BROCADE

Brocade University

BCVRE in a Nutshell Study Guide for Exam 170-010

Brocade University

Revision 0114

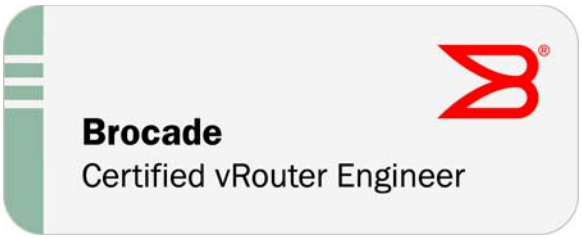
Corporate Headquarters - San Jose, CA USA
T: (408) 333-8000
info@brocade.com

European Headquarters - Geneva, Switzerland
T: +41 22 799 56 40
emea-info@brocade.com

Asia Pacific Headquarters - Singapore
T: +65-6538-4700
apac-info@brocade.com

Revision 0114

Introduction to BCVRE in a Nutshell First Edition



Objective: The BCVRE Nutshell guide is designed to help you prepare for the BCVRE 2013 Certification.

Audience: The BCVRE Nutshell self-study guide is intended for those who have successfully completed the recommended vRouter self-paced training courses and who wish to review concepts and commands before taking the actual BCVRE 2013 exam. The Nutshell guide is not intended as a substitute for training or hands-on time with Brocade products.

How to make the most of the BCVRE guide: The BCVRE guide summarizes the key topics on the BCVRE exam for you in an easy to use format. It is organized closely around the exam objectives. To benefit from the BCVRE guide, we strongly recommend you have successfully completed the following courses:

- SDN 111-WBT - Brocade Vyatta vRouter Software Installation
- SDN 121-WBT - Brocade Vyatta vRouter Command Line Interface
- SDN 132-WBT - Brocade Vyatta vRouter Dynamic Addressing and DNS
- SDN 133-WBT - Brocade Vyatta vRouter Dynamic Host Configuration Protocol
- SDN 211-WBT - Brocade Vyatta vRouter Ethernet and VLAN Configuration
- SDN 321-WBT - Brocade Vyatta vRouter Static Routes
- SDN 341-WBT - Brocade Vyatta vRouter OSPF Basics
- SDN 411-WBT - Brocade Vyatta vRouter Network Address Translation
- SDN 421-WBT - Brocade Vyatta vRouter Firewall Basics
- SDN 511-WBT - Brocade Vyatta vRouter Management and Logging

We hope you find this useful in your journey towards BCVRE Certification, and we welcome your feedback by sending an email to jcannata@brocade.com.

Joe Cannata
Certification Manager

Table of Contents

1 -Brocade Vyatta vRouter System Operations 1
Show Commands. 1
Key CLI Operations 2
Commit and Save Process 3

2- Ethernet 5
Ethernet operations and settings. 5
VLAN operations and settings 6
Link Aggregation 7

3 - TCP/IP 9
End-to-End Packet Flow. 9
TCP and UDP 12
IP Addressing 14
Understanding IP Addresses. 14
Network Masks 14
Understanding Subnetting 15
4 - DHCP and DNS 16
DHCP 16
DNS. 18

5 - Routing 20
Routing Tables. 20
Static Routes 21
Static Default Routes 22
Floating Static Routes. 22

6 - Firewalls 23
Stateful firewalls 23
The Firewall Rulebase. 24
State-Based Rules. 27
Applying Rulebases 27

7 - NAT. 29
Network Address Translation 29
vRouter Packet Processing. 29
NAT Rulebases. 29
Exclusion Filters. 32

8 - Licensing and Upgrades. 34

Entitlements 34
Upgrading the vRouter 34

9 - Logging 36
Logging Basics 36
Feature-Specific Logging 36
Monitoring in Real-Time 37
Sample Log Output 38

10 - OSPF Single-Area 40
Open Shortest Path First 40
Designated Router (DR) *40*
Neighbor Adjacency *41*
Areas *43*
OSPFs Four Level Routing Hierarchy *44*
Link State Advertisement *45*
Configuring OSPF 45
Verifying OSPF Operations 46

Taking the Test 49

List of Figures

End-to-End Flow Example 1 of 4 9
End-to-End Flow Example 2 of 4 10
End-to-End Flow Example 3 of 4 11
End-to-End Flow Example 4 of 4 12
TCP three-way handshake 13
: Static Routes 21
vRouter packet processing 24
Where to place stateful rules 27
OSPF DR Election 41
OSPF Areas 43
OSPF AS 44

List of Tables

Default administrative distances for common protocols 22

1 -Brocade Vyatta vRouter System Operations

When you have completed this section, be sure you can do the following:

- Describe show command usage and output
- Identify key CLI operations
- Describe the commit and save processes

Show Commands

To determine who has been accessing your vRouter, and from what location, use the command **show system login users**.

```
vyatta@vyatta:~$ show system login users
Username    Type      Tty    From          Last login
operator    vyatta    hvc0                 Mon Jun 17 18:34:25 2013
vyatta      vyatta    pts/0 10.224.7.101 Wed Jun 19 20:47:39 2013
vyatta@vyatta:~$
```

To display disk utilization on your vRouter, use the command **show system storage**.

```
vyatta@training:~$ show system storage
Filesystem            Size  Used Avail Use% Mounted on
unionfs               1.9G  1.6G  194M  90% /
tmpfs                 248M     0  248M   0% /lib/init/rw
udev                  241M  156K  241M   1% /dev
tmpfs                 248M  4.0K  248M   1% /dev/shm
/dev/sda1             1.9G  1.6G  194M  90% /live/image
/dev/sda1             1.9G  1.6G  194M  90% /live/cow
tmpfs                 248M     0  248M   0% /live
tmpfs                 248M   12K  248M   1% /tmp
/dev/sda1             1.9G  1.6G  194M  90% /opt/vyatta/etc/config
tmpfs                 248M  120K  248M   1% /var/run
none                  248M  384K  248M   1% /opt/vyatta/config
vyatta@training:~$
```

To display system memory utilization on your vRouter, use the command **show system memory.**

```
vyatta@VYA1:~$ show system memory
             total       used       free     shared    buffers     cached
Mem:        514484     252428     262056          0      89084     109528
Swap:            0          0          0
Total:      514484     252428     262056
vyatta@VYA1:~$
```

To display information about the software version, use the command **`show version`**. Note the differences in output between a bare metal installation and a virtual installation.

Bare metal:

```
vyatta@training:~$ show version
Version:      VSE6.5R3
Description:  Vyatta Subscription Edition 6.5 R3
Copyright:    2006-2013 Vyatta, Inc.
Built by:     autobuild@vyatta.com
Built on:     Thu Jan 24 21:34:57 UTC 2013
Build ID:     1301242135-3e8ca0b
System type:  Intel 32bit
Boot via:     image
HW UUID:      Not Present
Uptime:       00:34:26 up 2 days, 23:39,  1 user,  load average: 0.00, 0.03,
0.05

vyatta@training:~$
```

Virtual:

```
vyatta@VYA1:~$ show version
Version:      VSE6.6R1
Description:  Brocade Vyatta 5410 vRouter 6.6 R1
Copyright:    2006-2013 Vyatta, Inc.
Built by:     autobuild@vyatta.com
Built on:     Wed Jul 24 16:19:58 UTC 2013
Build ID:     1307241637-7e824ac
System type:  Intel 32bit Virtual
Boot via:     image
Hypervisor:   VMware
HW model:     VMware Virtual Platform
HW S/N:       VMware-56 4d 4c 44 b2 e4 14 cd-6c 7f e1 ae d7 bf 5c 1b
HW UUID:      564D4C44-B2E4-14CD-6C7F-E1AED7BF5C1B
Uptime:      00:08:39 up 78 days,  5:07,  2 users,  load average: 0.00, 0.01,
0.05

vyatta@VYA1:~$
```

Key CLI Operations

The vRouter has two CLI modes:

- **Operational mode.** In this mode, you can issue `show` commands to examine system and feature operations, view and clear statistical counters, reset connections, and reboot the vRouter device. Both device operators and administrators can access operational mode commands.

- **Configuration mode.** In this mode, you can issue `set` commands to enable or alter system operations. You can view the current configuration with show commands. You can also issue operational mode commands by using the keyword `run` followed by the operational mode command. Configuration mode can only be accessed by device administrators.

Context-sensitive help is available at any point by typing the ?. If issued directly at the prompt, all first-level commands for that mode are displayed. If issued within a command (see example below), the system displays only the output relevant for the command.

```
vyatta@VYA1:~$ show ip ospf ?
Possible completions:
  <Enter>       Execute the current command
  border-routers
                Show IPv4 OSPF border-routers information
  database      Show IPv4 OSPF database information
  interface     Show IPv4 OSPF interface information
  neighbor      Show IPv4 OSPF neighbor information
  route         Show IPv4 OSPF route information


vyatta@VYA1:~$ show ip ospf neighbor ?
Possible completions:
  <Enter>       Execute the current command
  <x.x.x.x>     Show IPv4 OSPF neighbor information for specified IP address
or interface
  address       Show IPv4 OSPF neighbor information for specified IP address
  detail        Show detailed IPv4 OSPF neighbor information
```

You can abbreviate commands to a unique character string at any level. You can also use the TAB key to complete commands.

Commit and Save Process

Making permanent changes to the vRouter configuration is a three-step process. The first step is to type your configuration commands. The vRouter stores these commands in the configuration buffer until you are ready to activate them with the `commit` command. When you type `commit`, the vRouter copies your changes from the configuration buffer to the active configuration file in memory. You can now monitor the results of your changes. However, your changes still aren't permanent until you save them.

The `save` command is only available in configuration mode. By default, the configuration is saved to the file `/config/config.boot`. Although you can save configuration to alternate file names, the vRouter will only load `/config/config.boot` when starting up.

When you enter the `commit` command, the vRouter automatically archives the configuration buffer contents. The previous buffer contents are saved to the /config/archive directory. By default, the vRouter stores up to 20 previous committed configurations. You can view information about the archived files with the operational command `show system commit`.

```
vyatta@VYA1:~$ show system commit
0   2013-10-21 17:39:35 by vyatta via cli
1   2013-10-21 17:39:22 by vyatta via cli
2   2013-10-21 17:06:17 by vyatta via cli
3   2013-10-18 19:30:18 by vyatta via cli
4   2013-10-17 19:28:42 by vyatta via cli
5   2013-10-11 18:44:04 by vyatta via cli
6   2013-10-08 22:21:35 by vyatta via cli
7   2013-10-08 22:21:06 by vyatta via cli
8   2013-10-08 22:17:48 by vyatta via cli
9   2013-10-07 19:03:04 by vyatta via cli
10  2013-10-07 19:02:13 by vyatta via cli
11  2013-09-25 17:13:32 by vyatta via cli
12  2013-09-25 17:00:48 by vyatta via cli
13  2013-09-25 16:58:49 by vyatta via cli
14  2013-09-25 16:57:28 by root via boot-config-loader
15  2013-09-25 16:54:27 by vyatta via cli
16  2013-09-25 16:49:38 by vyatta via cli
17  2013-09-23 21:49:55 by vyatta via cli
18  2013-09-11 22:08:16 by vyatta via cli
19  2013-09-11 22:08:10 by vyatta via cli
vyatta@VYA1:~$
```

You can load a previously-saved configuration file directly into the configuration buffer using the `load` command. Just specify the name of the saved configuration file you want to load. Note that this loads the file into the configuration buffer, not directly into memory. If you want the changes to take effect, you need to issue the `commit` command.

If you haven't issued any save commands while making changes, you can return to the bootup configuration without having to reboot the box by using the `load` command with the default configuration file `config.boot`. This will only work if you haven't saved the changes you've made.

Another option is to merge the existing configuration with a saved configuration file using the `merge` command. You might use this if you have standard settings, such as host name tables, or a basic set of firewall rules that you apply to all devices. Merge adds the specified file to the existing configuration buffer rather than replacing it. Again, the changes don't take effect until you issue the `commit` command.

2- Ethernet

When you have completed this section, be sure you can do the following:

- Configure basic Ethernet port operations
- Configure VLAN interfaces
- Configure bonded interfaces

Ethernet operations and settings

Like most Ethernet ports, you can manually configure the port speed and duplex level of a vRouter Ethernet port. The default for all Ethernet ports is to auto-detect speed and duplex level.

