.5 → 10.0.0.7

25. The HNV filter finds the lookup record associated, on the 5001 VSID, with the CA (10.0.0.7) and the MAC address ($\text{MAC}_{\text{Contoso2}}$). It finds the PA (192.168.4.22) associated with this lookup record. With all the required information, it now encapsulates the original IP packet with an NVGRE packet that will be delivered on the wire.

NOTE **The VSID (5001) is explicitly put into the packet and can be seen on the wire. The lighter shaded part of the packet is called the outer packet and the darker shaded part of the packet is called the inner packet.**

26. The NVGRE encapsulated packet is passed through the networking stack, to the physical network adapter, and then to the physical network infrastructure.

27. The physical network infrastructure uses the outer packet (destination MAC and IP address) to route the packet to the specified Hyper-V host.

28. The Hyper-V host corresponding to IP address 192.168.4.22 and MAC_{PA2} receives the packet and delivers it to the HNV filter.

NOTE **The PA is associated with the HNV filter. This is why the PA should not be the same IP address as the underlying physical NIC or NIC team.**

29. The HNV filter de-encapsulates the packet and now has the inner packet (the original IP packet), plus the OOB data contains the VSID (5001).

30. The HNV filter delivers the IP packet (the inner packet from the previous step) to the Hyper-V switch with the corresponding OOB data containing the VSID (5001).

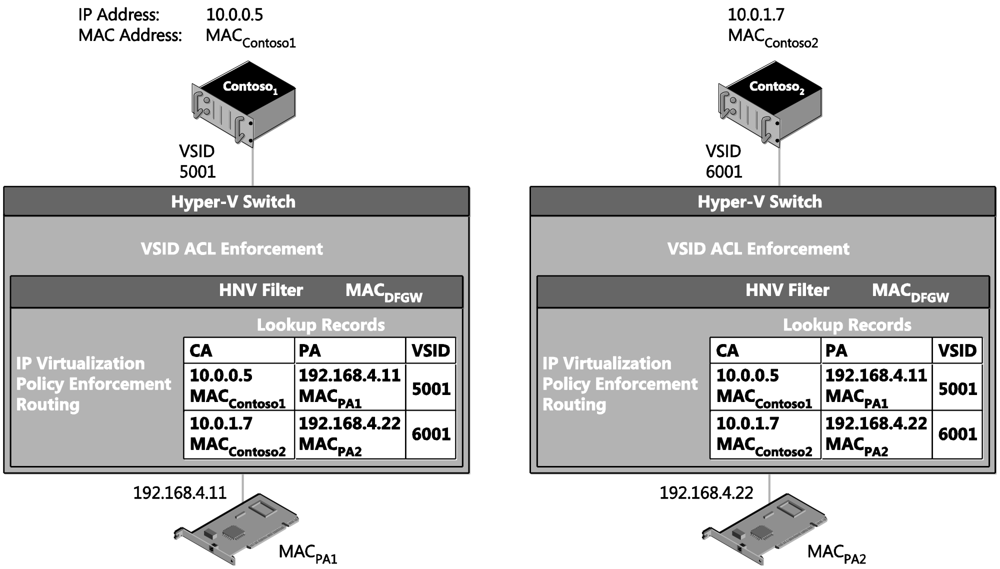
31. The Hyper-V switch does any required VSID ACL’ing and then delivers the IP packet to the VM. In this scenario, the packet was NVGRE encapsulated but the encapsulation was completely transparent to both the sending and receiving VM.

These steps show the packet flow when dynamic IP address learning is enabled. A few things to emphasize are:

- The ARP packets flow across the wire as unicast encapsulated packets.
- The destination VM generates the actual ARP reply in this packet flow.
- Neither the source nor the destination Hyper-V host has the CA MAC/IP address and PA MAC/IP address pair; it will be added as a new dynamic lookup record and VMM will be notified of the new lookup record.

Two VMs on different virtual subnets, different hosts

The next scenario shows what happens when the VMs are on different Hyper-V hosts. This example walks through the packet flow where two VMs are on different virtual subnets and on the same host. Figure 1-11 shows the setup in this scenario with both VMs on different Hyper-V hosts and different VSID. The figure shows the configured lookup records in the HNV filter.



CA	PA	VSID
10.0.0.5 $MAC_{Contoso1}$	192.168.4.11 MAC_{PA1}	5001
10.0.1.7 $MAC_{Contoso2}$	192.168.4.22 MAC_{PA2}	6001

CA	PA	VSID
10.0.0.5 $MAC_{Contoso1}$	192.168.4.11 MAC_{PA1}	5001
10.0.1.7 $MAC_{Contoso2}$	192.168.4.22 MAC_{PA2}	6001

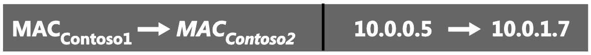
FIGURE 1-11 Packet flow for two VMs on different virtual subnets and different hosts.

When Contoso1 communicates with Contoso2, the packet flow is as follows:

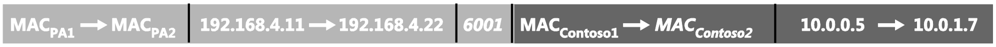
1. The IP addresses are on different subnets so Contoso1 sends ARP messages for the default gateway, not the IP address directly.
2. The Hyper-V switch broadcasts the ARP to the HNV filter.
3. The HNV filter responds to the ARP with MAC_{DFGW}. MAC_{DFGW} is associated with the HNV filter itself. As noted previously, the IP address associated with this MAC address is the .1 address in the virtual subnet, 10.0.0.1 in this case.
4. Contoso1 learns to use MAC_{DFGW} for the default gateway (10.0.0.1) on VSID 5001 (included in the OOB data for this MAC address).

$MAC_{Contoso1}$ → MAC_{DFGW}	10.0.0.5 → 10.0.1.7

5. Contoso1 sends an IP packet destined for Contoso2 with the MAC address of the default gateway so that the packet is delivered to the default gateway to be routed appropriately.
6. This packet is delivered to the Hyper-V switch and gets the VSID associated (5001) with the sender's VM network adapter as out-of-band (OOB) data.
7. The HNV filter verifies Contoso1 and Contoso2 are in same VM network, otherwise the packet is dropped.
8. The HNV filter uses its lookup records to determine the PA of the destination VM. If there is no lookup record for the IP address, the packet is dropped.


$\text{MAC}_{\text{Contoso1}}$ → $\text{MAC}_{\text{Contoso2}}$
10.0.0.5 → 10.0.1.7

9. The HNV filter rewrites the packet to change the destination MAC address to $\text{MAC}_{\text{Contoso2}}$.


MAC_{PA1} → MAC_{PA2}
192.168.4.11 → 192.168.4.22
6001
$\text{MAC}_{\text{Contoso1}}$ → $\text{MAC}_{\text{Contoso2}}$
10.0.0.5 → 10.0.1.7

10. The HNV filter finds the lookup record associated with the CA (10.0.1.7) and the MAC Address ($\text{MAC}_{\text{Contoso2}}$). It sees the VSID (6001) associated with the lookup record is in the same VM network as the original VSID (5001) but is different. It uses the VSID (6001) of the destination VM network adapter when creating the NVGRE encapsulated packet. It finds the PA (192.168.4.22) associated with this lookup record. With all the required information, it now encapsulates the original IP packet with an NVGRE packet that will be delivered on the wire.

> ***NOTE*** **The VSID (6001) is explicitly put into the packet and can be seen on the wire. The lighter shaded part of the packet is called the outer packet and the darker shaded part of the packet is called the inner packet.**

11. The NVGRE encapsulated packet is passed through the networking stack, to the physical network adapter, and then to the physical network infrastructure.
12. The physical network infrastructure uses the outer packet (destination MAC and IP address) to route the packet to the specified Hyper-V host.
13. The Hyper-V host corresponding to IP address 192.168.4.22 and MAC_{PA2} receives the packet and delivers it to the HNV filter.

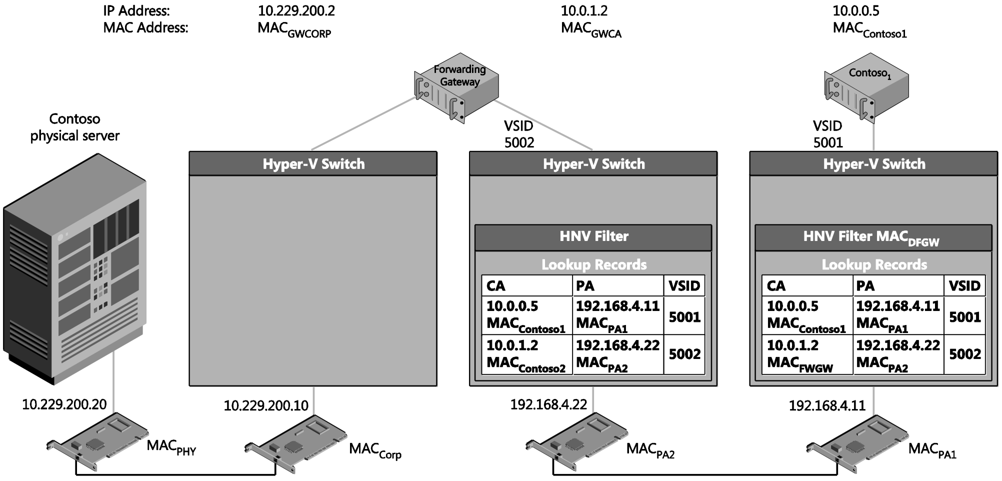
> ***NOTE*** **The PA is associated with the HNV filter. This is why the PA should not be the same IP address as the underlying physical NIC or NIC team.**

14. The HNV filter de-encapsulates the packet and now has the inner packet (the original IP packet), plus the OOB data contains the VSID (6001).
15. The HNV filter delivers the IP packet (the inner packet from the previous step) to the Hyper-V switch with the corresponding OOB data containing the VSID (6001).
16. The Hyper-V switch does any required VSID ACL'ing and then delivers the IP packet to the VM. In this scenario, the packet was NVGRE encapsulated but the encapsulation was completely transparent to both the sending and receiving VMs.

This packet flow includes NVGRE encapsulation and provides a more complete picture of how HNV typically works. One thing to emphasize is that when sending packets on different subnets the destination VSID is used.

VM to a physical host through the inbox forwarding gateway

Figure 1-12 shows a packet flow configuration where a VM in a virtual network is communicating through a forwarding gateway to a physical server.



CA	PA	VSID
10.0.0.5 MAC$_{Contoso1}$	192.168.4.11 MAC$_{PA1}$	5001
10.0.1.2 MAC$_{Contoso2}$	192.168.4.22 MAC$_{PA2}$	5002

CA	PA	VSID
10.0.0.5 MAC$_{Contoso1}$	192.168.4.11 MAC$_{PA1}$	5001
10.0.1.2 MAC$_{FWGW}$	192.168.4.22 MAC$_{PA2}$	5002

FIGURE 1-12 Forwarding gateway packet flow scenario.

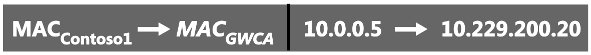
When Contoso1 communicates with the Contoso physical server, the packet flow is as follows:

1. These IP addresses are on different subnets so Contoso1 sends ARP messages to the default gateway and not the IP address directly.
2. The Hyper-V switch broadcasts the ARP to the HNV filter.
3. The HNV filter responds to the ARP with MAC$_{DFGW}$. MAC$_{DFGW}$ is associated with the HNV filter itself. As noted previously, the IP address associated with this MAC address is the .1 address in the virtual subnet, 10.0.0.1 in this case.
4. Contoso1 learns to use MAC$_{DFGW}$ for the default gateway (10.229.202.1) on VSID 5001 (included in the OOB data for this MAC address).

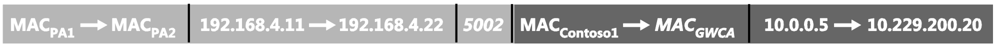
MAC$_{Contoso1}$ → MAC$_{DFGW}$	10.0.0.5 → 10.229.200.20

5. Contoso1 sends an IP packet destined for the physical Contoso server with the MAC address of the default gateway so that the packet is delivered to the default gateway to be routed appropriately.
6. This packet is delivered to the Hyper-V switch and gets the VSID associated (5001) with the sender's VM network adapter as out-of-band (OOB) data.
7. The HNV filter sees that there is a customer route for the 10.229.200.x subnet that points to 10.0.1.2 (the forwarding gateway) as the next hop address.

8. The HNV filter uses its lookup records to determine the PA of the next hop VM. If there is no lookup record for the IP address, the packet is dropped.



9. The HNV filter rewrites the packet to change the destination MAC address to MAC$_{GWCA}$ (the MAC address of the forwarding gateway).



10. The HNV filter finds the lookup record associated with the CA (10.0.1.2) and the MAC address (MAC$_{GWCA}$). It sees the VSID (5002) associated with the lookup record is in the same VM network as the original VSID (5001) but is different. It uses the VSID (5002) of the destination VM network adapter when creating the NVGRE encapsulated packet. It finds the PA (192.168.4.22) associated with this lookup record. With all the required information, it now encapsulates the original IP packet with an NVGRE packet that will be delivered on the wire.

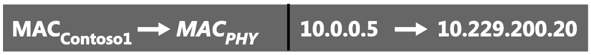
> ***NOTE*** **The VSID (5002) is explicitly put into the packet and can be seen on the wire. The lighter shaded part of the packet is called the outer packet and the darker shaded part of the packet is called the inner packet.**

11. The NVGRE encapsulated packet is passed through the networking stack, to the physical network adapter, and then to the physical network infrastructure.
12. The physical network infrastructure uses the outer packet (destination MAC and IP address) to route the packet to the specified Hyper-V host.
13. The Hyper-V host corresponding to IP address 192.168.4.22 and MAC$_{PA2}$ receives the packet and delivers it to the HNV filter.

> ***NOTE*** **The PA is associated with the HNV filter. This is why the PA should not be the same IP address as the underlying physical NIC or NIC team.**

14. The HNV filter de-encapsulates the packet and now has the inner packet (the original IP packet), plus the OOB data contains the VSID (5002).
15. The HNV filter delivers the IP packet (the inner packet from the previous step) to the Hyper-V switch with the corresponding OOB data containing the VSID (5002).
16. The Hyper-V switch does any required VSID ACL'ing and then delivers the IP packet to the VM. In this scenario, the packet was NVGRE encapsulated but the encapsulation was completely transparent to both the sending and receiving VMs.
17. At this point, the packet is in the forwarding gateway VM.

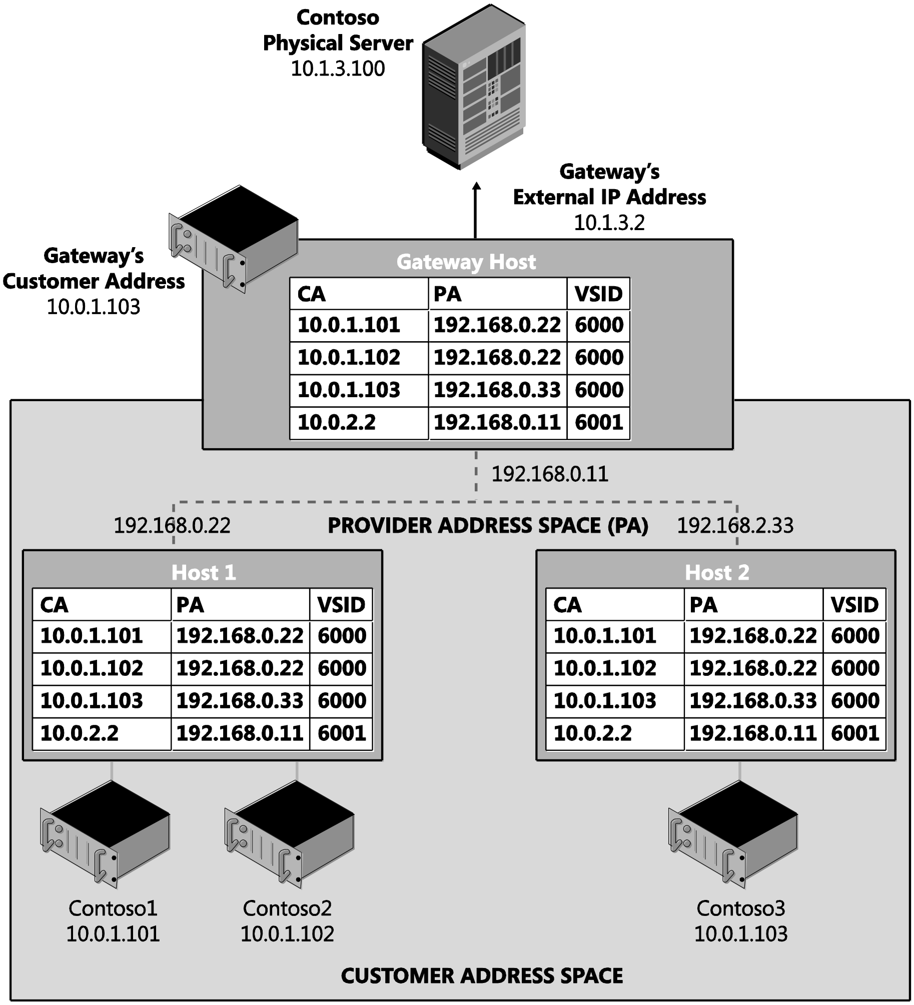
18. The forwarding gateway VM has been configured to forward packets between the two network interfaces in the VM.