The vRouter has two settings relating to Ethernet MAC addressing:

- `hw-id`: refers to the 48-bit burned-in address on a physical Ethernet port, or the 48-bit virtual address generated by a hypervisor for a virtual Ethernet port. This parameter associates the rest of the Ethernet port configuration with a specific device, and is auto-detected the first time you install a vRouter.
- `mac-addr`: allows you to overwrite the MAC address used when the vRouter generates frames on the specified interface.

All Ethernet numbers are referenced by the format `eth` followed by a number. The number is assigned during the first boot-up of a vRouter, in order of hardware device discovery. You will need to match the `hw-id` detected during boot-up with the physical or virtual ports to ensure you are applying the correct configuration to the desired port. This is especially important on virtual machines, as the order defined by the hypervisor many not match the order that the interfaces are discovered by the vRouter at bootup.

Commands:

```
[set | edit] interface ethernet ethn
   set speed [auto | 10 | 100 | 1000]
   set duplex [auto | half | full]
   set mac mac-address
   set address address/mask
```

To verify Ethernet operations, use the command `show interfaces`.

```
vyatta@vyatta:~$ show interfaces
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface        IP Address                     S/L  Description
---------        ----------                     ---  -----------
eth0             10.10.1.1/24                   u/u
eth1             192.168.12.1/24                d/d
vyatta@vyatta:~$
```

The S/L column refers to State/Link.

You can view detailed operational statistics by adding the word ethernet and the interface number to the show command.

```
vyatta@vyatta:~$ show interfaces ethernet eth0
eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
qlen 1000
    link/ether 36:0a:27:2b:50:06 brd ff:ff:ff:ff:ff:ff
    inet 172.24.42.51/24 brd 172.24.42.255 scope global eth0
    inet6 fe80::340a:27ff:fe2b:5006/64 scope link
       valid_lft forever preferred_lft forever

    RX:  bytes    packets     errors    dropped    overrun      mcast
       3656114      49450          0          0          0          0
    TX:  bytes    packets     errors    dropped    carrier collisions
      42829027     729851          0          0          0          0
vyatta@vyatta:~$
```

VLAN operations and settings

A Virtual LAN (VLAN) is a logical subgroup within a local area network (LAN) that is created through software rather than manually moving cables in the wiring closet. It combines user stations and network devices into a single unit regardless of the physical LAN segment they are attached to and allows traffic to flow more efficiently within populations of mutual interest. VLANs have the following characteristics:

- Creates a broadcast domain
- Done through software configuration
- Implemented in port switching and LAN switches

802.1Q tagging is an IEEE standard that allows a networking device to add information to a Layer 2 frame in order to identify its VLAN membership. A port can belong to only one port-based VLAN, unless 802.1Q tagging is applied to the port. A port that performs tagging is also called a trunk port.

On the vRouter, trunk ports are implemented by configuring virtual interfaces on a physical interface. Each sub-interface has its own IP address, and is associated with a VLAN tag number. 802.1Q supports 4092 tags. There are two ways to reference a virtual VLAN interface in the vRouter:

- The first format is used when configuring the virtual interface itself. You'll specify the physical interface, the keyword vif, then the VLAN id number. Example: `interface ethernet eth0 vif 10`
- When configuring other features that reference a VIF, such as NAT or OSPF, you'll use the format physical interface dot vlan number. The keyword VIF isn't used. Example: `eth0.10`

When issuing the `show interface` command in operational mode, either format is accepted by the CLI, and will generate the same output. Example:

```
vyatta@vyatta1:~$ show interfaces ethernet eth1.10
eth1.10@eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UP
    link/ether 36:0a:27:2b:50:06 brd ff:ff:ff:ff:ff:ff
```

```
    inet 10.10.10.1/24 brd 10.10.10.255 scope global eth0.10
    inet6 fe80::340a:27ff:fe2b:5006/64 scope link
       valid_lft forever preferred_lft forever

    RX:  bytes    packets     errors    dropped    overrun      mcast
               0          0          0          0          0          0
    TX:  bytes    packets     errors    dropped    carrier collisions
             368          4          0          0          0          0
vyatta@vyatta:~$
```

Commands:

```
[set | edit] interface ethernet ethn vif vlan-id
   set address address/mask
```

Link Aggregation

An aggregated link is a bundle of separate Ethernet interfaces bonded into a single logical pipe. As far as the IP network is concerned, the bundle is a single high speed interface. A bonded interface supports all features of a physical Ethernet interface, including multiple IP addresses, VLAN subinterfaces, firewalls, and so on. The one exception is that you cannot assign pseudo-Ethernet interfaces to a bonded interface.

Benefits include:

- Additional bandwidth. Frequently, the order of magnitude jump from one level of Ethernet technology to the next is cost-prohibitive. You may not need a 1 gigabit link, but your 100 megabit link is experiencing congestion. Link aggregation allows you to increase the bandwidth incrementally, without making the jump directly to 1 gigabit.
- Redundancy. If one of the links in the aggregate group goes down, the link itself still functions, with traffic distributed over the remaining active links.
- Active monitoring of link status. Instead of relying on electrical or optical link state, link status is actively monitored by the link aggregation control protocol, part of the IEEE standard for link aggregation. Link state can sometimes report false positives, depending on the device and connection, but LACP accurately determines whether a link is up or down.

In order to implement link aggregation, all the links in the bundle must be configured for full duplex connectivity and must be point-to-point links. connected to the same Ethernet switch. The Ethernet switch must comply with the 802.1AX standard for link aggregation. Links can be different speeds, and there is no limit to the number of links in a bundle.

To configure an aggregate link, you first create a bond interface, configure that interface as you would a physical interface, then assign the physical links to the bond interface group. If you want to delete an aggregate interface, you must first remove the physical links from the bond group, then delete the bond interface.

Commands:

```
[set | edit] interface bonding bondn
set interface ethernet ethn bond-group bondn
show interfaces bonding
```

3 - TCP/IP

When you have completed this section, be sure you can do the following:

- Explain the relationship between Layer 2, IP, and TCP/UDP
- Identify the differences between TCP and UDP
- Identify IP address subnets

End-to-End Packet Flow

The following example details how a packet is routed from Host A to Host B on another subnet or network address.

1. If the destination host's network number was the same as the source host's, then the destination host would be considered local and on the same subnet. This is determined by taking Host A taking its own IP address and subnet mask and determining its own network address and then doing the same operation with the destination IP and source's subnet mask and comparing the results. If they are the same then the destination Host B would be considered local; Otherwise the packets will be forwarded to the default gateway in order to be sent to a remote host. In this example the destination Host B's Network ID of 192.168.3.0 is different from the source Host A's Network ID of 192.168.1.0 and therefore the packets will need to be routed to the destination Host B.
2. The source Host A must check its own Local Route Table for its default gateway (this is the general behavior unless a special route has been defined). The default gateway IP is the IP of the routing interface for that subnet. In this example it is 192.168.1.1 which is the IP of Router 1 Interface E1. Since this is an Ethernet LAN, Host A will need to encapsulate the frame in order to send it out to the routing interface of E1 and to do so it needs to know the MAC address of the routing interface. If it is not in its local cache an ARP broadcast will need to be initiated in order to send the encapsulated frames to the routing interface (E1 on Router1).

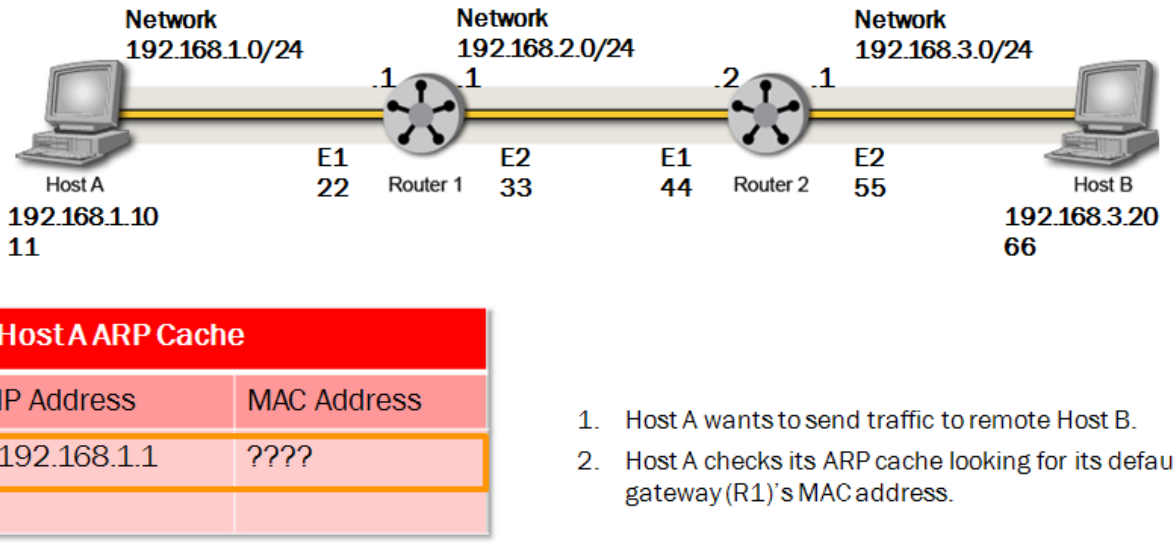


FIGURE 1 End-to-End Flow Example 1 of 4

3. In Figure 2, the default gateway's MAC address is not in Host A's cache. Host A initiates a local ARP broadcast request attempting to resolve the IP address to a physical MAC address.
4. Router 1 responds with a unicast ARP response to Host A with its MAC address of 22.
5. Host A creates/encapsulates an Ethernet frame with its own MAC 11 as the source and a destination MAC address of 22. Notice the destination IP still remains 192.168.3.20 and the frame can be sent on the wire.