19. The forwarding gateway VM's network adapter connected to the physical network rewrites the destination MAC address to MAC_{PHY} (corresponding to the MAC address of the Contoso physical server).
20. The Contoso physical server receives the packet none the wiser that it was originated in a virtual network.

Hyper-V Network Virtualization: Simple setup

While most HNV deployments are performed using VMM, the basic concepts are best understood by doing a simple HNV deployment using Windows PowerShell. This section provides a step-by-step walkthrough on how to deploy an HNV network that includes a forwarding gateway. The result will be two VMs sitting in a virtual network with a forwarding gateway connecting them to the physical server. Figure 1-13 shows the setup and configuration. The walkthrough begins with Host 1, then goes through Host 2, then the gateway, and finally the physical machine.



Gateway Host

CA	PA	VSID
10.0.1.101	192.168.0.22	6000
10.0.1.102	192.168.0.22	6000
10.0.1.103	192.168.0.33	6000
10.0.2.2	192.168.0.11	6001

Host 1

CA	PA	VSID
10.0.1.101	192.168.0.22	6000
10.0.1.102	192.168.0.22	6000
10.0.1.103	192.168.0.33	6000
10.0.2.2	192.168.0.11	6001

Host 2

CA	PA	VSID
10.0.1.101	192.168.0.22	6000
10.0.1.102	192.168.0.22	6000
10.0.1.103	192.168.0.33	6000
10.0.2.2	192.168.0.11	6001

FIGURE 1-13 Simple HNV setup.

The setup prerequisites are as follows:

- Four physical hosts on the same L2 switch
  - Each physical host running Windows Server 2012 R2
  - Each host with a minimum of 8 gigabytes of RAM
  - One host (the gateway host) with two network adapters.
- Four VHDs with Windows Server 2012 R2 installed
- Two of the VMs placed on Host 1, one on Host 2, and one on the gateway host

NOTE You can get away with using only three physical hosts if you want to drop the third virtual machine on a separate host.

Host 1 setup

The sections that follow provide the steps for:

- Host 1 VM setup
- Host 1 HNV policy configuration
- Configuring Contoso1
- Configuring Contoso2

Host 1 VM setup

The steps for Host 1 VM setup are as follows:

1. Install Windows Server 2012 R2.
2. Install the Hyper-V role using the following command:

```
Install-WindowsFeature
   –Name Hyper-V
   -ComputerName localhost
   -IncludeManagementTools
   –Restart
```

NOTE Windows PowerShell must be run as Administrator for this command to work.

3. Create a virtual switch using the following command:

```
New-VMSwitch "HNVSwitch"
   -NetAdapterName "Ethernet"
   -AllowManagementOS 1
```

This creates a virtual switch. You might need to update the name of the –NetAdapterName parameter if your network adapter is not called “Ethernet”. Go to the Network And Sharing Center in Control Panel to determine the name of your network.

4. Create the VM for Contoso1. You will need to update the –VHDPath with the actual location of a VHD with Windows Server 2012 R2 installed.

```
New-VM
   -Name "Contoso1"
   –VHDPath "Location of VHD"
   -SwitchName "HNVSwitch"
   -Generation 2
```

5. Create the VM for Contoso2. You will need to update the –VHDPath with the actual location of a VHD with Windows Server 2012 R2 installed.

```
New-VM
   -Name "Contoso2"
   –VHDPath "Location of VHD"
   -SwitchName "HNVSwitch"
   -Generation 2
```

6. Set the MAC address for the three VM network adapters to a static MAC address

> **NOTE** This step is not required but makes the rest of the configuration easier.

```
Set-VMNetworkAdapter
   -VMName "Contoso1"
   -StaticMacAddress 600060000101
Set-VMNetworkAdapter
   -VMName "Contoso2"
   -StaticMacAddress 600060000102
```

Host 1 HNV policy configuration

The steps for Host 1 HNV policy configuration are as follows:

1. Run an elevated PowerShell window. All the following steps will run in this PowerShell Window.
2. Configure the PA on the host. This PA should not be the IP address of the underlying NIC or NIC team. It should be routable on the network to and from any other host that has a VM using the same virtual network.

```
$iface =
   ((Get-VMSwitch -name "HNVSwitch").NetAdapterInterfaceDescription)).ifIndex)

New-NetVirtualizationProviderAddress
   -ProviderAddress 192.168.0.22
   -MacAddress 334455667788
   -InterfaceIndex ((get-netadapter –InterfaceDescription $iface))
   -PrefixLength 24
```

> **NOTE** You can set a VLAN on a PA to associate the PA to a VLAN. You might want to do this if for instance you want all HNV traffic to be isolated and on the same VLAN.

3. Set the VSID on the three VM network adapters to the same VSID, in this case 6000. This is the command that specifies a VM network adapter should send virtualized network traffic.

   ```
   Set-VMNetworkAdapter
      -VMName "Contoso1"
      -VirtualSubnetId 6000
   Set-VMNetworkAdapter
      -VMName "Contoso2"
      -VirtualSubnetId 6000
   ```

4. Configure the customer route for the 10.0.1.0/24 subnet. This is the IP address range for the virtual subnet 6000.

   ```
   New-NetVirtualizationCustomerRoute
      -RoutingDomainID "{12345678-1000-2000-3000-123456780001}"
      -VirtualSubnetID 6000
      -DestinationPrefix 10.0.1.0/24
      -NextHop 0.0.0.0
   ```

> ***NOTE*** **This is where a VSID is assigned to a routing domain (called a VM network in VMM). Unlike a VSID which is explicitly carried in the packet, the routing domain is a logical concept that is enforced in the HNV module. HNV does all routing inside a routing domain. This means the next hop is "onlink" and HNV will handle all the routing. This is also where a VSID is assigned an IP address subnet. A virtual subnet supports all valid sized subnets so it is not required to be a /24 but can range from a /24 to a /30.**

5. Configure the customer route for the Contoso gateway's 10.0.2.0/24 subnet. This is the IP address range for the virtual subnet 6001. It is also the customer route for the HNV Gateway.

   ```
   New-NetVirtualizationCustomerRoute
      -RoutingDomainID "{12345678-1000-2000-3000-123456780001}"
      -VirtualSubnetID 6001
      -DestinationPrefix 10.0.2.0/24
      -NextHop 0.0.0.0
   ```

> ***NOTE*** **The Gateway must be on its own VSID. This is because policy (as seen in the next step) is configured to send all traffic outside the virtual network through the Gateway as the next hop. This limits other VMs being on the same VSID as the Gateway. The Gateway and any VSID that has to go through the Gateway must be on the same routing domain.**

6. Configure a wildcard route for Contoso's routing domain. This route must be on the same VSID as the gateway and its next hop will be the IP address of the HNV Gateway.

```
New-NetVirtualizationCustomerRoute
   -RoutingDomainID "{12345678-1000-2000-3000-123456780001}"
   -VirtualSubnetID 6001
   -DestinationPrefix 0.0.0.0/0
   -NextHop 10.0.2.2
```

7. Configure a static route for the default gateway in the virtual subnet. The provider address should be the provider address of the host that this lookup record is being set on.

```
New-NetVirtualizationLookupRecord
   -CustomerAddress 10.0.1.1
   -VirtualSubnetID 6000
   -MACAddress 600060000100
   -ProviderAddress 192.168.0.22
   -Type Static
   -Rule TranslationMethodEncap
```

NOTE **This is a new feature in Windows Server 2012 R2 that allows VMs on a virtualized network to ping the default HNV Gateway. This is a useful diagnostic mechanism to ensure based connectivity on a virtualized network. The default gateway for a subnet is fixed at the .1 address for that subnet.**

8. Configure the lookup record for Contoso1 located on Host 1.

```
New-NetVirtualizationLookupRecord
   -CustomerAddress 10.0.1.101
   -VirtualSubnetID 6000
   -MACAddress  600060000101
   -ProviderAddress 192.168.0.22
   -Rule TranslationMethodEncap
```

9. Configure the lookup record for Contoso2 located on Host 1.

```
New-NetVirtualizationLookupRecord
   -CustomerAddress 10.0.1.102
   -VirtualSubnetID 6000
   -MACAddress  600060000102
   -ProviderAddress 192.168.0.22
   -Rule TranslationMethodEncap
```

10. Configure the lookup record for Contoso3 located on Host 2.

```
New-NetVirtualizationLookupRecord
   -CustomerAddress 10.0.1.103
   -VirtualSubnetID 6000
   -MACAddress 600060000203
   -ProviderAddress 192.168.0.33
   -Rule TranslationMethodEncap
```

11. Configure the lookup record for Gateway located on the Gateway host.

```
New-NetVirtualizationLookupRecord
   -CustomerAddress 10.0.2.2
   -VirtualSubnetID 6001
   -MACAddress 600160010301
   -ProviderAddress 192.168.0.11
   -Rule TranslationMethodEncap
```

You can validate the setup by running the four commands described next.

Get-VMNetworkAdapter

The Get-VMNetworkAdapter cmdlet gets the virtual network adapters of the specified virtual machine, snapshot, or management operating system:

```
Get-VMNetworkAdapter * | fl VMName,MacAddress,VirtualSubnetId,SwitchName
```

The output from this command should look like this:

```
VMName          : Contoso1
MacAddress      : 600060000101
VirtualSubnetId : 6000
SwitchName      : HNVSwitch

VMName          : Contoso2
MacAddress      : 600060000102
VirtualSubnetId : 6000
SwitchName      : HNVSwitch
```

Get-NetVirtualizationCustomerRoute

The Get-NetVirtualizationCustomerRoute cmdlet gets virtual network routes in a virtual network:

```
Get-NetVirtualizationCustomerRoute
```

The output from this command should look like this:

```
RoutingDomainID:          {12345678-1000-2000-3000-123456780001}
VirtualSubnetID:          6000
DestinationPrefix:        10.0.1.0/24
```

```
NextHop:                  0.0.0.0
Metric:                   0

RoutingDomainID:          {12345678-1000-2000-3000-123456780001}
VirtualSubnetID:          6001
DestinationPrefix:        10.0.2.0/24
NextHop:                  0.0.0.0
Metric:                   0

RoutingDomainID:         {12345678-1000-2000-3000-123456780001}
VirtualSubnetID:         6001
DestinationPrefix:       0.0.0.0/0
NextHop:                 10.0.2.2
Metric:                  0
```

Get-NetVirtualizationLookupRecord

The Get-NetVirtualizationLookupRecord cmdlet gets lookup record policy entries for IP addresses that belong to a virtual network:

```
Get-NetVirtualizationLookupRecord
```

The output from this command should look like this:

```
CustomerAddress:          10.0.1.1
VirtualSubnetID:          6000
MACAddress:               600060000100
ProviderAddress:          192.168.0.22
CustomerID:               {00000000-0000-0000-0000-000000000000}
Context:
Rule:                     TranslationMethodEncap
VMName:
UseVmMACAddress:          False
Type:                     Static

CustomerAddress:          10.0.1.101
VirtualSubnetID:          6000
MACAddress:               600160010101
ProviderAddress:          192.168.0.22
CustomerID:               {00000000-0000-0000-0000-000000000000}
Context:
Rule:                     TranslationMethodEncap
VMName:
UseVmMACAddress:          False
Type:                     Static
```

```
CustomerAddress:         10.0.1.102
VirtualSubnetID:         6000
MACAddress:              600160010102
ProviderAddress:         192.168.0.22
CustomerID:              {00000000-0000-0000-0000-000000000000}
Context:
Rule:                    TranslationMethodEncap
VMName:
UseVmMACAddress:         False
Type:                    Static

CustomerAddress:         10.0.1.103
VirtualSubnetID:         6000
MACAddress:              600160010203
ProviderAddress:         192.168.0.33
CustomerID:              {00000000-0000-0000-0000-000000000000}
Context:
Rule:                    TranslationMethodEncap
VMName:
UseVmMACAddress:         False
Type:                    Static

CustomerAddress:         10.0.2.2
VirtualSubnetID:         6001
MACAddress:              600160010301
ProviderAddress:         192.168.0.11
CustomerID:              {00000000-0000-0000-0000-000000000000}
Context:
Rule:                    TranslationMethodEncap
VMName:
UseVmMACAddress:         False
Type:                    Static
```

Get-NetVirtualizationProviderAddress

The Get-NetVirtualizationProviderAddress cmdlet gets provider addresses configured in Hyper-V Network Virtualization:

```
Get-NetVirtualizationProviderAddress
```

The output from this command should look like this:

```
ProviderAddress:            192.168.0.22
InterfaceIndex:             12
PrefixLength:               24
VlanID:                       0
AddressState:               Preferred
MACAddress:                 334455667788
ManagedByCluster:           False
```

Configuring Contoso1

The steps for configuring Contoso1 are as follows:

1. Start the Contoso1 VM on Host1.
2. Log into the VM and run an elevated PowerShell window. All the following steps will run in this PowerShell window.
3. Configure the Contoso1 VM's NIC to have a static IP address that matches the IP address of the VM.

   ```
   $wmi = Get-WmiObject win32_networkadapterconfiguration
      -filter "ipenabled = 'true'"
   $wmi.EnableStatic("10.0.1.101", "255.255.255.0")
   $wmi.SetGateways("10.0.1.1", 1)
   ```

4. Configure Windows Firewall to allow pings.

   ```
   Import-Module NetSecurity
   Set-NetFirewallRule
      -DisplayName "File and Printer Sharing (Echo Request - ICMPv4-In)"
      -enabled True
   ```

5. On the Contoso1 VM, you should now be able to ping the default gateway.

   ```
   ping 10.0.1.1
   ```

Configuring Contoso2

The steps for configuring Contoso2 are as follows:

1. Start the Contoso2 VM on Host1.
2. Log into the VM and run an elevated PowerShell window. All the following steps will run in this PowerShell window.
3. Configure the Contoso2 VM's NIC to have a static IP address that matches the IP address of the VM.

   ```
   $wmi = Get-WmiObject win32_networkadapterconfiguration
      -filter "ipenabled = 'true'"
   $wmi.EnableStatic("10.0.1.102", "255.255.255.0")
   $wmi.SetGateways("10.0.1.1", 1)
   ```

4. Configure Windows Firewall to allow pings.

```
Import-Module NetSecurity
Set-NetFirewallRule
   -DisplayName "File and Printer Sharing (Echo Request – ICMPv4-In)"
   -enabled True
```

5. On the Contoso2 VM, you should now be able to ping both the default gateway and Contoso1.

```
ping 10.0.1.1
ping 10.0.1.101
```

6. On the Contoso1 VM, you should now be able to ping Contoso2.

```
ping 10.20.20.102
```

Congratulations! If this is working, you have a virtual network up and running!

Host 2 setup

The sections that follow provide the steps for:

- Host 2 VM setup
- Host 2 HNV policy configuration
- Contoso3 configuration

Host 2 VM setup

The steps for Host 2 VM setup are as follows:

1. Install Windows Server 2012 R2.
2. Install the Hyper-V role using the following command:

```
Install-WindowsFeature
   –Name Hyper-V
   -ComputerName localhost
   -IncludeManagementTools
   –Restart
```

NOTE **Windows PowerShell must be run as Administrator for this command to work.**

3. Create a virtual switch using the following command:

```
New-VMSwitch "HNVSwitch"
   -NetAdapterName "Ethernet"
   -AllowManagementOS 1
```

You might need to update the name of the –NetAdapterName parameter if your network adapter is not called “Ethernet”. Go to the Network and Sharing Center in

Control Panel to determine the name of your network adapter.

4. Create the VM for Contoso3. You will need to update the –VHDPath with the actual location of a VHD with Windows Server 2012 R2 installed.

   ```
   New-VM
      -Name "Contoso3"
      –VHDPath "Location of VHD"
      -SwitchName "HNVSwitch"
      -Generation 2
   ```

5. Set the MAC address for the three VM network adapters to a static MAC address.

   ```
   Set-VMNetworkAdapter
      -VMName "Contoso3"
      -StaticMacAddress 600060000203
   ```

> ***NOTE*** **This step is not required but makes the rest of the configuration easier.**

Host 2 HNV policy configuration

The steps for Host 2 HNV policy configuration are as follows:

1. Run an elevated PowerShell window. All the following steps will run in this PowerShell window.

2. Configure the PA address on the host. This PA should not be the IP address of the underlying NIC or NIC team. It should be routable on the network to and from any other host that has VM using the same virtual network.