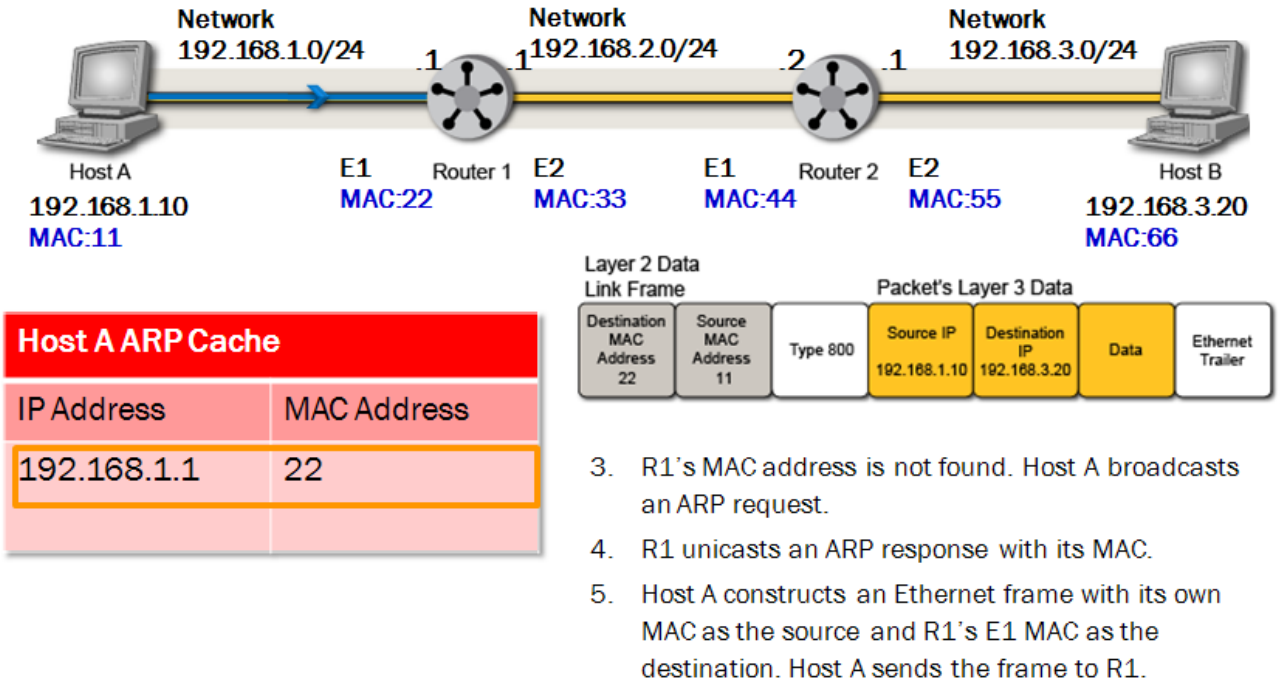
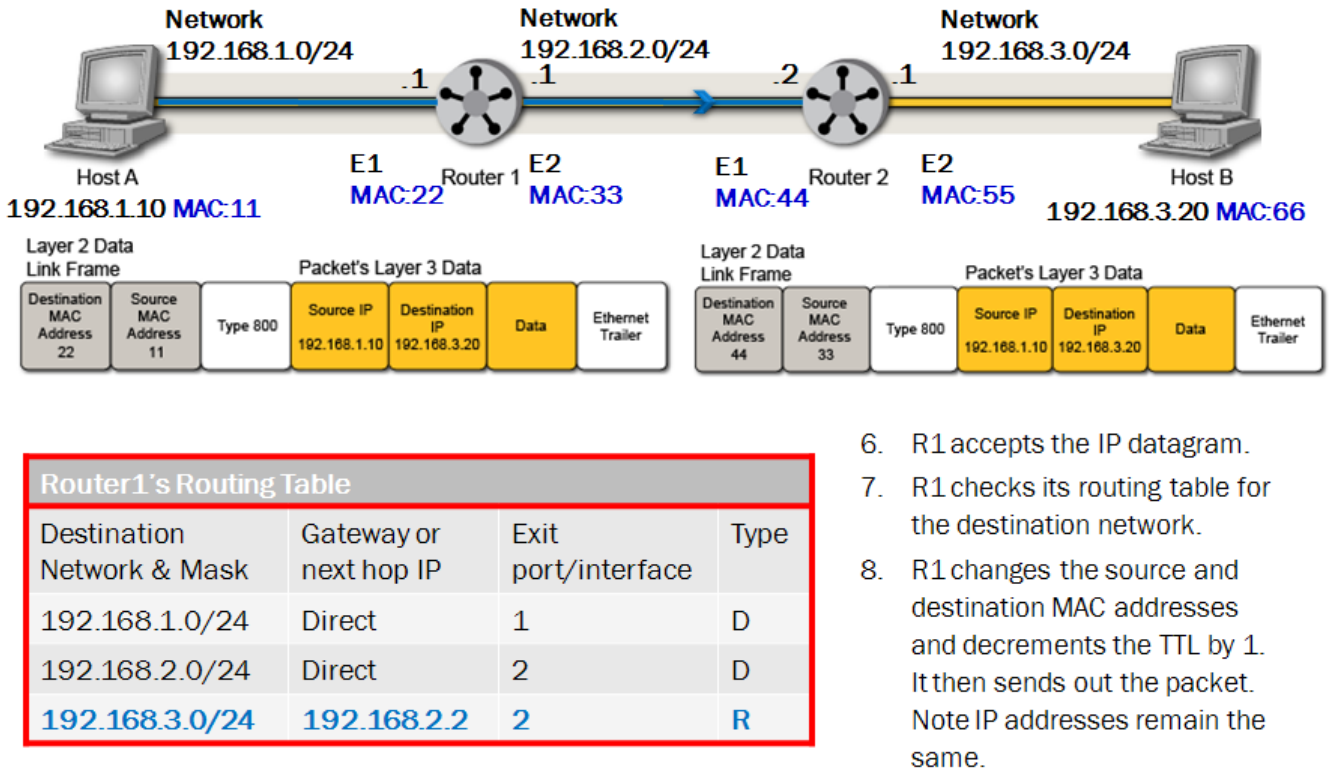


FIGURE 2 End-to-End Flow Example 2 of 4

6. Once Interface E1 on R1 receives the Ethernet frame it looks at the destination MAC address of the frame to check it if matches his own in order to determine if he is the recipient of the frame. In this case R1 interface E1 is the default gateway of Host A and therefore the intended recipient. R1 checks the frame's Type field and notices 0x800 which indicates that there is an IP packet in the data portion of the Ethernet frame. R1 then proceeds to decapsulate the Ethernet frame in order to analyze the destination IP of the packet.
7. The Router must then consult its routing table to determine what to do with the packet. In general terms it looks to identify network routes in its table which would include the destination IP address as a host address on that network. Note: If there are several viable routes to the destination network it will choose the route with the longest "subnet mask" match. How routing tables become populated and an in depth look of how they are evaluated are beyond the scope of this example. After viewing R1's routing table it finds that the network address of 192.168.3.0 is the destination network where these packets need to be routed. It also notices the next hop IP of 192.168.2.2 which represents the next stop for the packets on its way to the 192.168.3.0 network and this can be reached through local interface E2.

8. In order for R1 to do the frame encapsulation process it needs to know the MAC address of the 192.168.2.2 interface. So it must check its local ARP cache and again if the MAC address is not found, it must send an ARP broadcast to request the MAC address. In this case it will be already present. Therefore the frame encapsulation process can continue. Notice that the source and destination IP addresses stay the same but the source MAC becomes 33 and the destination MAC becomes 44. Note that it also will decrement the Time to Live field of the packet (in the IP header) by 1. Now the packet is sent on the wire.



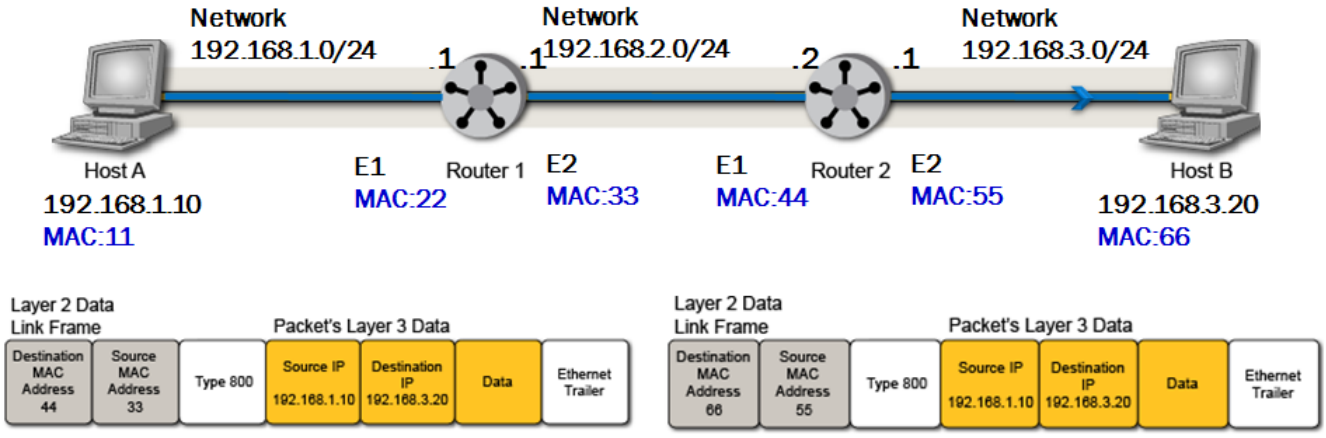
Router1's Routing Table			
Destination Network & Mask	Gateway or next hop IP	Exit port/interface	Type
192.168.1.0/24	Direct	1	D
192.168.2.0/24	Direct	2	D
192.168.3.0/24	192.168.2.2	2	R

6. R1 accepts the IP datagram.
7. R1 checks its routing table for the destination network.
8. R1 changes the source and destination MAC addresses and decrements the TTL by 1. It then sends out the packet. Note IP addresses remain the same.

FIGURE 3 End-to-End Flow Example 3 of 4

9. Once Interface E1 on R2 receives the Ethernet frame, it looks at the destination MAC address of the frame to check it if matches his own in order to determine if he is the recipient of the frame. In this case R2 interface E1 is the next hop IP of R1 and therefore the intended recipient. R2 checks the frame's Type field and notices 0x800 which indicates that there is an IP packet in the data portion of the Ethernet Frame. R2 then proceeds to decapsulate the Ethernet frame in order to analyze the destination IP of the packet.
10. R2 must then consult its routing table to determine what to do with the packet. After consulting its routing table it finds that the Network Address of 192.168.3.0 is the destination network where these packets need to be forwarded and this is a directly connected route in its table through interface E2.

11. In order for R2 to do the frame encapsulation process it needs to know the MAC address of the final destination host B with the IP 192.168.3.20. So it must check its local ARP cache and again if the MAC address is not found it must be a ARP broadcast to resolve the IP Address to a matching physical MAC address. In this case it will be already present and therefore the frame encapsulation process can continue. Notice that the source MAC is 55 and the destination MAC becomes 66. Now the frame(s) are sent on the wire.
12. Once Host B receives the frame, it recognizes its own MAC address. It then decapsulates the frame and notices that itself is the intended host with an IP of 192.168.3.20.



9. R2 accepts the IP datagram.
10. R2 checks its routing table for the destination network.
11. R2 changes the source and destination MAC addresses and decrements the TTL by 1. It then sends out the packet. Note IP addresses remain the same.
12. Host B accepts the packet.

FIGURE 4 End-to-End Flow Example 4 of 4

Note

Throughout the packet flow the source and destination IP addresses stay the same, but the source and destination MAC changes. Also, the Time to Live field of the packet (in the IP header) is decremented by 1.

TCP and UDP

Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) are layer 4 protocols used for end-to-end data transmission over IP. They have different characteristics and applications, which you need to understand for proper firewall implementations.

TCP is a connection-oriented protocol. This means that the two end stations establish a connection before exchanging any application data. This exchange is called a three-way handshake.

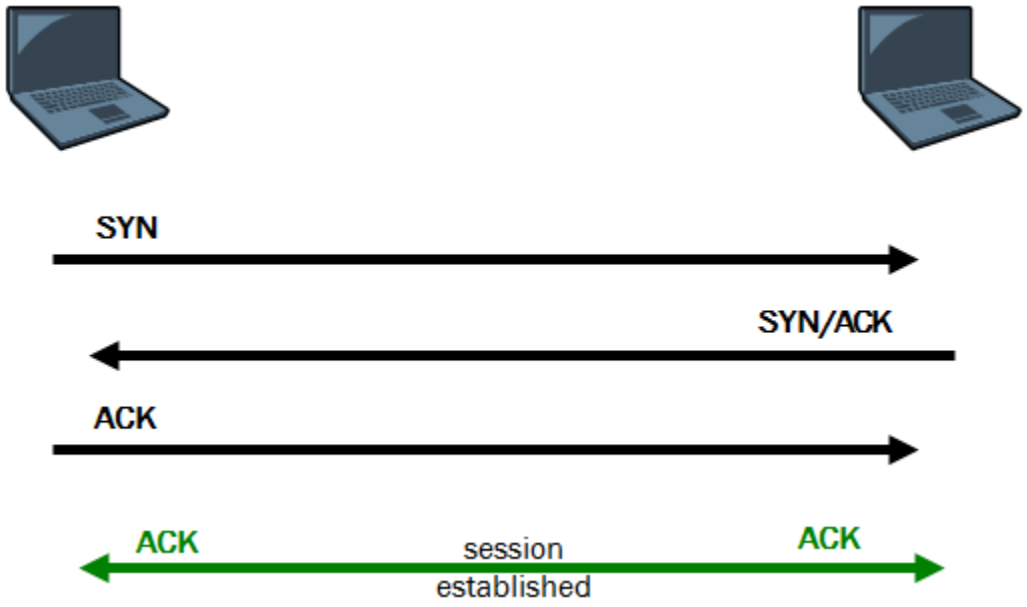


FIGURE 5 TCP three-way handshake

1. The initiating station sends a packet with the SYN bit set, indicating a session start.
2. The responder sends a packet with both the SYN and ACK bits set – the SYN bit to synchronize the session counters, and the ACK bit to acknowledge the initial SYN packet.
3. The initiator then sends a packet with just the ACK bit set, indicating that the two devices have synchronized their counters and can begin exchanging data.

During the data exchange, workstations will continue to count and acknowledge packets during the exchange. This ensures that the data arrives in the correct order, and that no data is missing. If packets are missing, the receiver will not acknowledge their receipt, allowing the transmitter to resend them as needed. The latency introduced by the acknowledgment process, especially on unreliable media, can result in slower overall connectivity.

UDP, on the other hand, is a connectionless protocol. End stations do not establish a connection, do not acknowledge packets, and do not perform retransmissions of individual packets.

TCP is typically used for connections where reliability is more important than speed, such as HTTP traffic, mail server exchanges, and FTP. UDP is used where speed is key, or when data does not have to be acknowledged, such as streaming media, VoIP, and one-way queries such as DNS.

IP Addressing

Understanding IP Addresses

An IP address is an address used in order to uniquely identify a device on an IP network. An IP address consists of 32 bits, which can be divided into two portions: the network, and the host. A second 32-bit binary number called the network mask determines which bits are which.