   ```
   $iface =
      ((Get-VMSwitch -name "HNVSwitch").NetAdapterInterfaceDescription)).ifIndex)
   New-NetVirtualizationProviderAddress
      -ProviderAddress 192.168.0.33
      -MacAddress 445566778899
      -InterfaceIndex ((get-netadapter –InterfaceDescription $iface))
      -PrefixLength 24
   ```

> ***NOTE*** **On the provider address, you can set a VLAN. This associates a PA to a VLAN. You might want to do this if for instance you want all HNV traffic to be isolated and on the same VLAN. The PA assigned to this host must be different then the PA assigned to Host 1.**

3. Set the VSID on the three VM network adapters to the same VSID, in this case 6000. This is the command that specifies a VM network adapter should send virtualized network traffic.

   ```
   Set-VMNetworkAdapter
      -VMName "Contoso3"
      -VirtualSubnetId 6000
   ```

4. Configure the customer route for the 10.0.1.0/24 subnet. This is the IP address range for the virtual subnet 6000.

   ```
   New-NetVirtualizationCustomerRoute
      -RoutingDomainID "{12345678-1000-2000-3000-123456780001}"
      -VirtualSubnetID 6000
      -DestinationPrefix 10.0.1.0/24
      -NextHop 0.0.0.0
   ```

5. Configure the customer route for the Contoso gateway's 10.0.2.0/24 subnet. This is the IP address range for the virtual subnet 6001. It is also the customer route for the HNV Gateway.

   ```
   New-NetVirtualizationCustomerRoute
      -RoutingDomainID "{12345678-1000-2000-3000-123456780001}"
      -VirtualSubnetID 6001
      -DestinationPrefix 10.0.2.0/24
      -NextHop 0.0.0.0
   ```

6. Configure the wildcard route for Contoso's routing domain. This route must be on the same VSID as the gateway and its next hop will be the IP address of the HNV Gateway.

   ```
   New-NetVirtualizationCustomerRoute
      -RoutingDomainID "{12345678-1000-2000-3000-123456780001}"
      -VirtualSubnetID 6001
      -DestinationPrefix 0.0.0.0/0
      -NextHop 10.0.2.2
   ```

7. Configure a static route for the default gateway in the virtual subnet. The provider address should be the provider address of the host that this lookup record is being set on.

   ```
   New-NetVirtualizationLookupRecord
      -CustomerAddress 10.0.1.1
      -VirtualSubnetID 6000
      –MACAddress 600060000200
      -ProviderAddress 192.168.0.33
      -Type Static
      -Rule TranslationMethodEncap
   ```

8. Configure the lookup record for Contoso1 located on Host 1.

```
New-NetVirtualizationLookupRecord
   -CustomerAddress 10.0.1.101
   -VirtualSubnetID 6000
   -MACAddress  600060000101
   -ProviderAddress 192.168.0.22
   -Rule TranslationMethodEncap
```

9. Configure the lookup record for Contoso2 located on Host 1.

```
New-NetVirtualizationLookupRecord
   -CustomerAddress 10.0.1.102
   -VirtualSubnetID 6000
   -MACAddress 600060000102
   -ProviderAddress 192.168.0.22
   -Rule TranslationMethodEncap
```

10. Configure the lookup record for Contoso3 located on Host 2.

```
New-NetVirtualizationLookupRecord
   -CustomerAddress 10.0.1.103
   -VirtualSubnetID 6000
   -MACAddress 600060000203
   -ProviderAddress 192.168.0.33
   -Rule TranslationMethodEncap
```

11. Configure the lookup record for Gateway located on the Gateway host.

```
New-NetVirtualizationLookupRecord
   -CustomerAddress 10.0.2.2
   -VirtualSubnetID 6001
   -MACAddress 600160010301
   -ProviderAddress 192.168.0.11
   -Rule TranslationMethodEncap
```

You can validate the setup by running the four commands described next.

Get-VMNetworkAdapter

The Get-VMNetworkAdapter cmdlet gets the virtual network adapters of the specified virtual machine, snapshot, or management operating system:

```
Get-vmnetworkadapter * | fl VMName,MacAddress,VirtualSubnetId,SwitchName
```

The output from this command should look like this:

```
VMName:                          Contoso3
MacAddress:                      600060000203
VirtualSubnetId:                 6000
SwitchName:                      HNVSwitch
```

Get-NetVirtualizationCustomerRoute

The Get-NetVirtualizationCustomerRoute cmdlet gets virtual network routes in a virtual network:

```
Get-NetVirtualizationCustomerRoute
```

The output from this command should look like this:

```
RoutingDomainID:                 {12345678-1000-2000-3000-123456780001}
VirtualSubnetID:                 6000
DestinationPrefix:               10.0.1.0/24
NextHop:                         0.0.0.0
Metric:                          0

RoutingDomainID:                 {12345678-1000-2000-3000-123456780001}
VirtualSubnetID:               6001
DestinationPrefix:             10.0.2.0/24
NextHop:                       0.0.0.0
Metric:                        0

RoutingDomainID:               {12345678-1000-2000-3000-123456780001}
VirtualSubnetID:               6001
DestinationPrefix:             0.0.0.0/0
NextHop:                       10.0.2.2
Metric:                        0
```

Get-NetVirtualizationLookupRecord

The Get-NetVirtualizationLookupRecord cmdlet gets lookup record policy entries for IP addresses that belong to a virtual network:

```
Get-NetVirtualizationLookupRecord
```

The output from this command should look like this:

```
CustomerAddress:                 10.0.1.1
VirtualSubnetID:                 6000
MACAddress:                      600060000100
ProviderAddress:                 192.168.0.22
CustomerID:                      {00000000-0000-0000-0000-000000000000}
Context:
Rule:                            TranslationMethodEncap
VMName:
UseVmMACAddress:                 False
Type:                            Static

CustomerAddress:                 10.0.1.101
VirtualSubnetID:                 6000
```

```
MACAddress:                      600160010101
ProviderAddress:                 192.168.0.22
CustomerID:                      {00000000-0000-0000-0000-000000000000}
Context:
Rule:                            TranslationMethodEncap
VMName:
UseVmMACAddress  :               False
Type:                            Static

CustomerAddress:                 10.0.1.102
VirtualSubnetID:                 6000
MACAddress:                      600160010102
ProviderAddress:                 192.168.0.22
CustomerID:                      {00000000-0000-0000-0000-000000000000}
Context:
Rule:                            TranslationMethodEncap
VMName:
UseVmMACAddress:                 False
Type:                            Static

CustomerAddress:                 10.0.1.103
VirtualSubnetID:                 6000
MACAddress:                      600160010203
ProviderAddress:                 192.168.0.33
CustomerID:                      {00000000-0000-0000-0000-000000000000}
Context:
Rule:                            TranslationMethodEncap
VMName:
UseVmMACAddress:                 False
Type:                            Static

CustomerAddress:                 10.0.2.2
VirtualSubnetID:                 6001
MACAddress:                      600160010301
ProviderAddress:                 192.168.0.11
CustomerID:                      {00000000-0000-0000-0000-000000000000}
Context:
Rule:                            TranslationMethodEncap
VMName:
UseVmMACAddress:                 False
Type:                            Static
```

Get-NetVirtualizationProviderAddress

The Get-NetVirtualizationProviderAddress cmdlet gets provider addresses configured in Hyper-V Network Virtualization:

```
Get-NetVirtualizationProviderAddress
```

The output from this command should look like this:

```
ProviderAddress:                 192.168.0.33
InterfaceIndex:                  12
PrefixLength:                    24
VlanID:                          0
AddressState:                    Preferred
MACAddress:                      445566778899
ManagedByCluster:                False
```

Configuring Contoso3

The steps for configuring Contoso3 are as follows:

1. Start the Contoso3 VM on Host 2.
2. Log into the VM and run an elevated PowerShell window. All the following steps will run in this PowerShell window.
3. Configure the Contoso3 VM's NIC to have a static IP address that matches the IP address of the VM

   ```
   $wmi = Get-WmiObject win32_networkadapterconfiguration
      -filter "ipenabled = 'true'"
   $wmi.EnableStatic("10.0.1.103", "255.255.255.0")
   $wmi.SetGateways("10.0.1.1", 1)
   ```

4. Configure Windows Firewall to allow pings.

   ```
   Import-Module NetSecurity
   Set-NetFirewallRule
      -DisplayName "File and Printer Sharing (Echo Request - ICMPv4-In)"
      -enabled True
   ```

5. On the Contoso3 VM, you should now be able to ping the default gateway, Contoso1, and Contoso2

   ```
   ping 10.0.1.1
   ping 10.0.1.101
   ping 10.0.1.102
   ```

Gateway host setup

The sections that follow provide the steps for:

- Gateway VM setup
- Gateway host HNV policy configuration
- MT-GW configuration

Gateway VM setup

The steps for Gateway VM setup are as follows:

1. Install Windows Server 2012 R2.
2. Install the Hyper-V role using the following command:

   ```
   Install-WindowsFeature
      –Name Hyper-V
      -ComputerName localhost
      -IncludeManagementTools
      –Restart
   ```

NOTE **Windows PowerShell must be run as Administrator for this command to work.**

3. Create a virtual switch for the HNV network using the following command:

   ```
   New-VMSwitch "HNVSwitch"
      -NetAdapterName "Ethernet"
      -AllowManagementOS 1
   ```

   You might need to update the name of the –NetAdapterName parameter if your network adapter is not called “Ethernet”. Go to the Network and Sharing Center in Control Panel to determine the name of your network adapter.

4. Create a virtual switch for the external network that will connect to the physical server.

   ```
   New-VMSwitch "ExternalSwitch"
      -NetAdapterName "Ethernet2"
      -AllowManagementOS 1
   ```

   You might need to update the name of the –NetAdapterName parameter if your network adapter is not called “Ethernet2”. Go to the Network and Sharing Center in Control Panel to determine the name of your network adapter.

5. Create the VM for MT-GW. You will need to update the –VHDPath with the actual location of a VHD with Windows Server 2012 R2 installed.

   ```
   New-VM
      -Name "MT-GW"
      –VHDPath "Location of VHD"
      -Generation 2
   ```

6. Create a VM network adapter on the “External” virtual switch in the MT-GW VM.

```
Add-VMNetworkAdapter
   -VMName MT-GW
   -Name IntNIC
   -SwitchName "HNVSwitch"
```

7. Set the MAC address for the VM network adapter on the HNV network to a static MAC address.

```
Set-VMNetworkAdapter
   -VMName "MT-GW"
   -StaticMacAddress 600160010301
```

NOTE **This step is not required but makes the rest of the configuration easier.**

8. Create a VM network adapter on the “External” virtual switch in the MT-GW VM.

```
Add-VMNetworkAdapter
   -VMName MT-GW
   -Name ExtNIC
   -SwitchName "ExternalSwitch"
```

9. Set the MAC address for the VM network adapter on the External network to a static MAC address.

```
Set-VMNetworkAdapter
   -VMName "MT-GW"
   -StaticMacAddress 600160010302
```

NOTE **This step is not required but makes the rest of the configuration easier.**

Gateway host HNV policy configuration

The steps for Gateway host HNV policy configuration are as follows:

1. Run an elevated PowerShell window. All the following steps will run in this PowerShell window.
2. Configure the PA address on the host. This PA should not be the IP address of the underlying NIC or NIC team. It should be routable on the network to and from any other host that has a VM using the same virtual network.

```
$iface =
   ((Get-VMSwitch -name "HNVSwitch").NetAdapterInterfaceDescription)).ifIndex)
New-NetVirtualizationProviderAddress
   -ProviderAddress 192.168.0.11
   -MacAddress 223344556677
   -InterfaceIndex ((get-netadapter –InterfaceDescription $iface -PrefixLength
24 ))
```

3. Set the VSID on the VM network adapters to the Gateway's VSID, in this case 6001. This is the command that specifies a VM network adapter should send virtualized network traffic.

```
Set-VMNetworkAdapter
   -VMName "MT-GW"
   -VirtualSubnetId 6001
```

4. Configure the customer route for the 10.0.1.0/24 subnet. This is the IP address range for the virtual subnet 6000.

```
New-NetVirtualizationCustomerRoute
   -RoutingDomainID "{12345678-1000-2000-3000-123456780001}"
   -VirtualSubnetID 6000
   -DestinationPrefix 10.0.1.0/24
   -NextHop 0.0.0.0
```

5. Configure the customer route for the Contoso gateway's 10.0.2.0/24 subnet. This is the IP address range for the virtual subnet 6001. It is also the customer route for the HNV Gateway.

```
New-NetVirtualizationCustomerRoute
   -RoutingDomainID "{12345678-1000-2000-3000-123456780001}"
   -VirtualSubnetID 6001
   -DestinationPrefix 10.0.2.0/24
   -NextHop 0.0.0.0
```

6. Configure the wildcard route for Contoso's routing domain. This route must be on the same VSID as the gateway and its next hop will be the IP address of the HNV Gateway.

```
New-NetVirtualizationCustomerRoute
   -RoutingDomainID "{12345678-1000-2000-3000-123456780001}"
   -VirtualSubnetID 6001
   -DestinationPrefix 0.0.0.0/0
   -NextHop 10.0.2.2
```

7. Configure a static route for the default gateway in the virtual subnet. The provider address should be the provider address of the host that this lookup record is being set on.

```
New-NetVirtualizationLookupRecord
   -CustomerAddress 10.0.2.1
   -VirtualSubnetID 6001
   –MACAddress 600160010300
   -ProviderAddress 192.168.0.33
   -Type Static
   -Rule TranslationMethodEncap
```

8. Configure the lookup record for Contoso1 located on Host 1.

```
New-NetVirtualizationLookupRecord
   -CustomerAddress 10.0.1.101
   -VirtualSubnetID 6000
   -MACAddress  600060000101
   -ProviderAddress 192.168.0.22
   -Rule TranslationMethodEncap
```

9. Configure the lookup record for Contoso2 located on Host 1.

```
New-NetVirtualizationLookupRecord
   -CustomerAddress 10.0.1.102
   -VirtualSubnetID 6000
   -MACAddress  600060000102
   -ProviderAddress 192.168.0.22
   -Rule TranslationMethodEncap
```

10. Configure the lookup record for Contoso3 located on Host 2.

```
New-NetVirtualizationLookupRecord
   -CustomerAddress 10.0.1.103
   -VirtualSubnetID 6000
   -MACAddress 600060000203
   -ProviderAddress 192.168.0.33
   -Rule TranslationMethodEncap
```

11. Configure the lookup record for Gateway located on the Gateway host.

```
New-NetVirtualizationLookupRecord
   -CustomerAddress 10.0.2.2
   -VirtualSubnetID 6001
   -MACAddress 600160010301
   -ProviderAddress 192.168.0.11
   -Rule TranslationMethodEncap
```

12. Set the VM network adapter isolation mode for the gateway.

```
Set-VMNetworkAdapterIsolation
   -VMName MT-GW
   -VMNetworkAdapterName IntNic
   -MultiTenantStack on
   -IsolationMode NativeVirtualSubnet
```

13. Map a routing domain in the MT-GW to a VM network adapter on the host virtual switch.

```
Add-VmNetworkAdapterRoutingDomainMapping
   -VMName MT-GW
   -VMNetworkAdapterName IntNic
   -RoutingDomainId "{00000000-0000-0000-0000-000000000000}"
```

```
        -RoutingDomainName "ContosoTenant"
        -IsolationId 6001
        -IsolationName "ContosoGWSubnet"
```

You can validate the setup by running the four commands described next.

Get-VMNetworkAdapter

The Get-VMNetworkAdapter cmdlet gets the virtual network adapters of the specified virtual machine, snapshot, or management operating system:

```
Get-VMNetworkAdapter * | fl VMName,MacAddress,VirtualSubnetId,SwitchName
```

The output from this command should look like this:

```
     VMName:                             MT-GW
     MacAddress:                         600160010301
     VirtualSubnetId:                    6001
     SwitchName:                         HNVSwitch
```

Get-NetVirtualizationCustomerRoute

The Get-NetVirtualizationCustomerRoute cmdlet gets virtual network routes in a virtual network:

```
Get-NetVirtualizationCustomerRoute
```

The output from this command should look like this:

```
RoutingDomainID:                          {12345678-1000-2000-3000-123456780001}
VirtualSubnetID:                          6000
DestinationPrefix:                        10.0.1.0/24
NextHop:                                  0.0.0.0
Metric:                                   0

RoutingDomainID:                          {12345678-1000-2000-3000-123456780001}
VirtualSubnetID:                          6001
DestinationPrefix:                        10.0.2.0/24
NextHop:                                  0.0.0.0
Metric:                                   0

RoutingDomainID:                          {12345678-1000-2000-3000-123456780001}
VirtualSubnetID:                          6001
DestinationPrefix:                        0.0.0.0/0
NextHop:                                  10.0.2.2
Metric:                                   0
```

Get-NetVirtualizationLookupRecord

The Get-NetVirtualizationLookupRecord cmdlet gets lookup record policy entries for IP addresses that belong to a virtual network:

```
Get-NetVirtualizationLookupRecord
```

The output from this command should look like this:

```
CustomerAddress:                          10.0.1.1
VirtualSubnetID:                          6000
MACAddress:                               600060000100
ProviderAddress:                          192.168.0.22
CustomerID:                               {00000000-0000-0000-0000-000000000000}
Context:
Rule:                                     TranslationMethodEncap
VMName:
UseVmMACAddress:                          False
Type:                                     Static

CustomerAddress:                          10.0.1.101
VirtualSubnetID:                          6000
MACAddress:                               600160010101
ProviderAddress:                          192.168.0.22
CustomerID:                               {00000000-0000-0000-0000-000000000000}
Context:
Rule:                                     TranslationMethodEncap
VMName:
UseVmMACAddress:                          False
Type:                                     Static

CustomerAddress:                          10.0.1.102
VirtualSubnetID:                          6000
MACAddress:                               600160010102
ProviderAddress:                          192.168.0.22
CustomerID:                               {00000000-0000-0000-0000-000000000000}
Context:
Rule:                                     TranslationMethodEncap
VMName:
UseVmMACAddress:                          False
Type:                                     Static

CustomerAddress:                          10.0.1.103
VirtualSubnetID:                          6000
MACAddress:                               600160010203
ProviderAddress:                          192.168.0.33
CustomerID:                               {00000000-0000-0000-0000-000000000000}
```

```
Context:
Rule:                                   TranslationMethodEncap
VMName:
UseVmMACAddress:                        False
Type:                                   Static

CustomerAddress:                        10.0.2.2
VirtualSubnetID:                        6001
MACAddress:                             600160010301
ProviderAddress:                        192.168.0.11
CustomerID:                             {00000000-0000-0000-0000-000000000000}
Context:
Rule:                                   TranslationMethodEncap
VMName:
UseVmMACAddress:                        False
Type:                                   Static
```

Get-NetVirtualizationProviderAddress

The Get-NetVirtualizationProviderAddress cmdlet gets provider addresses configured in Hyper-V Network Virtualization:

```
Get-NetVirtualizationProviderAddress
```

The output from this command should look like this:

```
ProviderAddress:                        192.168.0.11
InterfaceIndex:                         12
PrefixLength:                           24
VlanID:                                 0
AddressState:                           Preferred
MACAddress:                             223344556677
ManagedByCluster:                       False
```

Configuring MT-GW

The steps for configuring MT-GW are as follows:

1. Start the MT-GW VM on the gateway host.
2. Log into the VM and run an elevated PowerShell window. All the following steps will run in this PowerShell window.
3. Configure Windows Firewall to allow pings.

```
Import-Module NetSecurity
Set-NetFirewallRule
   -DisplayName "File and Printer Sharing (Echo Request - ICMPv4-In)"
   -enabled True
```

4. Set the IP address of the internal NIC on the gateway that is connected to the virtual network.

```
New-NetIPAddress 10.0.2.2
   -InterfaceAlias "ContosoGWSubnet"
   -PrefixLength 24
```

5. Add a new netroute to the gateway because there is no default route for each tenant. Routes to the tenants' subnets must explicitly be added and the next hop should be set to the .1 default gateway.

```
New-NetRoute
   -InterfaceAlias "ContosoGWSubnet"
   -AddressFamily IPv4
   -DestinationPrefix 10.0.1.0/24
   -NextHop 10.0.2.1
```

6. Enable forwarding on the external NIC.

```
Set-NetIPInterface -InterfaceAlias "External" -Forwarding Enabled
```

NOTE **You will need to replace the InterfaceAlias property with the InterfaceAlias of the NIC that will be used for the physical network.**

7. Enable forwarding on the ContosoGWSubnet network interface that was created in Step 5.

```
Set-NetIPInterface
   -InterfaceAlias "ContosoGWSubnet"
   -Forwarding Enabled
```

8. Configure the IP address for the external IP address.

```
New-NetIPAddress
   -InterfaceAlias "External"
   -IPAddress 10.1.3.2
   -PrefixLength 24
```

NOTE **You will need to replace the InterfaceAlias property with the InterfaceAlias of the NIC that will be used for the physical network.**

9. On the MT-GW VM, you should now be able to ping the default gateway, Contoso1, Contoso2, and Contoso3.

```
ping 10.0.2.1
ping 10.0.1.101
ping 10.0.1.102
ping 10.0.1.103
```

Contoso physical host setup

The steps for the Contoso physical host setup are as follows:

1. Install Windows Server 2012 R2.
2. Configure the IP address of the physical host using the following command:

   ```
   New-NetIPAddress
      -InterfaceAlias "Ethernet 2"
      -IPAddress 10.1.3.100
      -PrefixLength 24
   ```

   Replace the InterfaceAlias property with the InterfaceAlias of the NIC that will be used for the physical network.
3. Configure the IP address of the external NIC of the gateway to be the next hop for the 10.0.1.X subnet in the virtual network.

   ```
   New-NetRoute
      -InterfaceAlias "Ethernet 2"
      -DestinationPrefix 10.0.1.0/24
      -NextHop 10.1.3.2
   ```

   Replace the InterfaceAlias property with the InterfaceAlias of the NIC that will be used for the external network.
4. Configure the IP address of the external NIC of the gateway to be the next hop for the 10.0.2.X subnet in the virtual network.

   ```
   New-NetRoute
      -InterfaceAlias "Ethernet 2"
      -DestinationPrefix 10.0.2.0/24
      -NextHop 10.1.3.2.0.1
   ```

   Replace the InterfaceAlias property with the InterfaceAlias of the NIC that will be used for the external network.
5. Configure Windows Firewall to allow pings.

   ```
   Import-Module NetSecurity
   Set-NetFirewallRule
      -DisplayName "File and Printer Sharing (Echo Request - ICMPv4-In)"
      -enabled True
   ```

With these last commands, you should now be able to ping between all five of the servers (Contoso1, Contoso2, Contoso3, MT-GW, and the Contoso physical host) participating in the network. You can use Message Analyzer to look at the packets flowing as you do the pings to validate that they match the packet flows discussed in the previous section.

CHAPTER 2

Implementing cloud computing with Network Virtualization

The previous chapter examined the internal workings of Hyper-V Network Virtualization (HNV). This chapter will identify some key cloud computing scenarios and examine how HNV helps cloud service providers enable these scenarios. It will also cover the different gateway functionalities in Windows Server 2012 R2.

Key cloud computing scenarios enabled by HNV

Cloud computing enables organizations to leverage the compute, storage, and networking infrastructure of a cloud services provider to meet their business needs with agility and minimal capital expenditure. Without leveraging cloud computing, catering to dynamic compute needs would consume significant resources and time for a business. HNV helps cloud service providers offer cost-effective infrastructure as a service (IaaS) to their customers. The following are some of the key scenarios enabled by HNV:

- Cloud hosting
- Cloud bursting
- Cloud-based disaster recovery

Cloud hosting

To understand a cloud hosting scenario, consider the requirements of two organizations, Woodgrove Bank and Contoso, Ltd., and how a cloud service provider named Fabrikam is able to fulfill these requirements using HNV. Woodgrove Bank wants to deploy a two-tier mobile application. The first tier consists of a web front-end while the second tier consists of multiple compute-intensive backend application servers that service the mobile application. The number of second tier servers required depends on the number of client connections at any given point in time. Given the dynamic nature of the client connections, in a traditional environment, Woodgrove Bank would have to deploy enough redundant servers to cater to peak demand. The problem with this approach is that during non-peak hours the deployed servers remain idle, accruing avoidable capital expenditure (CapEx) and operational expenditure (OpEx). But if Woodgrove Bank deploys the application on virtual machines (VMs) hosted on the Fabrikam infrastructure, they avoid the CapEx and OpEx of idle servers since they pay only for the actual usage of the servers.

Now consider the networking requirements of Woodgrove Bank's deployment in the Fabrikam network. Woodgrove Bank's mobile application needs to be reachable over the Internet, so they need at least one public IP address to which initial connections can be established. For the rest of their VMs, Woodgrove Bank chooses to assign IP addresses from the private IP address range (10.0.0.0/24). Fabrikam has to design their infrastructure in a way that not only meets the requirements of Woodgrove Bank but also the similar needs of other customers. Here’s are the different options Fabrikam has for doing this and the disadvantages of each approach:

- **Isolated physical networks** Fabrikam creates a subnet 10.0.0.0/24 in their physical network for Woodgrove Bank and connects Woodgrove Bank’s VMs to this subnet. The advantage of this approach is that Fabrikam can offer “bring your own IP or subnet” cloud services to multiple customers with overlapping IP address ranges. The disadvantage is that another customer, such as Contoso, cannot be allocated the same subnet (10.0.0.0/24). As a result, Fabrikam must create a separate physical network for each tenant. This limits the flexibility Fabrikam can offer to customers and impedes business scalability.
- **Separate VLAN per tenant** Fabrikam creates a separate VLAN for Woodgrove Bank and creates the subnet 10.0.0.0/24 in the Woodgrove Bank VLAN. There are several disadvantages of this approach:
  - On-boarding of each tenant requires VLAN configuration of routers, switches, and hosts in the Fabrikam network. Because of the lack of a common interoperable protocol for configuring switches and routers from various vendors, Fabrikam will have to modify their automation configuration for every version of software or hardware that is deployed in their infrastructure. If Fabrikam chooses routers/switches from a single vendor to avoid this, they face the disadvantages caused by single vendor lock-in.
  - Movement of VMs from one host to another or one subnet to another requires reconfiguration of routers/switches.
  - Fabrikam's infrastructure cannot scale beyond 4,096 virtual networks due to the VLAN identifier field length in the header of Ethernet frames.

As described in the previous chapter, HNV solves these problems by enabling service providers like Fabrikam to deploy network services in software on top of their existing legacy network infrastructure. Fabrikam can then optimally utilize their physical infrastructure by deploying the VMs of customers like Woodgrove Bank and Contoso with overlapping IP addresses on the same hosts over the same physical network. The advantage of this approach is that Fabrikam can deploy the solution entirely in software, avoiding initial CapEx associated with hardware-based solutions.

As Figure 2-1 illustrates, Woodgrove Bank's VMs running the backend services of the mobile application must communicate with client applications on mobile devices in the Internet. Since Woodgrove Bank's VMs are assigned private IP addresses (10.0.0.0/24), packets from the VMs cannot be sent directly over Internet. The packets must be translated by network address translation (NAT) to public IP addresses when they are sent from a cloud service provider network. Similarly, incoming packets on the public IP address of Woodgrove Bank must be translated by NAT before they are forwarded to VMs in Woodgrove Bank's virtual network. So all the packets transiting the Internet and Woodgrove Bank's virtual network have to be translated by NAT. To do this, Fabrikam must deploy a gateway capable of NAT at the edge of Woodgrove Bank's virtual network. Because the same requirement is needed for other tenants such as Contoso, Fabrikam has to deploy NAT services for multiple tenant virtual networks. Fabrikam has two options for doing this: deploy one NAT VM per tenant, as shown in Figure 2-1, or deploy a single NAT VM for all the tenants. Obviously, deploying a single NAT for all tenants reduces the CapEx and Opex for Fabrikam. Since the IP address space of tenants can overlap, the single NAT gateway must be aware of the tenants while translating packets in both directions.

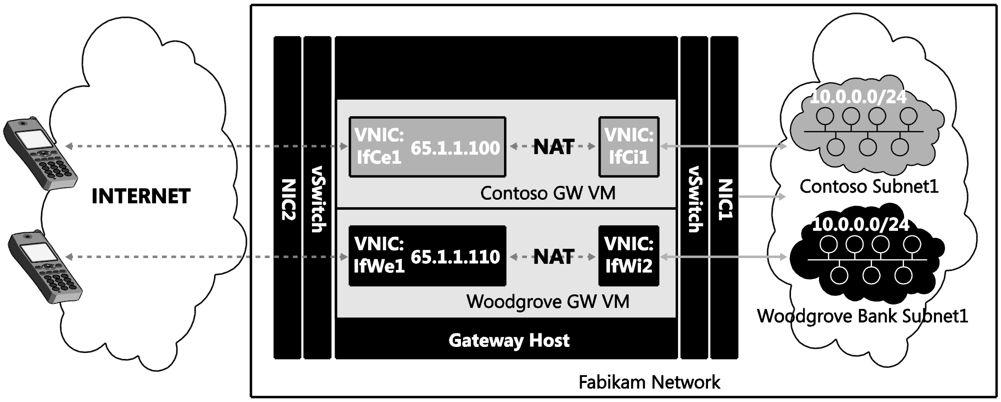


FIGURE 2-1 Separate gateway for each tenant.

Cloud bursting

Cloud bursting can be realized efficiently using HNV and Windows Server 2012 R2 gateway features. For example, Woodgrove Bank's payroll application is a two-tier application consisting of a web front-end and SQL backend servers. Woodgrove Bank currently deploys five front-end servers to take care of their peak load during the start and end of the month. During non-peak days, these servers run at less than 20 percent of their capacity. To avoid wasting resources due to underutilization of these front-end servers during non-peak days, Woodgrove Bank would like to host just one front-end server on-premises and deploy an additional four front-end servers as VMs running on the Fabrikam network. Since Woodgrove Bank has to pay for the four front-end servers in the Fabrikam network only four or five days a month, they can save on the overall cost of deploying their payroll application.

Woodgrove Bank prefers to deploy their SQL backend servers on-premises due to the privacy and confidentiality of their data. But when the four web front-end servers are deployed on the Fabrikam network, they will need to securely access SQL data on Woodgrove Bank's premises. To accomplish this, connectivity between Woodgrove Bank's VMs on the Fabrikam network and the on-premises SQL servers will be provided by a site-to-site (S2S) virtual private network (VPN) tunnel. Site-to-site VPN allows Woodgrove Bank's virtual network in Fabrikam to be seen as an extension of Woodgrove Bank's on-premises network. This communication is enabled by S2S VPN between Woodgrove Bank's edge network and the Fabrikam network. Fabrikam deploys an S2S VPN server at their cloud service provider's edge network to enable Woodgrove Bank to connect from their premises networks to their virtual network in Fabrikam.

As shown in Figure 2-2, Fabrikam prefers deploying S2S VPN for multiple tenants on a shared gateway to bring down the cost of their gateway infrastructure. To simplify route configuration, Fabrikam deploys a single gateway for both NAT and S2S VPN.

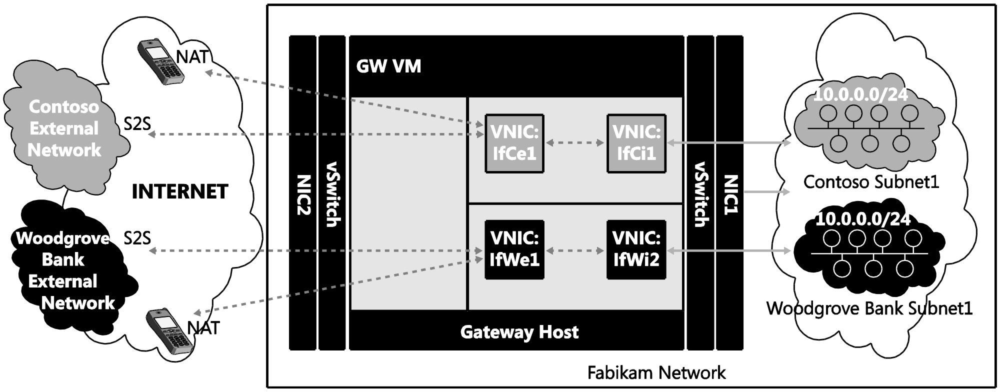


FIGURE 2-2 Gateway VM shared across multiple tunnels.

Cloud-based backup and recovery

Cloud services providers like Fabrikam can deploy cloud-based backup and recovery services for enterprises. For example, Contoso is a business with rented front office space in a commercial building. Contoso maintains their inventory management system on a single server in their office space. To ensure continued availability of the VMs that host their inventory system in the event of any disruption, Contoso replicates their VMs to the Fabrikam network using a technology like Hyper-V Replica in Windows Server 2012. Figure 2-3 shows how Fabrikam deploys their backup and recovery service using the same infrastructure used to deploy the cloud hosting and cloud bursting scenarios.