When writing an IP address down, the 32 bits are broken down into four octets. Each octet is then converted into a decimal number. The four decimal numbers are then separated by a period (dot). This format is called dotted decimal format. (for example, 172.16.32.45). The value in each octet ranges from 0 to 255 decimal, or 00000000 - 11111111 binary.

Here is how binary octets convert to decimal: The right most bit, or least significant bit, of an octet holds a value of 2. The bit just to the left of that holds a value of 4. This continues until the left-most bit, or most significant bit, which holds a value of 128. So if all binary bits are a one, the decimal equivalent would be 255 as shown here:

```
  1   1   1   1 1 1 1 1
128  64  32  16 8 4 2 1 (128+64+32+16+8+4+2+1=255)
```

Here is a sample octet conversion when not all of the bits are set to 1.

```
0   1 0 0 0 0 0 1
0  64 0 0 0 0 0 1 (0+64+0+0+0+0+0+1=65)
```

And this is sample shows an IP address represented in both binary and decimal.

```
       10.        1.       23.       19 (decimal)
00001010.00000001.00010111.00010011 (binary)
```

These octets are broken down to provide an addressing scheme that can accommodate large and small networks. There are five different classes of networks, A to E. This document focuses on addressing classes A to C, since classes D and E are reserved and discussion of them is beyond the scope of this document. The terms "Class A, Class B" and so on are used in this document to help facilitate the understanding of IP addressing and subnetting. These terms are rarely used in the industry anymore because of the introduction of classless interdomain routing (CIDR).

Network Masks

The network mask determines which portion of the IP address identifies the network and which portion identifies the individual device. Class A, B, and C networks have the default masks shown here:

Class A: 255.0.0.0

Class B: 255.255.0.0

Class C: 255.255.255.0

In the mask, the bits set to 1 represent the network, while the bits set to 0 represent the host. To see how the mask helps you identify the network and node parts of the address, let's convert a class A IP address into binary, then identify the network number and the host bits.

```
10.20.15.1 = 00001010.00010100.00001111.00000001
255.0.0.0  = 11111111.00000000.00000000.00000000
```

Once you have the address and the mask represented in binary, then identifying the network and host ID is easier. Any address bits which have corresponding mask bits set to 1 represent the network ID. Any address bits that have corresponding mask bits set to 0 represent the node ID.

```
10.20.15.1 = 00001010.00010100.00001111.00000001
255.0.0.0  = 11111111.00000000.00000000.00000000
             ----------------------------------
              net id |      host id

netid =  00001010 = 10
hostid = 00010100.00001111.00000001 = 20.15.1
```

Understanding Subnetting

Subnetting allows you to create multiple logical networks that exist within a single classful network. Each data link on a network must have a unique network number, with every node on that link being a member of the same network. If you break a major network (Class A, B, or C) into smaller subnetworks, it allows you to create a network of interconnecting subnetworks. Each subnet has a unique network number as determined by the network mask. All devices on the subnet have the same subnet number, with each device having a unique host portion of the address.

To subnet a network, you increase the length of the classful mask, using some of the bits from the host ID portion. This means you have fewer hosts per subnet, but more subnets to segment your network. The following tables show you the number of subnets and hosts available for each mask length for a class C network.

Table 1: Available Subnets and Hosts based on Mask Length - Class C

Bits	# Subnets	# Hosts
1	2	126
2	4	62
3	8	30
4	16	14
5	32	6
6	64	2

Let's look at a specific example. If you have a class C network of 204.17.5.0, and you need six network segments, the smallest subnet mask you can use is three bits according to the table. If we write that in binary,

```
204.17.5.0 -       11001100.00010001.00000101.00000000
255.255.255.224 -  11111111.11111111.11111111.11100000
                   -------------------------|sub|----
```

we can see that the last octet is now divided into two portions. We still have five bits for the host portion of the address, but we can now use some of those higher-order bits for the network number. The five host bits give us 32 unique host addresses per network. However, we can only use 30 of them; you cannot assign a device a host address of all zeros or all ones.

So, with this in mind, we've created the following subnets.

```
204.17.5.0 255.255.255.224     host address range 1 to 30
204.17.5.32 255.255.255.224    host address range 33 to 62
204.17.5.64 255.255.255.224    host address range 65 to 94
204.17.5.96 255.255.255.224    host address range 97 to 126
204.17.5.128 255.255.255.224   host address range 129 to 158
204.17.5.160 255.255.255.224   host address range 161 to 190
204.17.5.192 255.255.255.224   host address range 193 to 222
204.17.5.224 255.255.255.224   host address range 225 to 254
```

In the vRouter, subnet masks are typically noted in bitcount format. In this case, you are using a 27 bit mask, so you'd represent an address as 204.17.5.33/27.

4 - DHCP and DNS

When you have completed this section, be sure you can do the following:

- Describe, configure, and verify the vRouter support for DHCP
- Describe, configure, and verify the vRouter support for DNS

DHCP

Dynamic Host Configuration Protocol, or DHCP, is a protocol that allows end stations to request IP configuration information from a local sever rather than having to be manually configured. DHCP can provide several pieces of IP configuration data to the end station, including IP address, IP gateway address, and addresses for IP services, such as DNS servers and WINS servers.

DHCP offers several advantages to the network administrator.

- It centralizes management of IP addresses. Instead of manually assigning addresses, then having to track which addresses are in use, the administrator configures a pool of addresses on the DHCP server that will be assigned to end stations. The server takes care of the assignment and tracking.
- Because addresses are used in an on-demand fashion, available address space is used efficiently, rather than having hard-coded addresses consumed even if a device is not in use.

- DHCP also means less configuration needs to be done on individual end stations. Most devices today boot up with DHCP automatically enabled, so you don't have to touch an end station to enable IP connectivity to a network.

As a DHCP client, the vRouter can acquire an IP address on any Ethernet interface.

Commands:

```
set interface ethernet ethn address dhcp
show dhcp client leases
```

```
vyatta@rtr1:~$ show dhcp client leases
interface  : eth1
ip address : 10.224.7.189       [Active]
subnet mask: 255.255.255.0
router     : 10.224.7.1
name server: 10.0.0.30 10.0.0.31
dhcp server: 10.224.7.1
lease time : 86400
last update: Wed Dec 23 21:40:51 GMT 2009
expiry     : Thu Dec 24 21:40:50 GMT 2009
reason     : BOUND

vyatta@rtr1:~$
```

For DHCP server functionality, the vRouter has two options:

- DHCP server. A basic DHCP server setup consists of an address pool, with specific parameters associated with each address pool, including the start and end addresses for the pool, the default router, the address of the DNS server, and other optional parameters.
- DHCP relay agent. If you have a central DHCP server on a different network segment than your end stations, you can configure the vRouter to forward DHCP requests to that server.

DHCP server commands:

```
[set | edit] service dhcp-server
   [set | edit] shared-network-name name
[set | edit] subnet address/mask
         set default-router address
         set dns-server address
         set start address [stop address]
         set exclude address
```

DHCP relay commands

```
[set | edit] service dhcp-relay
   set interface int-name
   set server address
```

Use show commands to verify DHCP server functionality.

```
vyatta@vyatta:~$ show dhcp server leases
IP address       Hardware Address   Lease expiration     Pool      Client Name
----------       ----------------   ----------------     ----      -----------
192.168.42.10    00:0c:29:f5:40:6e  2009/11/04 23:52:07  DHCP-Eth0 JansPC
192.168.42.11    00:0c:29:a5:02:c7  2009/11/04 23:52:11  DHCP-Eth0 Desktop
192.168.42.22    00:15:c5:b3:2e:64  2009/11/04 17:55:01  DHCP-Eth0
192.168.42.23    00:04:f2:02:84:49  2009/11/04 17:24:59  DHCP-Eth0 FredsPC

vyatta@vyatta:~$
```

There are no show commands to verify DHCP relay functionality.

DNS

A vRouter participates in DNS services in one of three ways.

- **System DNS.** With system DNS, the vRouter acts as a DNS client and needs to resolve host names for its own operations. You can configure a static system DNS server, or the vRouter can acquire the address of its DNS server using DHCP.
- **Dynamic DNS.** Dynamic DNS is used when upstream devices use names to reach an address associated with the vRouter, but the address is assigned via DHCP. In order for name resolution to work, a device needs to have a unique name and address combination registered in DNS. If the vRouter has a static address, then the entry is statically added to DNS. However if the vRouter is acquiring its address via DHCP, then it may not acquire the same address every time. With a dynamic address, a static DNS entry won't work. In this case, we need to configure dynamic DNS. When configured, the vRouter will update the configured dynamic DNS server with its address and name whenever the address changes. As we mentioned on the previous slide, the DDNS provider should set the time to live for your Dynamic DNS entry to an appropriately short duration.
- **DNS forwarding.** In this case, downstream end stations need to reach DNS services on the Internet. Again, this scenario is common in environments where the vRouter acquires its public addressing and DNS server address via DHCP. If the address of the vRouter changes, the available DNS server address may also change. If clients have a static DNS server configuration, they may not always be able to reach a DNS server. The solution is to configure DNS forwarding on the vRouter. This solution requires the client-facing interface – in this case, ethernet 0 – to have a static IP address. Clients use the static address of the vRouter as the address of their DNS server. The vRouter relays client requests to its DNS server, either the system DNS server or one learned via DHCP. Because the interface connected to the end stations has a static address, those workstations will always use the same address for DNS. Because the vRouter can learn DNS server addresses via DHCP, it can always relay DNS queries correctly.

  Before you configure DNS forwarding, you need to determine which DNS server you want to use for DNS queries. By default, the vRouter will first try to reach any configured system DNS servers. If no system server exists, or if the configured server does not respond, the vRouter will next try any DNS servers learned via DHCP. You can override these defaults by selecting only system servers, only DHCP-learned servers, or an explicitly-configured server address just for DNS forwarding.

Commands:

```
set system name-server name
[set | edit] service dns dynamic interface name service provider
   set login name
   set password password
   set server [ip-addr | fqdn]
   set host-name name
[set | edit] service dns forwarding
   set listen-on int-name
   set system
   set dhcp int-name
   set name-server ip-addr
set system static-host-mapping host-name name inet ip-addr
```

To view operational statistics for DNS forwarding, use the command `show dns forwarding statistics`.

```
vyatta@rtr1:~$ show dns forwarding statistics
----------------
Cache statistics
----------------
Cache size: 150
Queries forwarded: 5
Queries answered locally: 2
Total DNS entries inserted into cache: 23
DNS entries removed from cache before expiry: 0

---------------------
Nameserver statistics
---------------------
Server: 10.0.0.30
Queries sent: 5
Queries retried or failed: 0

vyatta@rtr1:~$
```

Note the queries answered locally. This means that the device responded directly to those queries because it already had the DNS entry in the local cache.

Also note that individual server statistics only shows the first name server. The vRouter only displays statistics for servers it has attempted to communicate with. Because the first server in the list is 10.0.0.30, and this server has not experienced any failures, the vRouter has never sent any queries to 10.0.0.31, so there are no statistics for the second server.

5 - Routing

When you have completed this section, be sure you can do the following:

- Explain the purpose and function of routing
- Interpret show commands used with routing
- Configure different types of static routes

Routing Tables

A router uses its routing table to determine the next hop for the packet's destination and forwards the packet appropriately[1]. The next router repeats this process using its own routing table until the packet reaches its destination. At each stage, the IP address in the packet header is used to determine the next hop. If either a destination network or a default route are not in the routing table, the packet is dropped.

On a vRouter, use the operational command `show ip route` to display the device routing table.

```
vyatta@rtr2:~$ show ip route
Codes: K - kernel, C - connected, S - static, R - RIP, B - BGP
       O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter
area
       > - selected route, * - FIB route, p - stale info

Gateway of last resort is not set

C>* 10.1.1.0/30 is directly connected, eth1
C>* 10.2.2.0/30 is directly connected, eth2
C>* 127.0.0.0/8 is directly connected, lo
S>* 172.16.0.0/16 [1/0] via 10.1.1.1, eth1
O   172.16.0.0/16 [110/20] via 10.1.1.1, eth1, 00:00:35
C>* 192.168.0.0/24 is directly connected, eth0
vyatta@rtr2:~$
```

The first column indicates the source of the route information. Routes marked with an asterisk are active routes. The information between brackets is the administrative distance for the route entry, followed by the route metric.

1. When an IP packet is to be forwarded, a router uses its routing table to determine the next hop for the packet's destination (based on the destination IP address in the IP packet header), and forwards the packet appropriately. The next router then repeats this process using its own routing table, and so on until the packet reaches its destination. At each stage, the IP address in the packet header is sufficient information to determine the next hop; no additional protocol headers are required.

Static Routes

A static route is a route manually created by a network administrator to instruct a router how to reach a particular remote network. In order to configure a static route, the administrator needs the destination network and mask, and the address of the directly-connected router that's next along the path to the destination. Figure 5 shows a network of 3 routers. Below that is the configuration needed on each router to provide full network reachability. .

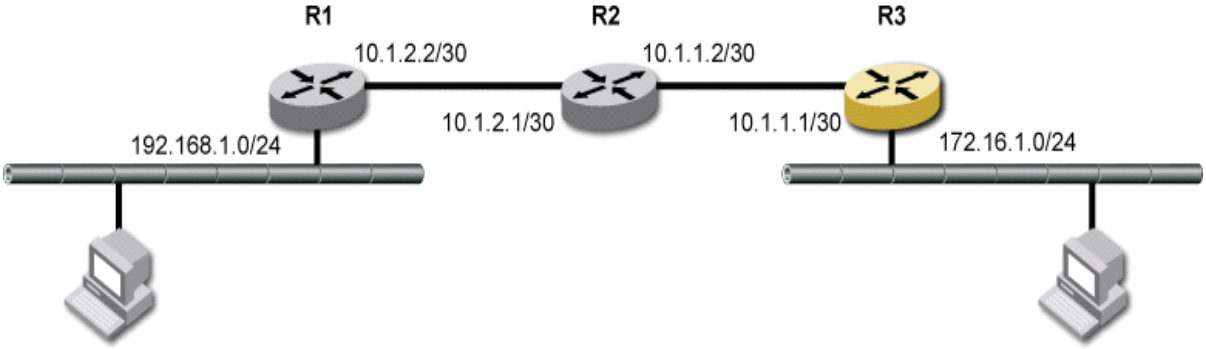


FIGURE 6 : Static Routes

```
[edit]
vyatta@R1# set protocol static route 172.16.1.0/24 next-hop 10.1.2.1

[edit]
vyatta@R2# set protocol static route 172.16.1.0/24 next-hop 10.1.1.1
[edit]
vyatta@R2# set protocol static route 192.168.1.0/24 next-hop 10.1.2.2

[edit]
vyatta@R3# set protocol static route 192.168.1.0/24 next-hop 10.1.1.2
```

Static routes provide several benefits to network managers. First, static routes are really simple to configure, all you need are the destination address, the next hop address and an optional metric. No knowledge of routing protocols is required. Second, once configured, static routes are simple to maintain. Just set and forget, without worrying about interoperability issues or other complexities than can be associated with dynamic routing protocols. Third, static routes can provide a measure of stability and security because they can not be impacted by changing input from other devices. Static routing also uses less CPU and memory because they do not need a separate dynamic protocol process to exchange routes with neighbors. Finally, the use of static routes is not an all or nothing decision. Static routes are frequently used to compliment dynamic routing protocols

Static Default Routes

A default route is a routing table entry used to route packets when an explicit route to a destination network is not in the routing table. It is the network route used by a router when no other known route exists for a given IP packet's destination address. It is last in the order of execution of the routing table. A vRouter can learn a static route via a dynamic routing protocol such as OSPF, or an administrator can configure a static default route. To do so, configure a static route to network 0.0.0.0/0.

```
[edit]
vyatta@vyatta# set protocol static route 0.0.0.0/0 next-hop 192.168.1.1
```

Floating Static Routes

Floating static routes are unused static routes in the routing table. They are typically used to provide backup for dynamic routing protocols, and are configured to have a higher administrative distance than the dynamic protocol.

Administrative distance measures the priority, or believability, of a routing protocol.The lower the administrative distance, the higher priority a given routing entry has. By default, static routes have a very high priority. In fact, the only higher priority is a directly connected network. Dynamic routing protocols, on the other hand, have relatively low priorities. This means that if we have two routes to the same network, one from OSPF and one static route, the vRouter will use the static route.

Protocol	Distance
Connected	0
Static	1
EBGP	20
OSPF	110
RIP	120
IBGP	200

TABLE 1 Default administrative distances for common protocols

To configure a floating static route, add a high administrative distance to the route configuration.

```
[edit]
vyatta@R3# set protocol static route 192.168.1.0/24 next-hop 10.1.1.2
distance 150
```

6 - Firewalls

When you have completed this section, be sure you can do the following:

- Explain the purpose and function of a vRouter firewall
- Describe how the firewall fits in with other vRouter packet processing
- Configure firewall rulebases and apply them to interfaces and zones
- Interpret show commands used with firewalls

Stateful firewalls

A firewall is a device that blocks unwanted traffic from entering your network. However, unlike a traditional router access list, firewalls are stateful. This means that the firewall tracks information about sessions between devices, and not just individual packets. In a stateful operation, the first packet in a session passes through the firewall rules. If the rules permit the traffic, the firewall not only passes the traffic, but adds information about the session to its session table, called the conntrack table in the vRouter. All other packets in the session match the entry in the session table and are permitted without having to look at the firewall rules again. Stateful firewalls can also automatically allow the reverse-direction flow of a session without needing any additional rules.

Another element of firewall functionality is the additional intelligence required to understand how complex protocols work and permit or deny all related traffic. This intelligence is called an application layer gateway, and it is commonly used for multi-port protocols like FTP and SIP.

In the vRouter, the firewall filtering function occurs after destination NAT and the routing lookup, but before source NAT. When configuring firewall filter rules, you need to consider whether the traffic you want to filter is being translated in order to configure the correct addresses in your rules.

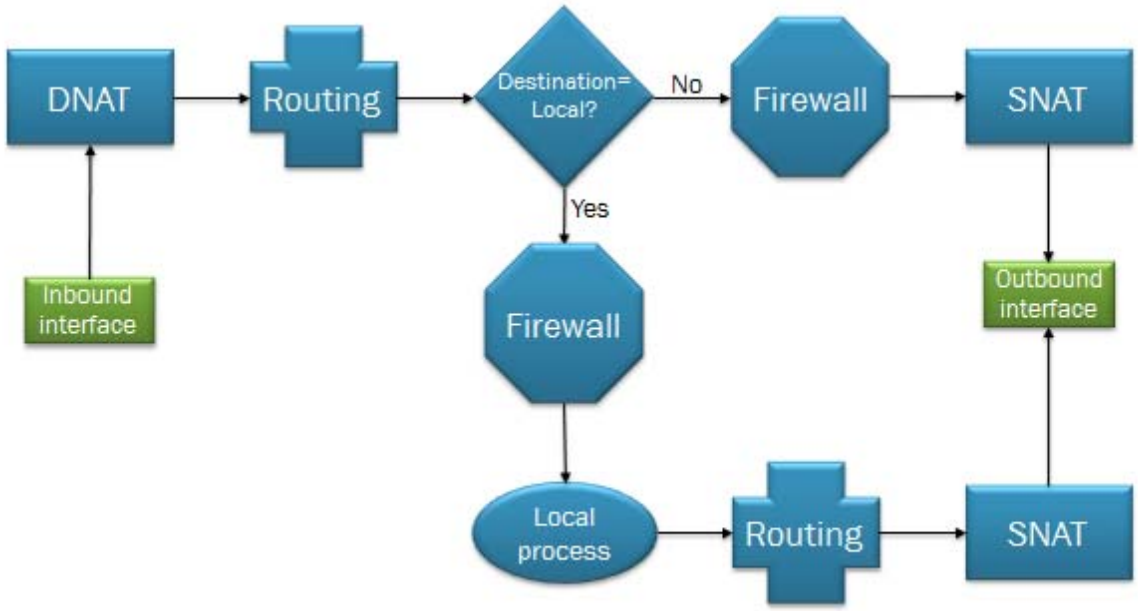


FIGURE 7 vRouter packet processing

The Firewall Rulebase

A firewall rulebase is simply a list of individual firewall rules. Each rule includes

- Match criteria: the traffic you want to filter. You can specify source and destination addresses, source and destination ports, protocol, or a combination of these in each rule. If you specify any port numbers, you need to specify the layer 4 protocol (TCP, UDP, or both). If you don't specify a match criteria, all traffic will match the rule.
- Action: the action to take on a packet that matches the filter. The options are Accept, which permits the packet into the network, Reject, which discards the packet and sends an ICMP unreachable message back to the traffic source, and Drop, which silently discards the packet The default rulebase action is to drop any unmatched packets. You can change this to reject or accept as needed.

In the vRouter, each rulebase has a unique name. You can create as many different rulebases as you need to support your security requirements, and can apply the same rulebase to multiple locations. A best practice is to name the rulebase according to its function.

Each rulebase is an ordered list, with each rule having a unique number within the list. When the vRouter compares a packet with the rulebase, it starts with the first rule in the list, and continues until a match is found. Once a match is found, the device performs the action for that rule and does not look any further. This means that the order of your list is important. You should specify the most specific rules first in the list, then

add your more general rules later in the list. You also need to remember that the default action for a list is to drop traffic, so if a packet arrives at a firewalled interface and does not match a rule, the packet will not go through the router. A best practice is to leave spaces between your rule numbers to allow you to add additional rules as needed.

Sample rulebase:

```
[edit firewall]
vyatta@vyatta1# show name PublicServers
 rule 10 {
     action accept
     state {
         established enable
         related enable
     }
 }
 rule 20
     action accept
     destination {
         address 10.6.7.0/24
     }
     source {
         address 10.2.3.0/24
     }
 }
 rule 30 {
     action accept
     destination {
         address 10.6.7.0/24
         port smtp
     }
     protocol tcp
     source {
         address 10.4.5.0/24
     }
 }
 rule 40 {
     action reject
     destination {
         address 10.6.7.0/24
     }
     source {
         address 10.4.5.0/24
     }
 }
 rule 50 {
     action accept
     destination {
```

```
        address 10.6.7.0/24
        port http,ftp,smtp
    }
    protocol tcp
}
```

You can also view the contents of the firewall rulebase without entering configuration mode.

```
vyatta@vyatta1:~$ show firewall name PublicServers

IPv4 Firewall "PublicServers":

 Active on (eth1,OUT)

rule  action   proto     packets  bytes
----  ------   -----     -------  -----
10    accept   all       5615675  6516819834
 condition - saddr 0.0.0.0/0 daddr 0.0.0.0/0 state RELATED,ESTABLISHED

20    accept   all       229      24120
  condition - saddr 10.2.3.0/24 daddr 10.6.7.0/24

Output omitted

10000 drop     all       384211   33476589

vyatta@vyatta1:~$
```

Note that the output displays the rule number, the action, the match criteria, and counters for packets/bytes that matched each rule. You can display a single rule by adding the rule number to the end of the command.

You can view counters in a different format with the command `show firewall statistics`. You can enter the command by itself to display all firewalls on the device, or add the rulebase name to display counters for a single rulebase.

```
vyatta@vyatta1:~$ show firewall name PublicServers statistics

IPv4 Firewall "PublicServers":
Active on (eth1,OUT)

rule  packets   bytes     action  source              destination
----  -------   -----     ------  ------              -----------
10    5.62M     6.52G     ACCEPT  0.0.0.0/0           0.0.0.0/0
20    51        13036     ACCEPT  10.2.3.0/24         10.6.7.0/24
30    0         0         ACCEPT  10.4.5.0/24         10.6.7.0/24
40    0         0         REJECT  10.4.5.0/24         10.6.7.0/24
50    0         0         ACCEPT  0.0.0.0/0           10.6.7.0/24
1025  2042      923057    DROP    0.0.0.0/0           0.0.0.0/0
```

Note that this output doesn't include details about protocols and ports, only the IP address information.

State-Based Rules

As we said earlier, vRouter firewall operations are stateful, and you may need to add rules regarding statefulness to your rulebases, depending on the flow of traffic in your network and where you place your firewalls. You have to look at both directions of a traffic session.