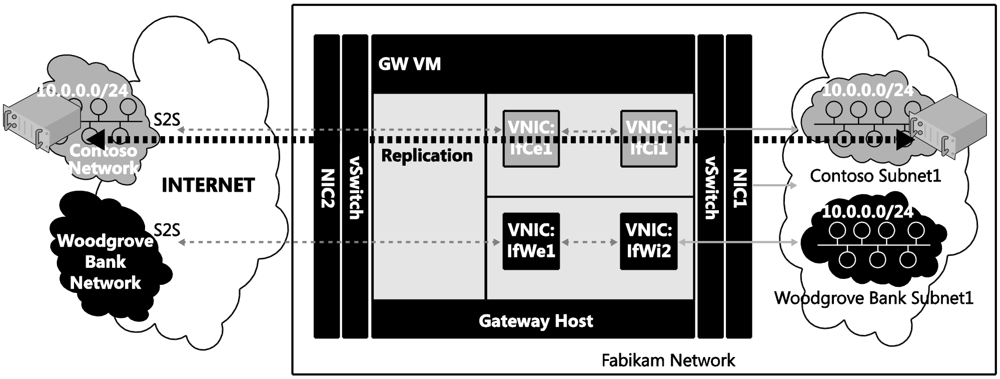


FIGURE 2-3 Shared gateway for shared services.

In this diagram, Contoso replicates business-critical VMs to the Fabrikam network. Replication traffic can then be sent to the Fabrikam network either over S2S VPN or directly over the Internet and be translated by NAT at the Fabrikam edge. In the event of unavailability of their office premises servers due to power or network failure, Contoso employees can access their VMs in the cloud service provider network using VPN. Fabrikam needs to provide such VPN services to all of their tenants, from those who bring in a single VM to those who bring in hundreds of VMs. Since multiple customers might bring in VMs with overlapping private IP addresses, the service provider deploys HNV for the same reasons mentioned earlier in the cloud bursting scenario: it is more cost-effective for Fabrikam to deploy a single VPN server to serve multiple customers with overlapping IP addresses than to deploy a separate VPN gateway for each customer. As shows in Figure 2-4, the new multi-tenancy capabilities of the Remote Access role services in Windows Server 2012 R2 helps cloud service providers to cost-effectively provide connectivity to VMs (in this case as part of cloud-based backup and recovery service) by enabling sharing of a gateway across multiple tenants.

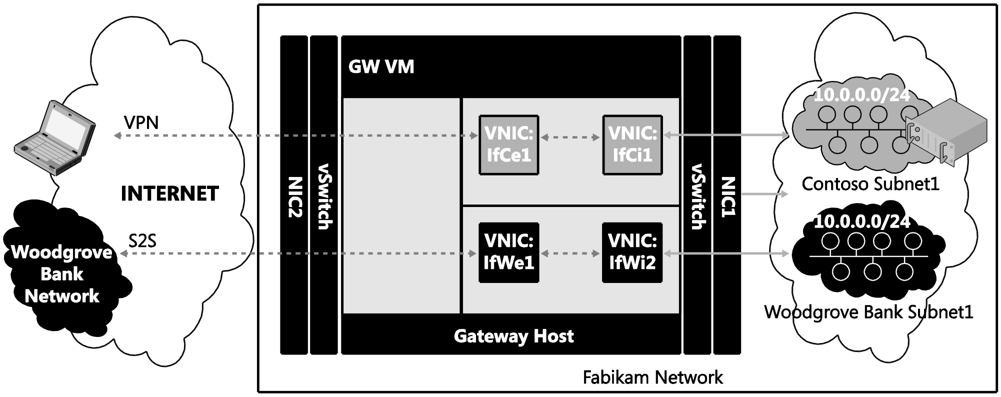


FIGURE 2-4 Single gateway from multiple services and multiple tenants.

HNV gateway

To provide connectivity from a tenant's virtual network to any external network such as Intranet or Internet or a tenant's corporate network, a gateway (GW) virtual machine is required. The gateway VM should generally have at least one interface connected to the virtual network and at least one more interface connected to the external network. For example, in Figure 2-5, a gateway VM dedicated to Contoso connects the Contoso virtual network in Fabrikam to external networks. Specifically, interface IfCi1 connects to the Contoso virtual network in Fabrikam while interface IfCe1 connects to the Contoso external network. When packets are routed from the Contoso external network to the Contoso subnet 10.0.0.0/24 on the Fabrikam network, they are forwarded to interface IfCe1. The TCP/IP stack on the gateway VM routes the packets to interface IfCi1, which delivers the packets inside the Contoso subnet 10.0.0.0/24. Similarly, when packets are routed from subnet 10.0.0.0/24 to the Contoso external networks, they are forwarded to interface IfCi1 and the TCP/IP stack on the GW VM routes the packets to interface IfCe1, which then delivers them to the Contoso external networks. Similarly, another gateway VM dedicated to Woodgrove Bank connects the Woodgrove Bank virtual network in Fabrikam to external networks.

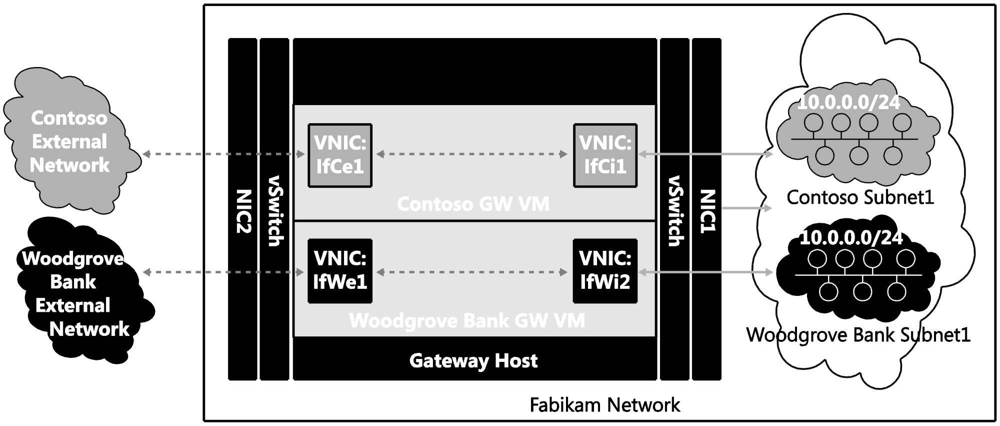


FIGURE 2-5 Gateway VM per tenant.

The problem with this approach is that the number of gateways increases linearly as the number of tenants grows. To ensure high availability, the cloud service provider would have to deploy two gateway VMs per tenant. Fabrikam would prefer deploying a gateway that could be shared across multiple tenants similar to deploying multiple tenant virtual networks over a shared physical network using HNV. In the sections that follow you will see how Fabrikam can reduce their gateway infrastructure costs by enabling isolation of routing and connectivity on a shared multi-tenant gateway using Windows Server 2012 R2.

Multi-tenant TCP/IP stack

With the new multi-tenant TCP/IP stack in Windows Server 2012 R2, a single VM can be used to route packets in a compartmentalized manner exclusively between tenants' interfaces, for example between Contoso interfaces (IfCe1 and IfCi1) for Contoso traffic and between Woodgrove Bank interfaces (IfWe1 and IfWi1) for Woodgrove Bank traffic. This is made possible by creating separate routing compartments for Contoso and Woodgrove Bank. Routing compartments virtualize the TCP/IP stack to enable multiple routing entities with overlapping IP addresses to co-exist on the same gateway VM. Packets and network entities including interfaces, IP addresses, route tables, and ARP entries of each tenant are isolated by routing compartments. Interfaces and packets in one compartment cannot be seen in another compartment. Isolation of multi-tenant traffic when it enters or leaves the shared gateway VM is accomplished by extending the routing domain concept of HNV to the gateway VM. Windows Server 2012 R2 thus provides end-to-end network virtualization across all network layers of the TCP/IP stack above the physical layer.

Having a separate routing compartment per routing domain on a gateway VM enables connectivity between the virtual and external networks of multiple tenants. For example, in Figure 2-6 there are three compartments in the gateway VM (GW-VM) namely the Default (compartment id 1), Contoso, and Woodgrove Bank compartments. By default all network entities, such as interface If1, are created in compartment id 1, hence the name Default compartment. In this example, the interfaces IfCe1 and IfCi1 in the Contoso compartment are exclusive to Contoso, but their IP addresses and routes can overlap with the interfaces in any other tenant compartment. The multi-tenant TCP/IP stack in GW-VM routes packets between IfCi1 and IfCe1 as if only these two interfaces exist for the routing domain, which provides isolation for all Contoso packets. Interface IfCi1 connects the Contoso compartment in GW-VM to the Contoso virtual network on the Fabrikam network. As shown in the figure, the gateway runs as a VM and the host's physical network interface card NIC1 is connected to the overlay network. So the packets of all tenants hit NIC1, and they are transmitted through interface IfCi1 to the Contoso compartment of the gateway VM by the Hyper-V virtual switch (vSwitch) connected to NIC1.

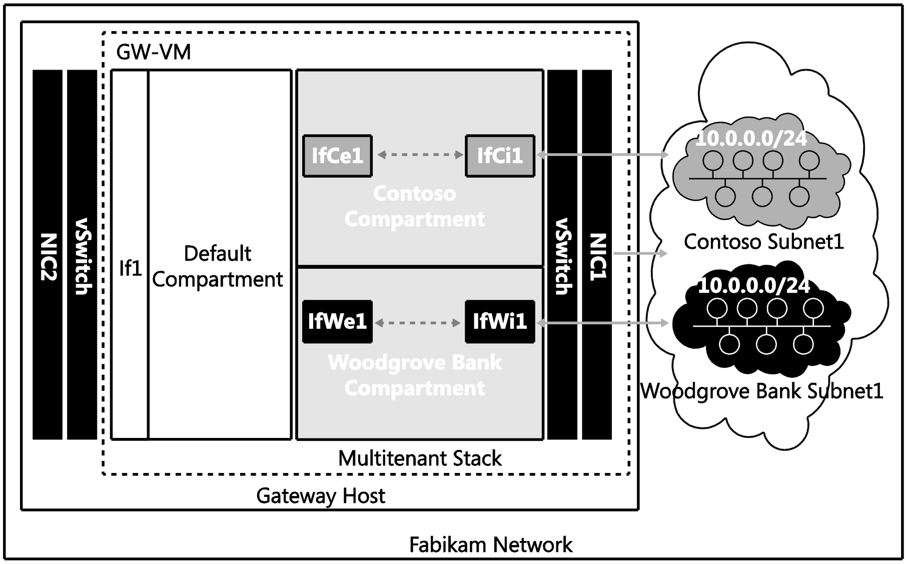


FIGURE 2-6 Multi-tenant gateway with compartment per tenant.

You can use Windows PowerShell to configure multi-tenancy, for example on GW-VM for Contoso. The following cmdlet enables multi-tenancy on NIC1:

```
Set-VmNetworkAdapterIsolation -VMName GW-VM -VMNetworkAdapterName NIC1
   -MultiTenantStack on -IsolationMode NativeVirtualSubnet
```

To ensure that that all Network Virtualization using Generic Routing Encapsulation (NVGRE) packets in the Contoso routing domain are transmitted to the Contoso interface IfCi1, the following configuration step is required on the host:

```
Add-VmNetworkAdapterRoutingDomainMapping -VMName GW-VM -VMNetworkAdapterName NIC1
  -RoutingDomainId "{12345678-1000-2000-3000-123456780001}" -RoutingDomainName "Contoso"
  -IsolationId 6001 -IsolationName "IfCi1"
```

After the above cmdlets are executed, a compartment for Contoso is created on GW-VM. This can be verified by executing the following cmdlets on GW-VM:

```
PS C:\> Get-NetCompartment
CompartmentId          : 1
CompartmentDescription : Default Compartment
CompartmentGuid        : {xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}
CompartmentId          : 2
CompartmentDescription : Contoso
CompartmentGuid        : {12345678-1000-2000-3000-123456780001}
```

The following cmdlet sets the IP address on interface IfCi1 to 10.0.254.2:

```
New-NetIpAddress –InterfaceAlias IfCi1 –AddressFamily IPv4 –Ipaddress 10.0.254.2
```

```
–PrefixLength 29
```

After this configuration, all packets in the Contoso virtual network that are forwarded to 10.0.254.2 are delivered to the Contoso compartment via interface IfCi1 by the vSwitch on the gateway host. Since interface IfCi1 is created in the Contoso compartment, these packets can be routed to any other interface, such as IfCe1, in the same compartment. In a similar configuration for the other tenant Woodgrove Bank, the interface IfWi1 would be created in the Woodgrove Bank compartment with the same IP address, 10.0.254.2. The traffic for the Contoso routing domain will then be injected into the Contoso compartment in the gateway VM via interface IfCi1, and the traffic of the Woodgrove Bank routing domain will be injected into the Woodgrove Bank compartment in the gateway VM via interface IfWi1.

Packets in the Woodgrove Bank compartment can be routed over S2S VPN to the premises of Woodgrove Bank or to the devices of Woodgrove Bank employees via remote access VPN to the Internet through NAT. Compartments also enable routing between virtual and physical networks using the forwarding gateway feature in Windows Server 2012 R2.

Border Gateway Protocol (BGP) can also be enabled on interfaces IfCi1 and IfWi1 using Windows Server 2012 R2 Routing and Remote Access Service (RRAS) even though they have same IP address. There are two BGP listeners on port 179 with the same IP address, 10.0.254.2, in two different compartments on GW-VM. The two listeners can learn overlapping routes (e.g., 10.1.1.0/24) from their respective peers using multi-tenant routing in Windows Server 2012 R2.

Multi-tenant S2S VPN gateway

Windows Server 2012 R2 can provide two modes of VPN connectivity:

- Site-to-site (S2S) VPN tunnels between tenant sites and tenant virtual networks
- Remote access VPN tunnels from devices on the Internet to tenant virtual networks

For example, S2S VPN connectivity can be enabled from the office sites of Woodgrove Bank and Contoso to their respective virtual networks using a single Windows Server 2012 R2 gateway VM named GW-VM. Figure 2-7 depicts a deployment with two tenants, Woodgrove Bank and Contoso, connected to their respective virtual networks over S2S VPN from their premises. Woodgrove Bank has two sites, one each in New York (NY) and San Francisco (SFO), with internal subnets 10.1.0.0/24 and 10.2.0.0/24, respectively. From both of these sites, Woodgrove Bank connects to its virtual network 10.0.0.0/24 in Fabrikam through GW-VM via S2S VPN. Contoso connects its internal network 10.1.0.0/24 in New York to its virtual network 10.0.0.0/24 in Fabrikam through the same GW-VM via S2S VPN. Both tenants dial to the same interface If2 (public IP address 131.107.10.10), which is on the Internet.

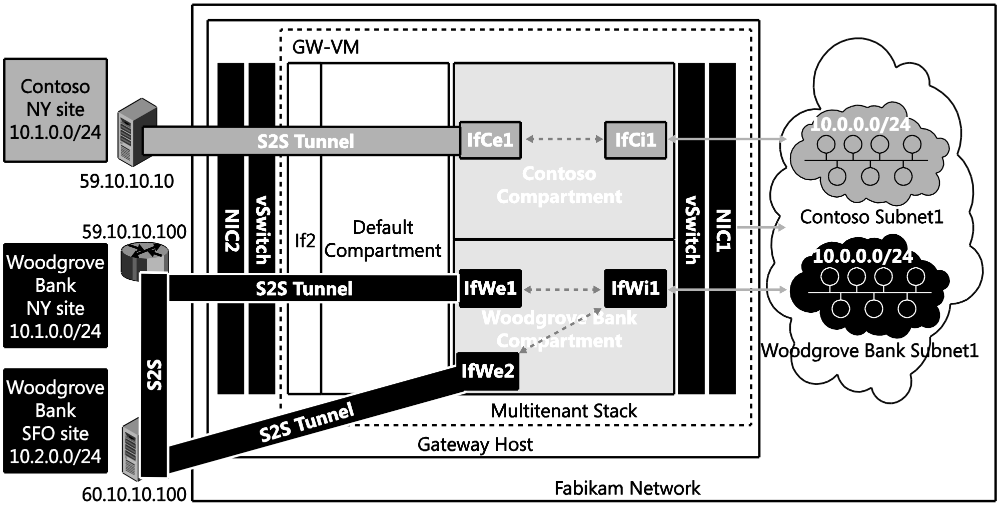


FIGURE 2-7 Multi-tenant S2S VPN.