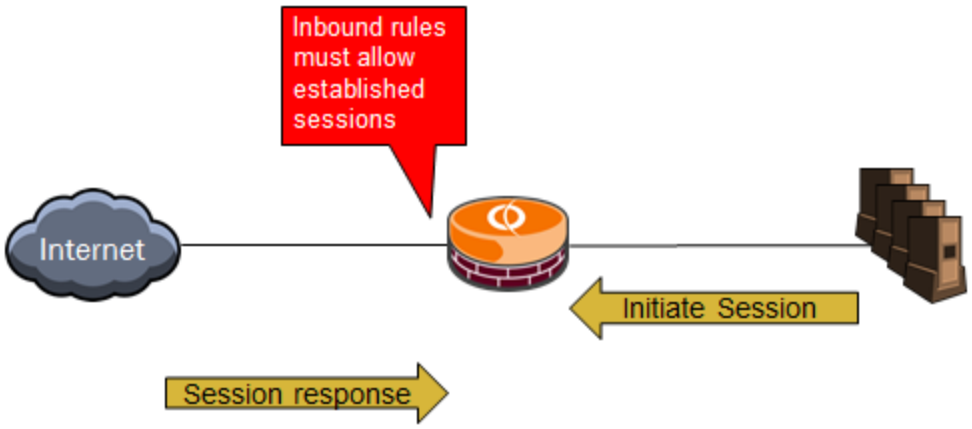


FIGURE 8 Where to place stateful rules

InFigure 8, we have a rulebase that prevents unwanted traffic from the Internet from reaching our servers. We've configured the filter to allow for specific protocols, and the default action of denying all other traffic is in place. However, when a workstation inside our network tries to browse to the Internet, they cannot establish a connection. The problem is that the conntrack table has no entry for the session originating inside our network; we have no outbound filters to collect any session information. When the response packet arrives at the firewall, it doesn't match any rules or existing conntrack entries, so it gets dropped.

The solution is to add a rule that checks for session state. An established session will have the ACK bit set in the TCP header, so the firewall can look for this in order to only allow "return trip" traffic through the firewall. An associated state is "related" traffic; this enables the ALG function for protocols like FTP and SIP. Note in the earlier example that the first rule in the rulebase was a state-based rule.

Applying Rulebases

On a vRouter, you have two options for applying firewalls:

- **Individual interfaces.** Each interface can have one rulebase in each direction – one for inbound traffic, and one for outbound traffic. You can apply the same rulebase to multiple interfaces, but you configure the application on an interface-by-interface basis.When creating a rulebase to apply to interfaces, you need to consider all possible source and destination addresses.

- **Zones.** A zone is a group of interfaces. You create the zone and specify which interfaces are included in the zone, then apply the rulebase to the zone. By grouping interfaces into zones, you implicitly associate all addresses within the zone with that zone. This means that you can apply general security policies to the zone without having to list specific addresses in the zone. When you apply the rulebase, you specify the source zone and the destination zone of the traffic.

7 - NAT

When you have completed this section, be sure you can do the following:

- Explain the purpose and function of NAT
- Describe how NAT fits in with other vRouter packet processing
- Configure NAT rulebases
- Verify NAT functionality.

Network Address Translation

Network address translation is the replacement of one IP address with another IP address in a packet header. The most common use for NAT is to replace private addresses used within a network with registered public IP addresses in order to communicate over the Internet. According to RFC 1918, certain network addresses are reserved for private use and cannot be routed across the Internet. So, if a device inside a network is assigned an address from the reserved network 10, that device cannot send traffic to the Internet without some kind of address translation.

NAT can also be used inside a network to handle overlapping address ranges. This might occur when two companies merge, and both are using the same range of private network addresses.

NAT can also be used to hide the real address of a publicly-reachable device, such as a Web server. In this case, NAT is used in combination with a firewall, which blocks outside traffic from reaching the real address and only allows certain types of traffic to access the translated address.

There are three basic types of NAT

- Source NAT replaces the source address of a packet as it passes through the vRouter.
- Destination NAT replaces the destination address of a packet as it passes through the vRouter.
- Bidirectional NAT combines source and destination NAT for translation in both directions.

The vRouter handles NAT as a stateful operation. Traffic that is translated in one direction is entered into the conntrack table. This allows the reverse-direction translation to occur automatically.

vRouter Packet Processing

Refer to Figure 7 on page 24.

NAT Rulebases

A NAT rulebase is a numbered list. This means that each rule has its own number. You'll have one rulebase for source translation, and another for destination translation, each with its own set of numbered rules. The vRouter evaluates the rules in numerical order. If a packet matches a rule, the vRouter performs the translation defined in the rule, then exits the list. No other list items are evaluated. Because item ordering is important, vRouter recommends that you leave room between each rule in the list for future additions. A typical practice is to increment rule numbers by 10, so you'd create rule 10, then 20, then 30, and so on.

Each rule includes three parameters:

- Filters, which identify the traffic to be translated. If you do not define a filter, all traffic will match the rule.
- Post-translation address, which defines the IP address the vRouter will substitute when performing NAT. You must specify a post-translation address. You can specify a single address, an address and port combination, or use the keyword “masquerade”, which instructs the vRouter to use the address of the outbound interface.
- The interface where the rule is applied, and the direction for the rule. You must specify an interface.

If you specify a port number in either the filter or the post-translation adddress, you must specify the layer 4 protocol (TCP, UDP, or both).

In the following sample source rulebase, the vRouter is translating all traffic from network 192.168.0.0/16 that transits interface ethernet 1. The post-translation source address is the address of ethernet 1.

```
[edit]
vyatta@vyatta# show nat source
 rule 10 {
     source {
         address 192.168.0.0/16
     }
     outbound-interface eth1
     translation {
         address masquerade
     }
}
```

In this sample destination rulebase, the vRouter is translating one public address to different private addresses based on the application (port number). This is a common application when you have a limited number of public addresses and have multiple services that need to be reached from the public address space.

```
[edit]
vyatta@vyatta# show nat destination
 rule 10 {
     destination {
         address 1.3.5.7
         port 80
     }
     inbound-interface eth1
     translation {
         address 10.2.3.4
     }
     protocol tcp
}
rule 20 {
     destination {
         address 1.3.5.7
         port 25
```

```
    }
    inbound-interface eth1
    translation {
        address 10.5.6.7
    }
    protocol tcp
}
rule 30 {
    destination {
        address 1.3.5.7
        port 53
    }
    inbound-interface eth1
    translation {
        address 10.8.9.1
    }
    protocol udp
}
```

You can view the same rulebases from operational mode.

```
vyatta@training:~$ show nat source rule
Disabled rules are not shown
Codes: X - exclude rule, M - masquerade rule

rule    intf             translation
----    ----             -----------
M10     eth1             saddr 192.168.100.0/24 to 216.134.166.19
        proto-all        sport ANY

vyatta@VYA1:~$ show nat destination rules
Disabled rules are not shown
Codes: X - exclude rule

rule    intf             translation
----    ----             -----------
10      eth1             daddr 1.3.5.7 to 10.2.3.4
        proto-tcp        dport 80

20      eth1             daddr 1.3.5.7 to 10.5.6.7
        proto-tcp        dport 25

30      eth1             daddr 1.3.5.7 to 10.8.9.1
        proto-udp        dport 53

vyatta@VYA1:~$
```

You can view translation counters for each rule.

```
vyatta@training:~$ show nat source statistics
rule    pkts    bytes   interface
----    ----    -----   ---------
10      528     38349   eth1
20      0       0       eth1
30      1359K   96M     eth1
vyatta@training:~$
```

You can also view the active translations occurring on the device.

```
vyatta@training:~$ show nat source trans
Pre-NAT               Post-NAT             Prot  Timeout
192.168.2.102         216.134.166.19       tcp   47
192.168.2.104         216.134.166.19       udp   0
192.168.2.102         216.134.166.19       udp   49
192.168.2.104         216.134.166.19       tcp   431740
192.168.2.104         216.134.166.19       tcp   431522
192.168.2.102         216.134.166.19       udp   179
192.168.2.104         216.134.166.19       tcp   431739
192.168.2.104         216.134.166.19       tcp   431988
192.168.2.104         216.134.166.19       tcp   431928
192.168.2.104         216.134.166.19       tcp   431810
192.168.2.106         216.134.166.19       tcp   326344
192.168.2.102         216.134.166.19       udp   28
192.168.2.102         216.134.166.19       udp   54
192.168.2.102         216.134.166.19       udp   179
192.168.2.104         216.134.166.19       udp   6
192.168.2.102         216.134.166.19       tcp   431848
```

Exclusion Filters

An exclusion filter allows you to specify traffic that you do NOT want translated. A typical application is when you are performing source NAT on an Internet connection that is also carrying a private VPN. In that case, you want to translate everything EXCEPT traffic crossing the VPN. Rather than trying to identify all the possible types of traffic to be translated, it's easier to identify the traffic you do not want to translate.

There are two formats for exclusion filters. The first format uses the exclamation mark, or "bang" as a NOT operator. You'll add this operator in front of the address range, protocol, or port in the filter statement. When you use the not operator, you're creating a single NAT rule that matches all traffic EXCEPT the traffic identified by the filter. Because of this, you can only use the NOT operator when your excluded traffic falls into a single address range, port, and protocol combination.

If you cannot define your excluded traffic with a single combination, you can use the exclude keyword. In this case, you'll create a rule for every address range/port/protocol combination you need to exclude, then create a default rule that translates everything else.

Exclusion example:

```
[edit]
vyatta@vyatta# show nat destination
rule 10 {
    destination {
        address 10.10.10.0/24
        }
    exclude
    outbound-interface eth0
}
rule 40 {
    outbound-interface eth0
    translation {
        address masquerade
    }
}
```

8 - Licensing and Upgrades

When you have completed this section, be sure you can do the following:

- Verify correct entitlement on your vRouter
- Describe the Brocade Vyatta upgrade process

Note

As of 1 November 2013, the entitlement and upgrade processes described here is no longer available. However, the material is still on the exam. Please consult your vRouter documentation for the latest procedures.

Entitlements

To register your vRouter software, you need to configure your entitlement credentials on your device. These credentials were provided to you when you purchased your software. You need to configure three parameters on your device:

- Repository username
- Repository password
- Entitlement key

You can verify that your device has been registered with the Vyatta entitlement server with the command `show entitlement`.

Upgrading the vRouter

To upgrade your vRouter, enter the command `upgrade system image`. This command automatically downloads the latest software image to your vRouter, and modifies the configuration file to associate it with the new software version. In order to check the vRouter repository, your device must have an active connection to the Internet and the vRouter must be able to perform name resolution (DNS). You also need your repository username and password. You can manually enter this during the upgrade process, or you can preconfigure your credentials as part of the system configuration.

Upgrading a virtual machine is a bit more complex. A vRouter virtual machine template has two major components: the vRouter software itself, and the underlying Linux drivers, system, hypervisor-specific optimization, and so on. When Brocade releases a vRouter update, you may or may not need to update the underlying system template, depending on Linux updates, driver updates, and so on. Rather than having you look this up yourself the upgrade system image command takes care of this for you. It will check not only for a new version of vRouter software, but whether the underlying template also needs to be updated. If the underlying template has not changed, the upgrade process continues. If the underlying template has changed, you will need to perform a manual upgrade.

A manual upgrade is similar to creating a new VM, with some additional steps:

1. Download the new template just as you did for your initial installation.
2. Copy the configuration file from your existing virtual machine. You can use SCP or FTP to copy it to an external server, or use simple copy-paste from a console window.
3. Edit the configuration file to remove the hardware-specific settings. We'll show you the details of what to remove on the next screen.
4. Install a new virtual machine using the new template.
5. When your new VM has booted up, copy your edited configuration file to /config/config.boot on the new system. This is the default configuration file for the vRouter device.
6. Reboot your new VM. When it boots, it will read the hardware values from the hypervisor software, and pull the rest of the configuration data from the configuration file you just copied over.
7. Once your new VM is fully operational, you can cut over operations from the old VM. This cut over represents the only downtime your network will experience during the upgrade process, and should be almost non-disruptive depending on your hypervisor software.

You can verify the success of your device upgrade with the commands `show version` and `show system image`.

9 - Logging

When you have completed this section, be sure you can do the following:

- View vRouter log entries
- Configure the vRouter to capture protocol- and feature-specific information

Logging Basics

One of the advantages of vRouter software is its extensive local logging capabilities. Unlike traditional networking devices, vRouter supports an integrated hard drive, providing capacity for storing local messages without the requirement of an external server. vRouter software uses the Linux standard syslogd process to store logs.

Log messages are stored in the file /var/log/messages. When this file reaches 500KB in size, the vRouter renames the file to messages.0 and opens a new messages file. This continues up to the file messages.9, for a total of 5 gigabytes of log messages. In addition, the vRouter system maintains separate logs for bootup messages, PPP connection setup, IPsec connection setup, and other features.