To enable multi-tenancy for remote access on GW-VM and to enable S2S VPN for the Contoso and Woodgrove Bank tenants, the following cmdlets need to be executed on GW-VM:

```
Add-WindowsFeature -Name Remote Access -IncludeAllSubFeature
   -IncludeManagementTools
Install-Remote Access –MultiTenancy
Enable-Remote AccessRoutingDomain  -Name Woodgrove –Type All
Enable-Remote AccessRoutingDomain  -Name Contoso –Type All
```

To enable S2S connectivity from the NY and SFO sites of Woodgrove Bank, the S2S VPN interfaces IfWe1 and IfWe2 can be configured using the following two cmdlets. Note that the RoutingDomain parameter defines the association between the tenant and the S2S VPN interface:

```
Add-VpnS2SInterface –RoutingDomain Woodgrove –Name IfWe1
   –Destination 59.10.10.100 -Protocol IKEv2 -AuthenticationMethod PSKOnly
   -SharedSecret abc -IPv4Subnet 10.1.0.0/24:100 -SourceIpAddress 131.107.10.10
Add-VpnS2SInterface –RoutingDomain Woodgrove –Name IfWe2
   –Destination 60.10.10.100 -Protocol IKEv2 -AuthenticationMethod PSKOnly
   -SharedSecret abc -IPv4Subnet 10.2.0.0/24:100 -SourceIpAddress 131.107.10.10
```

The following cmdlet configures the S2S VPN interface on GW-VM for tenant Contoso to connect from the NY site:

```
Add-VpnS2SInterface –RoutingDomain Contoso –Name IfCe1 –Destination 59.10.10.10
   -Protocol IKEv2 -AuthenticationMethod PSKOnly -SharedSecret abc
   -IPv4Subnet 10.1.0.0/24:100 –SourceIpddress 131.107.10.10
```

The previous series of cmdlets configure the S2S VPN virtual interfaces IfWe1, IfWe2, and IfCe1 on the gateway. These virtual interfaces are created in the Contoso or Woodgrove Bank compartment only after tunnel establishment.

Examining the lifecycle of the S2S VPN interface IfCe1 can help you better understand how the multi-tenancy of S2S VPN works. When the S2S VPN interface IfCe1 is configured using the above cmdlet, the Windows Server 2012 R2 Remote Access service plumbs the necessary IPsec policies on GW-VM to allow IPsec/IKEv2 (Internet Key Exchange version 2) tunnel connection from 59.10.10.10 to interface If1 (131.107.10.10) on GW-VM. Similarly, when IfWe1 and IfWe2 are configured, IPsec policies are plumbed to allow IPsec/IKEv2 tunnels from 59.10.10.100 and 60.10.10.100 to interface If1 (131.107.10.10) on GW-VM.

When an IPsec/IKEv2 S2S VPN connection is dialed from 59.10.10.10 to 131.107.10.10 (the IP address of interface If2 in GW-VM), it matches the IPsec policy configured for the S2S VPN interface IfCe1 on GW-VM. After IKE pre-shared key (PSK) authentication, the Remote Access service on GW-VM searches the list of S2S interfaces to find one with the matching destination IP address 59.10.10.10 and finds interface IfCe1. When the interface IfCe1 is identified, it is created in the Contoso compartment based on the RoutingDomain value for Contoso in the interface IfCe1 configuration. This is how the tenant is identified for an incoming S2S connection with PSK authentication.

After interface IfCe1 is created in the Contoso compartment in addition to the already existing interface IfCi1 (note that interface IfCi1 connects GW-VM to the virtual network of Contoso), packets can be routed between interfaces IfCi1 and IfCe1. The S2S interface IfCe1 connects the Contoso NY site to the gateway. Thus the Contoso virtual network in Fabrikam is connected over S2S VPN to Contoso NY site via S2S VPN. If the S2S interface IfCe1 is dialed out from the gateway, the Remote Access service creates interface IfCe1 in the Contoso compartment based on the RoutingDomain value of IfCe1 configuration.

Similarly, the S2S interfaces IfWe1 and IfWe2 are created in the Woodgrove Bank compartment, enabling the routing of packets between the Woodgrove Bank virtual network and the Woodgrove Bank sites in NY and SFO through interfaces IfWe1 and IfWe2, respectively. Thus the routing of packets between multiple tenants' sites and their respective virtual networks in Fabrikam is enabled on a shared multi-tenant S2S VPN gateway.

Authentication of S2S VPN

Authentication plays a key role in identifying the S2S interface for an incoming connection and for the subsequent determination of the compartment or tenant. S2S VPN connections on Fabrikam GW-VM can be authenticated using one of the following methods

- Pre-Shared Key (PSK)
- X.501 Certificates
- Extensible Authentication Protocol (EAP)

When an S2S VPN interface is dialed from the server, it is called an initiator; when the server accepts an incoming connection, it is called a responder. IKE/IPsec polices for S2S VPN can be specified at the server (gateway) level or at the interface level.

For outgoing connections (initiator), the gateway uses policies and parameters specified at the interface level. For non-PSK based authentication, interface level IPsec policies are plumbed only while dialing out the connection. In the case of PSK-based authentication, the IPsec policies corresponding to the S2S interface become effective immediately after configuration.

For incoming connections (responder), the gateway has a list of interfaces and needs to activate (create a tunnel for) a specific interface by registering with the TCP/IP stack. It can determine the specific interface for which the connection is being requested only after authentication. Hence the basis on which an incoming connection is identified is based on the authentication method.

If the authentication method is username, the S2S interface is identified based on the username that is being authenticated, and the S2S interface whose name matches the authenticated username is activated. If there is no matching S2S interface name, the incoming connection is treated as a remote access (dial-in) VPN connection.

If the authentication method is certificates, then the S2S interface is identified based on the subject name of the certificate being authenticated. The S2S interface whose name matches the certificate subject name is then activated (see *http://technet.microsoft.com/en-US/library/dd469750.aspx*).

If the authentication method is PSK, the S2S interface is identified based on the source IP address of the incoming IKE packets and is matched with the destination IP address of the configured S2S interfaces.

Since authentication in IKE happens after initial IPsec negotiation is complete, all incoming connections are governed by server level policies except for PSK-based authentication. In the case of PSK-based authentication, since identification of the tunnel is based on IP address, only one interface can be enabled per destination, and the initiator and responder policies are governed by S2S interface level policies. Also, if the authentication method is PSK, only IPv4 or IPv6 addresses can be specified as the destination. For other authentication methods, any resolvable name can be specified as the destination of the S2S interface.

MORE INFO **Detailed steps for configuring S2S VPN using pre-shared key authentication are outlined at *http://technet.microsoft.com/en-us/library/jj574210.aspx*. If X.501 certificates are used to authenticate S2S connections, all the incoming connections must chain up to the root certificate of Fabrikam. Detailed steps for using X.501 certificates for authenticating S2S VPN connections are outlined at *http://technet.microsoft.com/en-us/library/jj574149.aspx*. S2S connections can also be authenticated using Extensible Authentication Protocol. Please refer to *http://technet.microsoft.com/en-us/library/jj574129.aspx* for details on using EAP for S2S VPN authentication.**

Routing packets over S2S VPN interfaces

For connectivity to be established between applications in the customer site and the virtual network in the service provider network, it is important to have proper routes configured across S2S VPN tunnels. The following options are available to configure routes:

- Static routes
- Border Gateway Protocol (BGP)

Static routes

Static routes can be configured on S2S interfaces so that when packets arrive in the compartment of the tenant, they are routed over a specific S2S interface. In the Woodgrove Bank example, the routes 10.1.0.0/24 and 10.2.0.0/24 are added on the S2S interfaces IfWe1 and IfWe2, respectively, with a metric of 100. These routes trigger the S2S connection to be dialed (if it is disconnected) when packets that match the routes on the S2S interface end up in the Woodgrove Bank compartment. There is also a provision to add routes on an S2S interface only after the connection is established without triggering the connection by specifying "PostConnectionIPv4Subnet" using the following cmdlet:

```
Set-VpnS2SInterface –RoutingDomain Contoso –Name IfC1
   -PostConnectionIPv4Subnet 10.1.0.0/24:100
```

Route exchange via BGP

Routes can be exchanged between customer site gateways and the customer virtual network in the cloud service provider over S2S VPN via BGP. Details of BGP configuration and various topologies are described later in the section titled "Routing between virtual networks and tenant sites."

For BGP connection establishment, host-specific routes (/32 in the case of IPv4 and /128 in the case of IPv6) of the BGP peer are added as trigger routes on the S2S VPN interface. This ensures that when BGP connection establishment is attempted, the S2S connection is dialed and the routes of the BGP peers are added on the S2S interface so that BGP packets can be tunneled over the S2S interface. Once BGP peering is established, the rest of the routes are exchanged by BGP peers running at either end.

In the Woodgrove Bank example, the BGP peer can be configured on interface IfWe1, IfWe2, or IfWi1. Since interface IfWi1 is always present and has a fixed IP address, it is recommended to configure BGP on IfWi1, which connects to the virtual network. Once BGP on IfWi1 establishes a connection with BGP peers in the NY and SFO sites, the routes 10.1.0.0/24 and 10.2.0.0/24 are learned and plumbed on the S2S interfaces IfWe2 and IfWi1, respectively, and no manual configuration of routes on S2S interfaces is needed.

Rate limiting of traffic on an S2S VPN interface

Data processing of S2S tunnels is compute-intensive because of the encryption and decryption that occurs. With a multi-tenant gateway, the same VM is used to serve multiple S2S tunnels of multiple tenants. Hence, if the rate at which each tunnel can send or receive data is not controlled, it is possible for the data of one tunnel to consume all available resources on the gateway, thus starving the other tunnels. It is therefore recommended that you rate-limit traffic on each S2S tunnel. The following cmdlet is an example where bandwidth is limited to 1,024 kilobits per second (kbps) in each direction:

```
Set-VpnS2SInterface –RoutingDomain Contoso –Name If2 -TxBandwidthKbps 1024
   -RxBandwidthKbps 1024
```

By default, all connections are created with a 5,120 kbps limit in each direction so that no single tunnel hogs the CPU.

Static IP filtering on an S2S VPN interface

On each of the S2S interfaces, static IP filters can be configured to allow or block packets based on the source IP address, destination IP address, source port, destination port, and protocol in each direction. You can set packet filters per interface and configure them to do one of the following:

- Pass through all traffic except packets prohibited by specified filters
- Discard all traffic except packets allowed by specified filters

For example, Contoso could limit traffic going out of the virtual network by adding outbound filters like this:

```
Add-Remote AccessIpFilter -InterfaceAlias IfCi1 -Action Deny -Direction Outbound
   -AddressFamily IPv4 -List @("10.0.0.0/24:10.1.0.0/24:any","any:any:ICMP:0:0")
```

After configuring filters using the previous cmdlet, all outgoing packets through interface IfCi1 are dropped except for those that match the following conditions:

- Source IP subnet is 10.0.0.0/24 and destination IP subnet is 10.1.0.0/24
- Any ICMP traffic

Contoso could also restrict incoming traffic into the virtual subnet 10.0.0.0/16 to SSL (TCP port 443) and Ping (IMCP request code 8) traffic using the following cmdlet:

```
Add-Remote AccessIpFilter -InterfaceAlias IfCi1 -Action Deny -Direction Inbound
   -AddressFamily IPv4 -List @("any:10.0.0.0/16:TCP:0:443"," any:any:ICMP:8:0")
```

As a final example, the following cmdlet allows all outgoing traffic through interface IfCi1 except TCP traffic from IP subnet 10.0.0.0/16 to 10.2.0.0/16 with source port 1234 and destination port 4321:

```
Add-Remote AccessIpFilter -InterfaceAlias IfCi1 -Action Allow
   -List @("10.0.0.0/16:10.2.0.0/16:TCP:1234:4321") -Direction Outbound
   -AddressFamily IPv4
```

Multi-tenant Remote Access VPN gateway

Hybrid networking enables the enterprise or small- to medium-sized business (SMB) to deploy their workloads in the cloud by hosting them in a cloud service provider's datacenter. The workloads can then be accessed from inside the corporate network via an S2S VPN. When employees need to access these workloads from outside the corporate network, they can connect to their corporate network via their VPN gateway and access the workloads via S2S VPN. If the VPN gateway is not available for some reason, the employees can still access the workloads by directly establishing a VPN connection with the cloud service provider's gateway.

Businesses that do not have their own remote access (dial-in) VPN infrastructure can allow their employees to connect via VPN to the cloud service provider's gateway and access workloads in the cloud or on the business premises via S2S VPN. Employees of businesses without infrastructure that have their entire workloads running in the cloud need to connect via VPN to the cloud service provider's gateway to access their internal workloads.

> ***NOTE*** **"Access the workloads" means using client applications like SQL client for querying a database server. Cloud service providers can also allow administrators to have access to the actual VMs for diagnostic and management purposes via Remote Desktop Protocol or similar mechanisms.**

The remote access (dial-in) VPN requirements for Woodgrove Bank and Contoso, Ltd. can be fulfilled by the cloud service provider Fabrikam by using the multi-tenancy capability of the Windows Server 2012 R2 Remote Access role. In Figure 2-8, a gateway VM dedicated to Contoso enables VPN connections from employees of Contoso so they can connect to the Contoso virtual network in Fabrikam. Interface IfCi1 connects to the Contoso virtual network in Fabrikam while the VPN clients connect over interface IfCe1. After a VPN connection has been established, packets to the Contoso virtual subnet 10.0.0.0/24 are sent to the Contoso GW VM on the Fabrikam network. The TCP/IP stack on the GW VM routes the packets to interface IfCi1, which then delivers the packets inside the Contoso subnet 10.0.0.0/24. In the opposite direction, when packets are routed from subnet 10.0.0.0/24 to a VPN client, they are forwarded to interface IfCi1, and the TCP/IP stack on GW VM routes the packets over a VPN tunnel via interface IfCe1. Similarly, another GW VM dedicated for Woodgrove Bank enables VPN clients to connect to the Woodgrove Bank virtual network in Fabrikam.

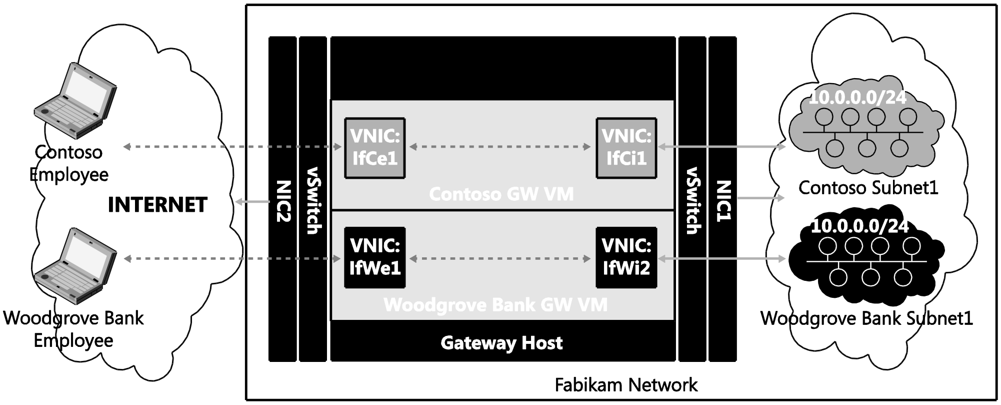


FIGURE 2-8 Separate gateway VM per tenant.

There are two problems with this approach:

- The number of gateways increases linearly as the number of tenants grows, and for high availability, the cloud service provider would have to deploy two gateway VMs per tenant.
- Each tenant requires at least one public IP address on the public interface of the GW VM. If multiple tenant gateways are deployed behind a NAT device that has only one public IP, the destination port to which the tenants connect should be different for different tenants. As a result, VPN tunnels that rely on SSL port 443 cannot be used by the tenants. If the port changes to some other random port, the packets may be dropped by firewalls in coffee shops or other access points from which tenants connect (the very reason why port 443 is used for Secure Socket Tunneling Protocol (SSTP) so that it can traverse firewalls).

Fabrikam would prefer deploying a gateway that could be shared across multiple tenants similar to deploying multiple tenant virtual networks over a shared physical network using HVN. Figure 2-9 depicts a deployment where employees of two tenants, Woodgrove Bank and Contoso, connect to their respective virtual networks over a VPN from the Internet. Note that the VPN connections for both tenants dial to the same interface If2 (IP address 131.107.10.10), which is connected to Internet.

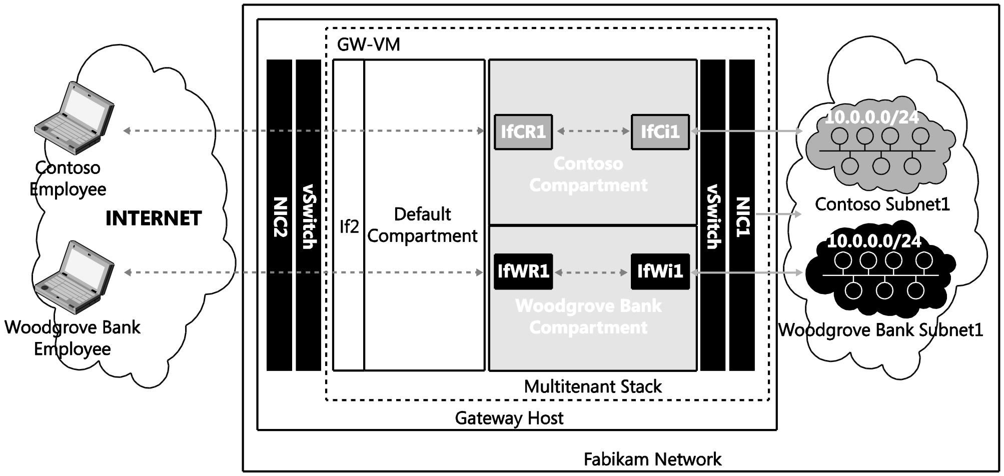


FIGURE 2-9 Multi-tenant remote access VPN.

Before examining how this multi-tenant VPN works, you must first understand in detail how Remote Access VPN works. When the first VPN client of a tenant dials in to the public interface of the VPN server, a new "RAS dial-in interface" is created on the VPN server. After connecting to the VPN server, the VPN client is assigned an IP address from the address pool configured on the VPN server. The traffic to and from the IP addresses in this pool is routable on the internal network and allows the VPN client to access resources behind the VPN server. For each VPN client, a sub-interface is created on the RAS dial-in interface and a /32 route with the IP address assigned to the VPN client is added on the sub-interface. The IP address assigned to the client identifies the specific sub-interface on which data is routed to a specific VPN client.

As explained in the section titled "Multi-tenant TCP/IP stack" earlier in this chapter, each compartment in the multi-tenant GW-VM isolates the traffic meant for each routing domain. So when a Contoso employee dials a VPN connection to the public interface of the cloud service provider's gateway VPN server, the multi-tenant aware Remote Access VPN server validates the user's credentials and identifies the corresponding tenant. It then creates a new sub-interface of the RAS dial-in interface in the Contoso compartment and assigns the client an IP address as configured in the Contoso compartment's routing domain. A VPN tunnel is then created between the client and the newly created RAS dial-in interface in the Contoso compartment.

When the RAS dial-in interface and the VPN tunnel have been created, the data packets received from the VPN client end up in the Contoso compartment. Inside the Contoso compartment, packets are routed based on their destination IP address in the usual way. If the destination address is part of the Contoso virtual network 10.0.0.0/24, the packets are forwarded to the Contoso virtual network using the VSID associated with Contoso. Similarly, when traffic in the opposite direction meant for the client reaches the internal vSwitch of Fabrikam's cloud gateway, it injects the packets in the Contoso compartment based on VSID.

Once a packet lands in the Contoso compartment, the /32 route on the dial-in interface causes the packet to be routed on the dial-in interface that delivers packets to the corresponding VPN tunnel.

Authentication of VPN clients

Since a single multi-tenant VPN gateway can be used by the cloud service provider for multiple tenants, the gateway has to authenticate incoming VPN connections as well as identify the tenant. The Remote Access VPN server supports two modes of tenant identification.

- Tenant identification based on username
- Tenant identification based on RADIUS ClassID attribute

The tenant to which an incoming VPN connection belongs is identified by the VPN server based on the credentials, and when authentication is complete, the tunnel is created in the identified tenant compartment.

Tenant identification based on username

In this approach, the tenant is identified based on the username in the credentials the VPN client supplies as part of the authentication process. For this to work, the Tenant Name parameter has to be configured per routing domain on the multi-tenant VPN gateway. Tenant Name can be any user-defined name or regular expression (e.g., “Contoso” or “*Contoso*”) to identify the tenant and routing domain, but it has to be unique across routing domains on a given multi-tenant VPN server.

The Set-RemoteAccessRoutingDomain Windows PowerShell cmdlet can be used to configure the Tenant Name parameter as follows:

```
Set-RemoteAccessRoutingDomain –Name "Contoso" –TenantName ".*Contoso.*"
Set-RemoteAccessRoutingDomain –Name "Woodgrove" –TenantName ".*Woodgrove.*"
```

For example, an employee John Doe from Contoso, Ltd. can dial a VPN connection using one of the following values for username: JohnDoe@Contoso.Fabrikam.com or Contoso\JohnDoe. On parsing this username, the multi-tenant VPN server tries to match the “Contoso” string with the configured set of tenant names on the server. If there is a match, the server creates the VPN tunnel in the RoutingDomain compartment of the associated TenantName after authentication. In this example, the associated RoutingDomain is Contoso.

Actual authentication of VPN clients for all of the tenants can be done by the VPN server or by a remote RADIUS server. Where authentication is done by the VPN server, Fabrikam should deploy the necessary authentication infrastructure so VPN clients of all tenants can be authenticated by the server. Fabrikam could implement such an authentication infrastructure as follows:

1. Fabrikam deploys a domain controller for the Fabrikam parent domain.

2. Fabrikam deploys two domain controllers for the child domains Contoso and Woodgrove Bank.
3. The multi-tenant VPN server is joined to the Fabrikam domain.

Where authentication is performed by a RADIUS server, the VPN server needs to have a RADIUS server configured for client authentication. This can be done by using an external server as RADIUS or by configuring the Windows Network Policy Server role on the VPN server (see *http://technet.microsoft.com/en-us/library/cc731108.aspx*).

Tenant identification based on the RADIUS ClassId attribute

Another tenant identification option available is using the RADIUS class attribute (see *http://technet.microsoft.com/en-us/library/dd197472(v=ws.10).aspx*) to identify the tenant. In this approach, the VPN server relies on a RADIUS server to complete the authentication and return the tenant information using the RADIUS ClassID (value 25) attribute. The RADIUS server must be configured to return the ClassId attribute in the format "TenantName=<string>" where <string> should be replaced with the value of TenantName as configured using the Set-RemoteAccessRoutingDomain cmdlet.

For example, Fabrikam could assign the following usernames to its tenants:

- JoeWoodgrove@Fabrikam.com or Fabrikam\JoeWoodgrove
- JoeContoso@Fabrikam.com or Fabrikam\JoeContoso

Fabrikam would then configure its RADIUS policies such that if the username has substring Woodgrove, the class attribute returned would be "TenantName=Woodgrove" while if the username it has substring Contoso, the class attribute returned would be "TenantName=Contoso." This approach gives Fabrikam flexibility in deploying its authentication servers and username combinations for its tenants' VPN clients.

When authentication is completed, the VPN client endpoint needs to be assigned an IPv4 address or IPv6 address. One of the following parameters therefore needs to be configured for every RoutingDomain on the remote access VPN server in a multi-tenant deployment:

- A valid IPv4 address range for the IP addresses to be assigned to VPN clients
- A valid IPv6 address prefix representing the pool of IP addresses to be assigned to VPN clients

As an example, the following cmdlets configure the same IPv4 and IPv6 address range for the Contoso and Woodgrove Bank tenants and cap the aggregate bandwidth of all VPN clients to 5,120 kbps:

```
Set-Remote AccessRoutingDomain -Name Contoso -IPAddressRange 11.11.11.1, 11.11.11.200
  -IPv6Prefix 2001:db8:a::/64 -EnableQoS Enabled -TxBandwidthKbps 5120
   -RxBandwidthKbps 5120
Set-Remote AccessRoutingDomain -Name Woodgrove -IPAddressRange 11.11.11.1, 11.11.11.200
  -IPv6Prefix 2001:db8:a::/64  -EnableQoS Enabled  -TxBandwidthKbps 5120
   -RxBandwidthKbps 5120
```

Note that the multi-tenancy of the remote access VPN server enables two clients of different tenants to be assigned the same IP address.

Routing between virtual networks and tenant sites

Border Gateway Protocol (BGP) can enable cloud service provider Fabrikam to manage routing between the virtual networks hosted in Fabrikam of the organizations Woodgrove Bank and Contoso, Ltd and the respective on-premises networks of these organizations. Figure 2-10, which reproduces Figure 2-7 from earlier in this chapter, depicts a deployment with two tenants, Woodgrove Bank and Contoso, connecting to their respective virtual networks over S2S VPN from their premises. Woodgrove Bank has two sites, one each in New York (NY) and San Francisco (SFO), with internal subnets 10.1.0.0/24 and 10.2.0.0/24 respectively. From both of these sites, Woodgrove Bank connects to its virtual network 10.0.0.0/24 in Fabrikam through a Windows Server 2012 R2 VM named GW-VM via S2S VPN. Contoso connects its internal network 10.1.0.0/24 in New York to its virtual network 10.0.0.0/24 in Fabrikam through the same GW-VM via S2S VPN.

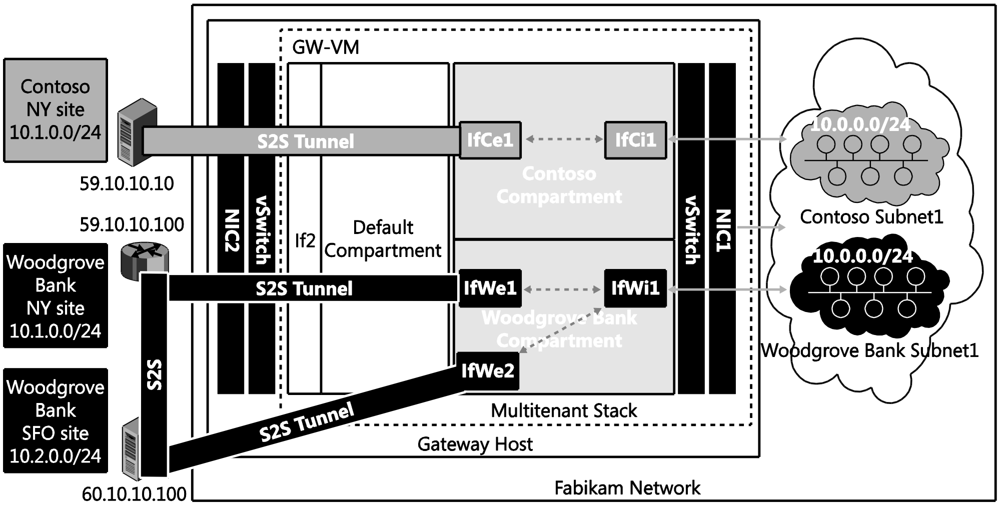


FIGURE 2-10 Connectivity between virtual networks and tenant sites.

To consider routing between the Woodgrove Bank virtual network and its SFO and NY sites, the following routes need to be configured in the Woodgrove Bank compartment:

- On interface IfWi1: 10.0.0.0/24
- On interface IfWe1: 10.1.0.0/24
- On interface IfWe2: 10.2.0.0/24

Whenever a new subnet (for example 10.3.0.0/24) is added in the NY site, the corresponding new route needs to be added on interface IfWe1 so that packets from the Woodgrove Bank virtual network can reach the NY site.

With the above specified routes, if the S2S VPN tunnel between the NY site and Fabrikam goes down, traffic cannot be sent to the NY site. There is a link between the NY and SFO sites, however, so traffic to the NY site could still be routed over SFO. This means route 10.1.0.0/24 needs to be added on interface IfWe2. When the S2S VPN tunnel between NY and Fabrikam is restored, traffic to NY will be split across two paths: directly to NY and via SFO. To restore routing so it uses the best path, the route 10.1.0.0/24 on IfWe2 needs to be removed after the tunnel between SFO and Fabrikam has been restored.

Adding and removing routes based on tunnel states like this requires manual intervention and results in sub-optimal routing that can lead to connectivity loss. One way to solve this problem is via route metrics (see *http://support.microsoft.com/kb/140859* for some background information on route metrics). The following routes with appropriate route metrics will solve the problem in this example:

- On interface IfWe1:
  - Route 10.1.0.0/24 with metric 100
  - Route 10.2.0.0/24 with metric 200
- On interface IfWe2:
  - Route 10.1.0.0/24 with metric 200
  - Route 10.2.0.0/24 with metric 100

With the above routes and route metrics configured, traffic to the NY site is normally routed over interface IfWe1 since the metric of route 10.1.0.0/24 is lower on interface IfWe1. Similarly, traffic to SFO is normally routed over interface IfWe2 since the metric of route 10.2.0.0/24 is lower on interface IfWe2. When the tunnel between the NY site and Fabrikam goes down, the S2S VPN interface IfWe1 also goes down and traffic to the NY site is routed over interface IfWe2 due to route 10.1.0.0/24 with metric 200. When the SFO tunnel is restored, interface IfWe1 comes back up along with route 10.1.0.0/24 with metric 100, optimally routing traffic to the NY site over interface IfWe1 again.

While static routes with metrics can work for a small set of sites and routes, the combination of sites, interfaces, and metrics quickly becomes unmanageable as the number of sites and routes increases. Clearly, as the number of customers (tenants) and their sites increases, the task of manually managing cloud service provider routes and ever-changing on-premises network routes becomes impossible. Similarly, on the enterprise side, keeping up-to-date route information for all of the subnets hosted in the cloud is a challenge in itself. This is where dynamic routing using a multi-tenant capable BGP router can ease manageability as well as provide faster convergence. The next section illustrates the concepts involved in detail.

Dynamic routing with BGP

With a dynamic routing protocol, routes can be exchanged between peers, obviating the need for adding static routes on each node along a path. Considering the Woodgrove Bank and Contoso scenario again, dynamic routing can play a role in enabling communication between an enterprise subnet and virtual networks in a cloud service provider.

Figure 2-11 shows the details of the interfaces on the edge routers in the NY and SFO sites. Interface IfNi1 is connected to subnet 10.1.0.0/24. An S2S VPN tunnel is established between interfaces IfFe1 on the SFO router to IfWe1 in the Woodgrove Bank compartment on the multi-tenant gateway GW-VM.

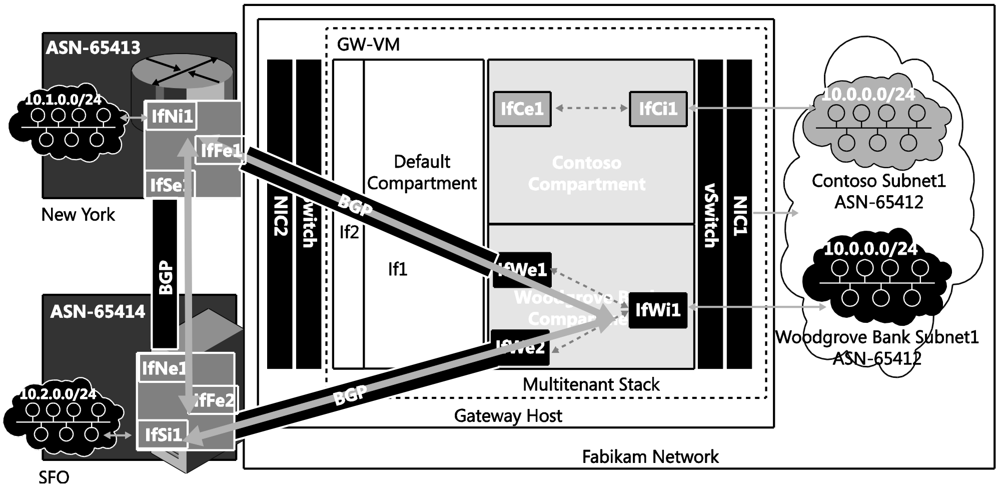


FIGURE 2-11 BGP routing and interface details.

If a dynamic routing protocol is configured between the router in NY and the router in the Woodgrove Bank compartment of GW-VM, then the route 10.0.0.0/24 on interface IfWi1 is propagated to the routing peer listening on the NY router, and the routing protocol adds the route 10.0.0.0/24 on interface IfFe1. Similarly, the route 10.1.0.0/24 is added on the interface IfWe1 in the Woodgrove Bank compartment on the multi-tenant gateway. The only manual configuration required for route exchanges is the configuration of the information required for the operation of the routing protocol. When this is done, routes are learned dynamically with no need for any manual intervention.

The rest of this section focuses on how one such routing protocol (BGP) can be used for dynamic route learning. For more information on BGP, see *http://www.bgp4.as*.

Examine in more detail the interfaces and routes shown in Figure 2-11. The router in the NY edge network has three interfaces of relevance:

- Interface IfNi1 (with IP address 10.1.0.1) connects to subnet 10.1.0.0/24.
- Interface IfSe1 connects to the SFO site via S2S VPN.
- Interface IfFe1 connects to the Woodgrove Bank virtual network 10.0.0.0/24 via S2S VPN.