To view the active log file, use the command `show log`. Because this can be up to 500kilobytes of messages, we have predefined some filters for you - for example, if you want to display entries relating to NAT, use the command `show log nat`. By adding the appropriate option, you'll only see messages relating to the specified feature or protocol. You can also use the Linux-style "pipe" option and specify a specific text string to search for. The device will display all lines in the log that contain the string. If the output is more than one screen, use the pipe and `more` to display one screenful of data at a time. Example:

```
vyatta@training:~$ show log | match ERROR | more
May 16 13:30:50 training pluto[5686]: ERROR: "peer-76.74.103.7-tunnel-1"
#995: sendto on pppoe1 to 76.74.103.7:500 failed in ISAKMP notify. Errno 22:
Invalid argument
May 16 13:31:20 training pluto[5686]: ERROR: "peer-76.74.103.7-tunnel-1"
#995: sendto on pppoe1 to 76.74.103.7:500 failed in ISAKMP notify. Errno 22:
Invalid argument
May 18 00:10:55 training pluto[5686]: ERROR: "peer-76.74.103.7-tunnel-1"
#1043:sendto on pppoe1 to 76.74.103.7:500 failed in ISAKMP notify. Errno 22:
Invalid argument
```

To view the entire set of log files in sequential order, use the command `show log all`. Again, you can use the match tool to search the output. If you just want to see what happened most recently on your device, you can use the `show log tail` command. This displays the last 10 log entries.

Feature-Specific Logging

In order to view protocol and feature activity, you first need to enable the vRouter to record the information in the system log. By default, the vRouter does not capture routine activity, such as routing protocol exchanges or NAT or firewall packets.

For protocol exchanges that only involve the vRouter (DNS, routing updates, etc.), you enable logging from operational mode using the `monitor` command. Be as specific as possible with your commands; protocol logging can be processor-intensive. Below is an example of the some of the OSPF-specific options for logging.

```
vyatta@VYA1:~$ monitor protocol ospf enable ?
Possible completions:
  database-timer
                Enable OSPF database-timer debugging
  events        Enable OSPF event packet debugging
  ifsm          Enable OSPF ifsm debugging
  lsa           Enable OSPF lsa debugging
  nfsm          Enable OSPF nfsm debugging
  nsm           Enable OSPF nsm debugging
  packet        Enable OSPF packet debugging
  route         Enable OSPF route debugging


vyatta@VYA1:~$ monitor protocol ospf enable events ?
Possible completions:
  <Enter>       Execute the current command
  abr           Enable OSPF abr event debugging
  asbr          Enable OSPF asbr event debugging
  lsa           Enable OSPF lsa event debugging
  nssa          Enable OSPF nssa event debugging
  os            Enable OSPF os event debugging
  router        Enable OSPF router event debugging
  vlink         Enable OSPF vlink event debugging
```

For end-user transit traffic (NAT packets, firewall activity, etc.), you have to enable logging for a specific rule or set of rules in configuration mode. Example:

```
[edit]
vyatta@vRouter1# set nat source rule 10 log enable
```

All log entries are captured in the system log at /var/log/messages. Regardless of the type of feature-specific logging you enable, make sure you disable it as soon as you have finished your troubleshooting, in order to preserve system resources and prevent the vRouter from capturing unnecessary log entries.

Monitoring in Real-Time

The monitor command is used to enable protocol-specific logging. It can also be used to enable real-time viewing of any logging, including end-user transit traffic if logging is enabled on a feature rule. To use real-time monitoring:

1. Enable logging for the appropriate protocol or feature. Remember to be as specific as possible; choose a particular rule for security, or a particular message or packet type for protocols.
2. Enable monitoring for the protocol or feature. Examples:

- `monitor protocol ospf`
- `monitor nat source`

3. Run your test traffic, or trigger a protocol update, or whatever you need to do to generate traffic relevant to your troubleshooting.
4. Type CTRL-C to exit monitoring and return to the prompt.

Sample Log Output

OSPF Hello packets:

```
vyatta@vyatta:~$ monitor protocol ospf enable packet hello
vyatta@vyatta:~$ monitor protocol ospf
Apr  5 20:30:51 vRouter ospfd[1949]: Hello received from [172.24.42.53] via
[eth2:192.168.13.1]
Apr  5 20:30:51 vRouter ospfd[1949]:  src [192.168.13.3],
Apr  5 20:30:51 vRouter ospfd[1949]:  dst [224.0.0.5]
Apr  5 20:30:51 vRouter ospfd[1949]: Packet 172.24.42.53 [Hello:RECV]:
Options *|-|-|-|-|-|E|*
Apr  5 20:30:51 vRouter ospfd[1949]: make_hello: options: 2, int:
eth1:192.168.12.1
Apr  5 20:30:51 vRouter ospfd[1949]: make_hello: options: 2, int:
eth2:192.168.13.1
Apr  5 20:30:51 vRouter ospfd[1949]: Hello sent to [224.0.0.5] via
[eth1:192.168.12.1].
Apr  5 20:30:51 vRouter ospfd[1949]: make_hello: options: 2, int:
eth3:192.168.101.1
Apr  5 20:30:51 vRouter ospfd[1949]: Hello sent to [224.0.0.5] via
[eth2:192.168.13.1].
Apr  5 20:30:51 vRouter ospfd[1949]: Hello sent to [224.0.0.5] via
[eth3:192.168.101.1].
```

NAT packets:

```
[edit]
vyatta@vyatta# set nat source rule 30 log enable
[edit]
vyatta@vyatta# commit
[edit]
vyatta@vyatta# run show log nat
Apr  5 18:17:01 vRouter kernel: [595980.330716] [NAT-SRC-30-MASQ] IN=
OUT=pppoe1
SRC=192.168.2.104 DST=173.12.167.194 LEN=56 TOS=0x00 PREC=0x00 TTL=62
ID=52504 PROTO=UDP SPT=7172 DPT=64544 LEN=36
Apr  5 18:17:01 vRouter kernel: [595980.341042] [NAT-SRC-30-MASQ] IN=
OUT=pppoe1
```

```
SRC=192.168.2.104 DST=173.12.167.194 LEN=56 TOS=0x00 PREC=0x00 TTL=62
ID=16918 PROTO=UDP SPT=7172 DPT=64545 LEN=36
Output omitted
[edit]
vyatta@vyatta#
```

10 - OSPF Single-Area

When you have completed this section, be sure you can do the following:

- Explain how OSPF works
- Configure the vRouter for OSPF in a single area
- Verify OSPF operations

Open Shortest Path First

Open Shortest Path First (OSPF) is a link state routing Interior Gateway Protocol (IGP) for medium to large networks. Its cost metric is based on aggregated link cost. OSPF supports Classless Inter-Domain Routing (CIDR) and Variable Length Subnet Masks (VLSM). It is hierarchy-based using OSPF areas. Its network topology is built using Link State Advertisements (LSAs) received from other routers. OSPF has the following characteristics:

- Decreases routing overhead
- Speeds up convergence
- Confines network instability to a single area of the network
- Communicates between routers using multicast advertisements
- Has no periodic updates

Designated Router (DR)

OSPF has a Designated Router which receives all the updates on a given segment. The Backup Designated Router is the second-in-command. This router receives the updates, too, but does not send them back out. When a router needs to update other routers, it sends the update to the Designated Router and the Backup Designated Router. The Designated Router sends the update to all of the other routers on its segment.

When the Designated Router becomes unavailable, the Backup Designated Router instantly becomes the Designated Router, and an election is held to see who becomes the next Backup Designated Router.

This way, each router need not be aware of all of the OSPF routers involved. It needs only to communicate its updates (and receive updates) from a single source. This minimizes traffic, and allows the OSPF design to expand nicely, as needed.

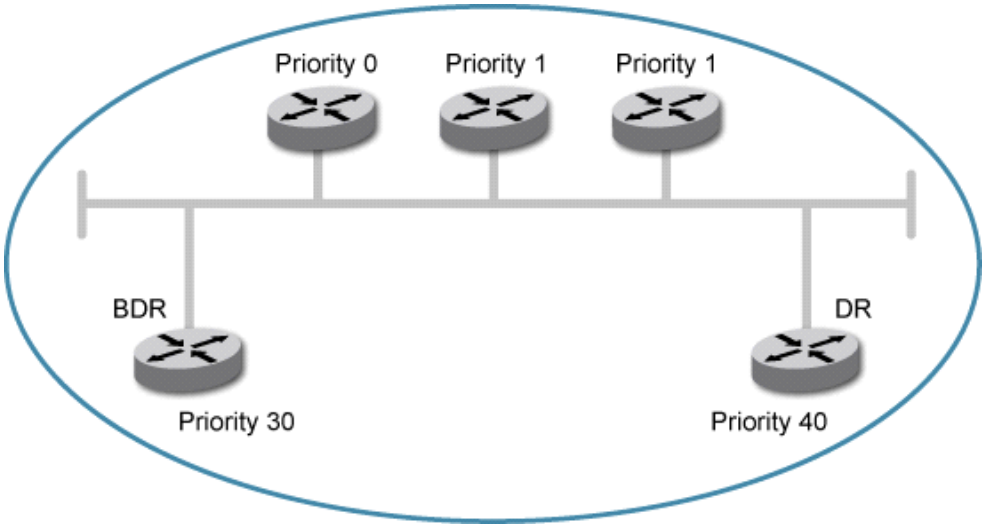


Figure 9: OSPF DR Election

Designated Router (DR) election is done by selecting the neighboring router with the highest priority. The router with the next largest priority is elected as the Backup DR (BDR). If the DR goes offline, the BDR automatically becomes the DR. The router with the next highest priority becomes the new BDR. If two neighbors share the same priority, the router with the highest router ID is designated as the DR.

1. The Router ID can be manually configured using the global `ip router-id x.x.x.x` command.
2. If the Router ID is not manually configured, the IP address configured on the lowest numbered loopback interface is used as the Router ID
3. If there is no loopback interface, then the router ID is the lowest numbered IP address configured on the device

When only one router on the network claims the DR role despite neighboring routers with higher priorities or router IDs; this router remains the DR. This is also true for BDRs.

The DR and BDR election process is performed when one of the following events occurs:

1. An interface is in a waiting state and the wait time expires
2. An interface is in a waiting state and a hello packet is received that addresses the BDR
3. A change in the neighbor state occurs, such as, a neighbor state transitions from 2 or higher, communication to a neighbor is lost, or a neighbor declares itself to be the DR or BDR for the first time

Neighbor Adjacency

OSPF defines a neighbor as a router that has an interface with an IP address in the same broadcast domain. The following parameters must match to become neighbors:

- Subnet mask

- Hello/Dead intervals
- Area-ID
- Authentication password
- Stub area flag

Neighbor States

The neighbors on a given router will go through six states on their way to fully becoming synchronized with its routing table. The first three states are referred to as the neighboring process. These are the states neighbors go through in order to establish themselves as neighbors.

- **Down:** The router has not sent a Hello packet, nor has it received one
- **Init:** A Hello packet has been sent to a neighbor, but no Hello packet has been received from that neighbor
- **2 Way:** The router has sent a Hello packet to a neighbor, and has received a Hello packet back; the reply will contain your router's Router ID inside; now, your router knows that the neighbor knows your router as well

The next three states are referred to as the adjacency process. To establish adjacency means that your OSPF database is synchronized with your neighbor (there's currently no further information to share).

- **Ex-Start:** This is short for “Exchange Start”; this is where the DR/BDR election takes place; routers are sending Hello packets to determine who will become the DR and BDR
- **Exchange:** Now the neighbors are finally talking; in this state, your router is sharing everything it knows, and the neighbor router is sharing everything it knows
- **Full:** We are adjacent! The neighbors have no more information to share; they've shared it all

Areas

Routers in OSPF are split into different groups called areas. The purpose is to reduce traffic and CPU load. The area that is the most restrictive uses the least resources (CPU and memory). Areas may be organized in any way that makes the most sense for a particular network. Areas are assigned numbers on the range of 1 through 4,294,967,295.