Similarly, the router in the SFO edge network has three interfaces of relevance:

- Interface IfSi1 (with IP address 10.2.0.1) connects to subnet 10.2.0.0/24.
- Interface IfNe1 connects to the NY site via S2S VPN.
- Interface IfFe2 connects to the Woodgrove Bank virtual network 10.0.0.0/24 via S2S VPN.

In the Woodgrove Bank compartment on the multi-tenant gateway, there are three interfaces of relevance:

- Interface IfWi1 (with IP address 10.0.254.2) connects to the virtual network 10.0.0.0/24.
- Interface IfWe1 connects to the NY site via S2S VPN.
- Interface IfWe2 connects to the SFO site via S2S VPN.

While BGP can be enabled on all interfaces, in this example BGP is enabled only on interface IfWi1 in the Woodgrove Bank compartment, interface ifSi1 in the SFO site, and interface ifNi1 in the NY site. BGP uses the concept of autonomous system (AS) and AS number (ASN) for making routing decisions. This example assumes that each site is identified by a unique ASN in the private range 64512 to 65535. The subnet on the Fabrikam network will be identified by ASN 65412, and the NY and SFO sites will be assigned ASNs of 65413 and 65414, respectively.

The following cmdlet enables BGP in the Woodgrove Bank compartment with ASN 65412 and uses the IPv4 address of interface IfWi1 as the BGP identifier:

```
# Add a BGP Router for Woodgrove Tenant on the Cloud Service provider Gateway
Add-BgpRouter –RoutingDomain Woodgrove  –BgpIdentifier  10.0.254.2 –LocalASN 65412
```

The next step is to enable BGP peering for the NY and SFO sites:

```
# Add a BGP Peer for New York site of Woodgrove tenant
Add-BgpPeer –RoutingDomain Woodgrove  –Name  NY –LocalIPAddress 10.0.254.2
   -PeerIPAddress 10.1.0.1 –PeerASN 65413 -LocalASN 65412  -OperationMode Mixed
   -PeeringMode Automatic
# Add a BGP Peer for SFO site of Woodgrove tenant
Add-BgpPeer –RoutingDomain Woodgrove  –Name  SFO –LocalIPAddress 10.0.254.2
   -PeerIPAddress 10.2.0.1 –PeerASN 65414 -LocalASN 65412  -OperationMode Mixed
   -PeeringMode Automatic
```

With the previous cmdlets, BGP running in the Woodgrove Bank compartment uses the local IP address 10.0.254.2 and the ASN 65412 to peer with BGP in the NY and SFO sites with IP 10.1.0.1/ASN 65413 and IP 10.2.0.1/ASN 65414, respectively. With this configuration, BGP operates in a mode where it can initiate connections to peers as well as receive connections from peers automatically (this is the default mode of operation for the Windows BGP router). After similar configuration on the edge devices in the NY and SFO sites of Woodgrove Bank, the BGP peering configuration is completed.

> ***NOTE*** **The edge devices in the NY and SFO sites can be third-party routers. In that case, the configuration of these devices will vary from vendor to vendor.**

BGP peering to the NY site can be established only if an S2S VPN connection to IfWe1 is established. To trigger S2S VPN connections to IfWe1, a host-specific route for the remote BGP peer is added on the S2S VPN interface IfWe1 with the following cmdlet:

```
# Add a BGP triggering route in the VPN S2S Interface for Woodgrove tenant
Set-VpnS2SInterface –Name IfWe1 -IPv4Subnet 10.1.0.1/32:100
Set-VpnS2SInterface –Name IfWe2  -IPv4Subnet 10.2.0.1/32:100
```

When BGP in the Woodgrove Bank compartment tries to establish peering with 10.1.0.1 in the NY site, the S2S VPN interface IfWe1 is dialed. Once the S2S VPN interface IfWe1 is connected, BGP packets are tunneled and peering with BGP on the NY edge gateway is established. Similarly, BGP peering with edge gateway in SFO is established over the S2S VPN interface IfWe2.

After peering has been established, the following cmdlet can be used to enable advertisement of route 10.0.0.0/24 in the Woodgrove Bank compartment:

```
# Configure BGP Router for Woodgrove tenant with the custom routes
Add-BgpCustomRoute –RoutingDomain Woodgrove -Interface IfWe1
```

With the previous cmdlet, all of the routes on interface IfWe1 are advertised to BGP peers with ASN 65412. After similar BGP configuration on edge devices in NY and SFO, advertisements for routes 10.1.0.0/24 and 10.2.0.0/24 are received by BGP in the Woodgrove Bank compartment.

To understand how the routes are added on S2S interfaces, examine in detail the contents of the route advertisements received by BGP running in the Woodgrove Bank compartment. In the advertisement from the NY site, BGP running in the Woodgrove Bank compartment receives the following relevant information.

PREFIX	**AS-PATH**		**NEXT-HOP**
	Length	Sequence	
SITE – NY			
10.1.0.0/24	1	65413	10.1.0.1

Based on this information, BGP running in the Woodgrove Bank compartment tries to perform a recursive route lookup for 10.1.0.1 and determines that 10.1.0.1 is reachable via the S2S VPN interface IfWe1 based on the route added on the interface. It deduces that 10.1.0.0/24 is reachable via the S2S VPN interface IfWe1 and adds this route on interface IfWe1. After this route has been added, packets in the Woodgrove Bank compartment with destination matching 10.1.0.0/24 are routed over interface IfWe1. Similar route updates from the BGP peer in SFO result in the route 10.2.0.0/24 being added on the S2S VPN interface IfWe2.

Since the NY and SFO sites of Woodgrove Bank are connected, it is possible to configure third-party BGP routers at the NY edge network to advertise routes learned from SFO to the Woodgrove Bank virtual network in Fabrikam. Similarly a third-party router at the SFO edge network can be configured to advertise routes learned from NY to the virtual network in Fabrikam.

Next examine how BGP running in the Woodgrove Bank compartment handles these routes and makes it easy when compared to the static route configuration discussed earlier. BGP in the Woodgrove Bank compartment receives the following relevant information in update messages from its peers in NY and SFO:

PREFIX	**AS-PATH**		**NEXT-HOP**
	Length	Sequence	
SITE - NY			
10.1.0.0/24	1	65413	10.1.0.1
10.2.0.0/24	2	65414, 65413	10.1.0.1
SITE - SFO			
10.2.0.0/24	1	65414	10.2.0.1
10.1.0.0/24	2	65414, 65414	10.2.0.1

For routes 10.1.0.0/24 and 10.2.0.0/24, two routes are advertised, one with AS path length 2 and another with AS path length 1. BGP prefers routes with shortest AS path length and therefore routes packets to 10.1.0.0/24 via ASN 65413 (NY site) and to 10.2.0.0/24 via ASN 65414 (SFO site). If the S2S VPN tunnel between NY and Fabrikam goes down, BGP peering is also torn down (after a configured time-out) and all the routes learned from NY are discarded. BGP then re-computes the best path for 10.1.0.0/24 and 10.2.0.0/24. It finds that the best path for 10.2.0.0/24 has not changed and retains the route via SFO. It finds that the old path to 10.1.0.0/24 via NY no longer exists and determines the new best path to 10.1.0.0/24 is via SFO, so it adds the route 10.1.0.0/24 on the S2S VPN interface IfWe2 so that packets are routed to NY via SFO. Note that re-routing is taken care of by BGP without any manual intervention and all this happens in a matter of seconds.

The above explanation describes the routing details of tenant Woodgrove Bank on a multi-tenant service provider's edge gateway. As explained previously, Windows Server 2012 R2 can be deployed for routing of traffic of multiple tenants with overlapping IP addresses. In the example, the Fabrikam edge gateway is configured with two tenants (Contoso and Woodgrove Bank) that have an overlapping IP address space. Both of these tenants can enable BGP routing with overlapping ASNs and prefixes. Isolation of routing information is maintained by storing routes using Windows Route Table Manager v2 (RTMv2) (see *http://msdn.microsoft.com/en-us/library/windows/desktop/aa373798(v=vs.85).aspx*). This enables the configuration of multiple virtual BGP routers, one per routing domain. To enable and configure BGP routing for the other tenant, Contoso, the cmdlets are the same but the value of the RoutingDomain parameter changes to "Contoso" with the BgpIdentifier as the IPv4 address of IfCi1. The values for other parameters can be substituted similarly.

Multi-tenant Network Address Translation

Multi-tenant Network Address Translation (NAT) in Windows Server 2012 R2 can enable cloud service provider Fabrikam to enable VMs that have overlapping IP addresses on virtual networks for Woodgrove Bank and Contoso, Ltd hosted in Fabrikam to access services on the Internet.

Woodgrove Bank has a VM with IP address 10.0.0.10 in its virtual network hosted in Fabrikam. An application in this VM that binds to source port 5001 needs to access port 80 of an Internet server 65.10.10.100. Since IP address 10.0.0.10 is not routable on the Internet, when the packet exits the Fabrikam network, the IP address of the packet is translated to the public IP address 131.107.10.10. The NAT mapping table on the NAT gateway is shown here:

	TENANT	WOODGROVE BANK
Original Packet	Source IP	10.0.0.10
	Source Port	5001
	Destination IP	65.10.10.100
	Destination Port	80
	Protocol	*
Translated Packet	Source IP	131.107.10.10
	Source Port	9001
	Destination IP	65.10.10.100
	Destination Port	80
	Protocol	*

When the return packet comes from 65.10.10.100 to the Fabrikam network, the following NAT mapping table is used:

	TENANT	**WOODGROVE BANK**
Original Packet	Source IP	65.10.10.100
	Source Port	*
	Destination IP	131.107.10.10
	Destination Port	9001
	Protocol	*
Translated Packet	Source IP	65.10.10.100
	Source Port	*
	Destination IP	10.0.0.10
	Destination Port	5001
	Protocol	*

For a regular single-tenant NAT, just translating the destination IP address and port would be good enough. With multi-tenancy, however, since there are multiple tenants with the same destination IP address 10.0.1.1, the NAT should also map the IP or port to the appropriate tenant. This is achieved by maintaining the mapping of the VSID with the tuple.

Figure 2-12 shows VMs with IP addresses 10.0.0.10 for the two tenants Woodgrove Bank and Contoso trying to access a web page hosted on a web server with IP address 65.10.10.10. When the packets are routed to the multi-tenant gateway, they end up in their respective compartments. Since 65.10.10.10 is reachable only via the Internet, there will not be any matching route in the Woodgrove Bank or Contoso compartment. If multi-tenant NAT is enabled and there is no matching route for the destination in the tenant compartment, the packets are translated by NAT with the configured Internet address.

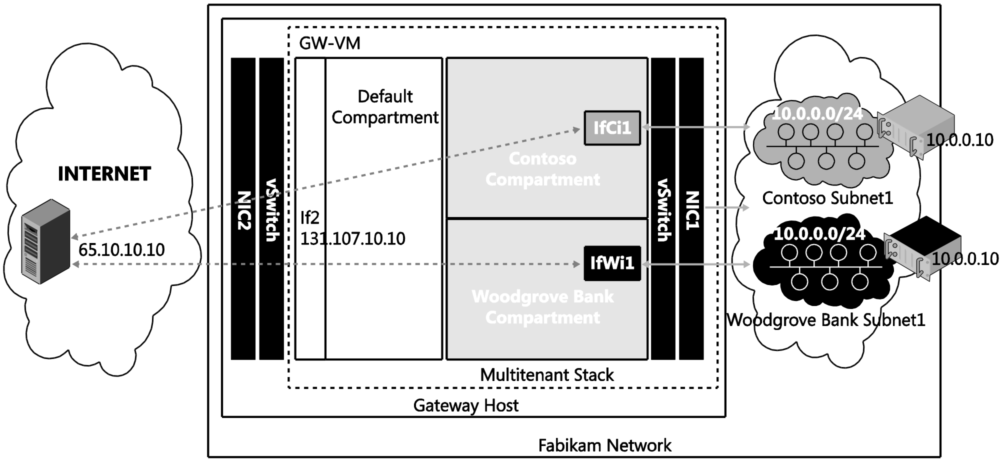


FIGURE 2-12 Multi-tenant NAT.

The following cmdlets show how to enable multi-tenant NAT:

```
New-NetNat -Name Contoso -ExternalIPInterfaceAddressPrefix 131.107.10.12/32
   -InternalRoutingDomainId "{12345678-1000-2000-3000-123456780001}"
New-NetNat -Name Woodgrove -ExternalIPInterfaceAddressPrefix 131.107.10.13/32
   -InternalRoutingDomainId "{12345678-1000-2000-3000-123456780002}"
Add-NetNatExternalAddress -NatName MT-NAT-Contoso -IPAddress 131.107.10.12
   -PortStart 443 -PortEnd 443
Add-NetNatStaticMapping -NatName MT-NAT-Contoso -Protocol "TCP"
   -ExternalIPAddress 131.107.10.12  -ExternalPort 443 –InternalIPAddress 10.0.0.100
   -InternalPort 443
```

Additional resources

The following additional resources build and elaborate on the concepts that have been covered in this book:

- For a walkthrough of how you can build a test lab for implementing HNV, see the blog post “Software Defined Networking: Hybrid Clouds using Hyper-V Network Virtualization (Part 2)” at *http://blogs.technet.com/b/privatecloud/archive/2013/11/21/software-defined-networking-hybrid-clouds-using-hyper-v-network-virtualization-part-2.aspx*.
- For an example of how a cloud hosting provider could offer Disaster Recovery as a Service (DRaaS) using HNV, see the blog post “Software Defined Networking: Hybrid Clouds using Hyper-V Network Virtualization (Part 3)” at *http://blogs.technet.com/b/privatecloud/archive/2013/11/28/software-defined-networking-hybrid-cloud-using-hyper-v-network-virtualization.aspx*.

About the authors

NADER BENMESSAOUD is a Program Manager on the Windows Server and System Center CAT team at Microsoft. Nader has deep expertise in private and hybrid cloud solutions, automation, and application management.

CJ WILLIAMS has worked at Microsoft since 2001 in a number of areas, including Windows, Windows Embedded, Bing, and Microsoft Research. CJ is currently a Principal Program Manager on the Microsoft Windows Networking team focused on Data Center and cloud networking. His specific area of expertise is Network Virtualization and Software Defined Networking. Before working on the Windows team, CJ worked in Microsoft Research, focusing on large-scale distributed systems and bringing distributed systems programming to networks.

UMA MAHESH MUDIGONDA is a program manager on the Windows Server and System Center networking team in the Microsoft India development center, Hyderabad. His areas of expertise include optical networks, VPN, routing, DNS, and cloud networking. He has a bachelor's degree in Computer Science from Osmania University Hyderabad and master's degree from the Indian Institute of Technology Madras.

About the series editor

MITCH TULLOCH is a well-known expert on Windows Server administration and virtualization. He has published hundreds of articles on a wide variety of technology sites and has written or contributed to over two dozen books, including *Windows 7 Resource Kit* (Microsoft Press, 2009), for which he was lead author*; Understanding Microsoft Virtualization Solutions: From the Desktop to the Datacenter* (Microsoft Press, 2010); and *Introducing Windows Server 2012* (Microsoft Press, 2012), a free ebook that has been downloaded almost three quarters of a million times.

Mitch has been repeatedly awarded Most Valuable Professional (MVP) status by Microsoft for his outstanding contributions to supporting the global IT community. He is a nine-time MVP in the technology area of Windows Server Software Packaging, Deployment & Servicing. You can find his MVP Profile page at *http://mvp.microsoft.com/en-us/mvp/Mitch%20Tulloch-21182*.

Mitch is also Senior Editor of WServerNews (*http://www.wservernews.com*), a weekly newsletter focused on system administration and security issues for the Windows Server platform. With more than 100,000 IT pro subscribers worldwide, WServerNews is the largest Windows Server–focused newsletter in the world.

Mitch runs an IT content development business based in Winnipeg, Canada, that produces white papers and other collateral for the business decision maker (BDM) and technical decision maker (TDM) audiences. His published content ranges from white papers about Microsoft cloud technologies to reviews of third-party products designed for the Windows Server platform. Before starting his own business in 1998, Mitch worked as a Microsoft Certified Trainer (MCT) for Productivity Point.

For more information about Mitch, visit his website at *http://www.mtit.com*.

You can also follow Mitch on Twitter at *http://twitter.com/mitchtulloch* or like him on Facebook at *http://www.facebook.com/mitchtulloch*.

Now that you've read the book...

Tell us what you think!

Was it useful?
Did it teach you what you wanted to learn?
Was there room for improvement?

Let us know at http://aka.ms/tellpress

Your feedback goes directly to the staff at Microsoft Press, and we read every one of your responses. Thanks in advance!

Microsoft