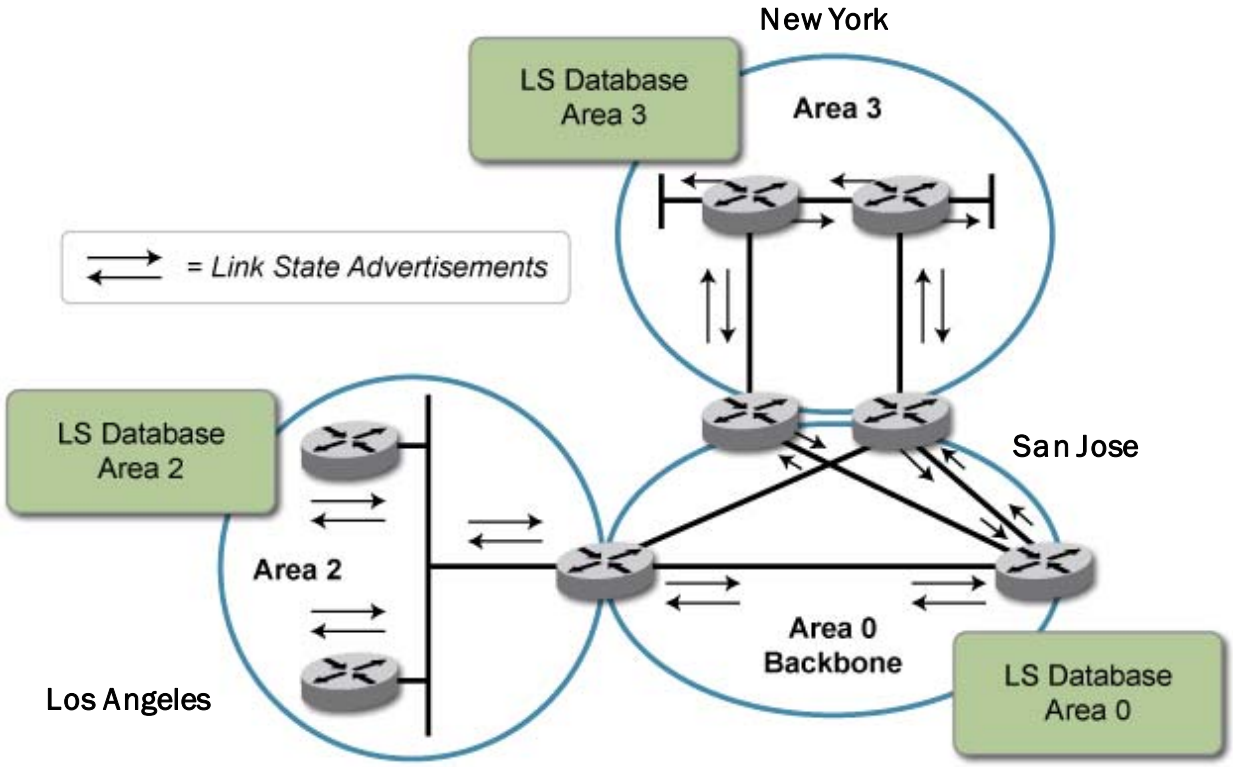


Figure 10: OSPF Areas

Area Types:

- Area 0 - Backbone
  - A required area to which all other areas must connect
- Ordinary or standard area (Normal or transit area)
  - All routers in a OSPF area have the same topological database, but their routing tables are based on the router's position in the area and are unique to the router
- Stub area
  - An area that does not accept external routes but accepts routes from within the same autonomous system
- Not So Stubby Area (NSSA)
  - A stub area does not accept external summary routes from the backbone, but can advertise external summary routes into the backbone

- Totally stubby area
  - This area won't accept routes from any other area. The Area Border Router (ABR) advertises 0.0.0.0/0 instead

OSPF Autonomous System (AS) is the entire OSPF routing domain. An OSPF AS can be divided into multiple areas. The propagation of Types 1 and 2 Link State Advertisements is limited to the bounds of an area.

An OSPF router can be a member of multiple areas. These routers are known as Area Border Routers (ABRs). Each ABR maintains a separate topological database for each area the router is in. Each topological database contains all of the LSAs within a given area. The routers within the same area have identical topological databases. The ABR is responsible for forwarding routing information or changes between its border areas.

An Autonomous System Boundary Router (ASBR) is a router that is running multiple protocols and serves as a gateway to routers outside an AS. The ASBR is able to import and translate different protocols routes into OSPF through a process known as redistribution.

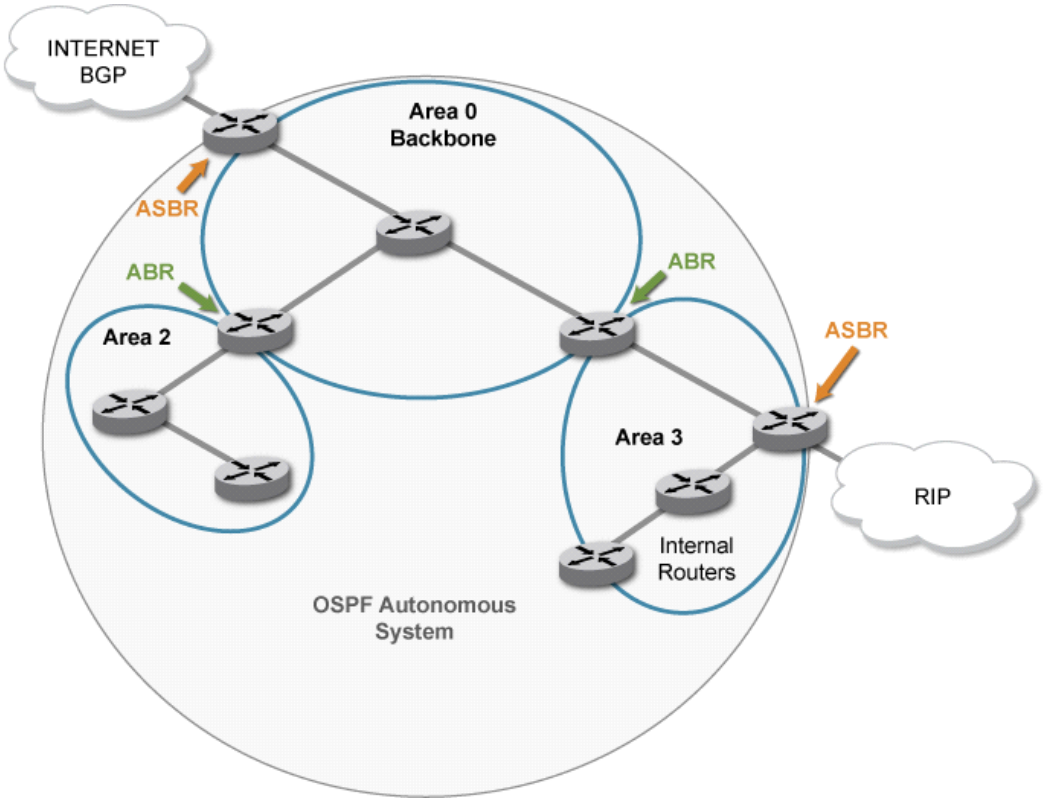


Figure 11: OSPF AS

OSPFs Four Level Routing Hierarchy

- Level 1 - Intra-area routing
- Level 2 - Inter-area routing
- Level 3 - External Type 1 Metrics
- Level 4 - External Type 2 Metrics

If there are two routing paths to choose from then paths that are internal to an OSPF routing domain are preferred over external routes. External routes can be imported into the OSPF domain at two separate levels, one that has Type 1 Metrics and the other Type 2 Metrics.

The use of Type 1 metrics assumes that in the path from the OSPF router to the destination, the internal OSPF AS component (path to the ASBR advertising the AS-external-LSA) and external component are of the same importance.

In Type 2 metrics, it is assumed that only the external component is more significant than the internal component. The aggregate cost to these external destinations does not change when viewed from different routers, since the internal costs are not important. But the cost of Intra-area and Inter-area destinations does change depending on which router the cost is observed.

Link State Advertisement

Link State Advertisement (LSA) is an OSPF data packet that communicates the router's local routing topology to all other local routers in the same OSPF area.

OSPF Link State process:

1. Link State Advertisements exchanged between routers
2. Topology Database is built
3. Router runs Shortest Path First (SPF) algorithm to calculate the best path
4. SPF tree is generated
5. Best routes are selected from SPF tree, and entered into the routing table, based on cost to individual networks

Configuring OSPF

Although OSPF is hierarchical, you do not have to deploy multiple areas. Deploying OSPF in a single-area design is appropriate for smaller networks, say, with 100 or fewer subnets in the OSPF autonomous system. If you do use single-area OSPF, you need to use area number 0 on all your routers. Of course, you won't be able to summarize routes within your OSPF network, But you can still use autonomous system boundary routers to connect to non-OSPF networks. This includes your Internet connection.

Sample OSPF Configuration:

```
[edit]
vyatta@VY-A# show protocols ospf
   area 0 {
       network 10.1.1.0/24
       network 10.10.1.0/24
       network 10.10.2.0/24
   }
[edit]
vyatta@VY-A#
```

The network statement under OSPF does two things: it adds the network specified to the LSA database, and it enables OSPF hellos on the interface(s) associated with the specified network. You can list individual subnets defined on your device, or you can use a supernet that includes one or more subnets.

For stub networks, you can disable the OSPF hellos while still including the subnet in the LSA database by using the `passive-interface` setting.

If you want to advertise a default route within OSPF, you can configure your router connected to the Internet to originate the default route.

```
vyatta@VY-A# show protocols
 ospf {
     area 0 {
         network 10.0.0.0/8
     }
     default-information {
         originate {
         }
     }
     passive-interface eth1
     passive-interface eth2
```

Verifying OSPF Operations

The easiest way to verify that OSPF is working is to view the routing table and check for routes learned from OSPF.

```
vyatta@vyatta:~$ show ip route

Codes: K - kernel, C - connected, S - static, R - RIP, B - BGP
       O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter
area
       > - selected route, * - FIB route, p - stale info

Gateway of last resort is not set

O       10.1.1.0/24 [110/1] is directly connected, eth1, 00:02:57
C    *> 10.1.1.0/24 is directly connected, eth1
O    *> 10.2.1.0/24 [110/2] via 10.10.1.2, eth2, 00:01:30
O    *> 10.2.2.0/24 [110/2] via 10.10.1.2, eth2, 00:01:30
O    *> 10.3.1.0/24 [110/2] via 10.10.2.2, eth3, 00:00:41
O    *> 10.3.2.0/24 [110/2] via 10.10.2.2, eth3, 00:00:41
O       10.10.1.0/24 [110/1] is directly connected, eth2, 00:02:57
Output omitted
```

Note that network 10.1.1.1/24 is in the routing table two times, once for a directly-connected route and once for OSPF. This indicates that the network is being advertised by OSPF, but because it's directly connected, the local connection is the best path to the destination network.

You can verify OSPF neighbor connectivity and DR status:

```
vyatta@vyatta:~$ show ip ospf neighbors

OSPF Process 0:
Neighbor ID    Pri State          Dead Time Address         Interface
172.24.42.52     1 Full/DR           31.721s 10.10.1.2       eth1
172.24.42.53     1 Full/DR           35.533s 10.10.2.2       eth2
vyatta@VY-A:~$
```

The output shows the router IDs of our neighboring routers, the state of the connection – in this case, we have a full adjacency with each neighbor, and the neighbor device is the DR. The dead time counts from 40 seconds down to 0, and is reset by hello messages. The address is the physical address of the neighbor, and the interface is the interface used to reach the neighbor

You can view the contents of the LSA database:

```
vyatta@vyatta:~$ show ip ospf database

            OSPF Router with ID (172.24.42.51) (Process ID 0)

                Router Link States (Area 0.0.0.0)

Link ID         ADV Router      Age  Seq#       CkSum  Link count
172.24.42.51    172.24.42.51     869 0x80000005 0x1d44 3
172.24.42.52    172.24.42.52     884 0x80000005 0x84f7 3
172.24.42.53    172.24.42.53     836 0x80000005 0xe555 4

                Net Link States (Area 0.0.0.0)

Link ID         ADV Router      Age  Seq#       CkSum
10.10.1.1       172.24.42.51     918 0x80000001 0x04a5
10.10.2.1       172.24.42.51     869 0x80000001 0x07a0

                AS External Link States

Link ID         ADV Router      Age  Seq#       CkSum  Route              Tag
0.0.0.0         192.168.200.1    10 0x80000002 0xa3f1 E2 0.0.0.0/0       254
vyatta@VYA1:~$
```

Router link states are a list of all the OSPF routers in the area.

Network link states are a list of networks that are connected to more than one OSPF router.

External links – network advertisements that originated outside of the OSPF network. We'll see this when we add a default route to our OSPF network.

Things to look for in this output: How many links each router in the network is advertising. You can then look at the network links to see which router is advertising which particular network. For example, in our database, we can see that this local router – identified at the top of the output – is advertising 3 links. Two of the links show up in the net link states list. The third is not shown because it is a stub network; it is only advertised by a single router.

You can view the complete list of LSAs advertised by an individual router:

```
vyatta@VY-A:~$ show ip ospf database router 172.24.42.51

            OSPF Router with ID (192.168.200.1) (Process ID 0)

                Router Link States (Area 0.0.0.0)

  LS age: 1587
  Options: 0x2 (-|-|-|-|-|-|E|-)
  Flags: 0x0
  LS Type: router-LSA
  Link State ID: 192.168.200.1
  Advertising Router: 192.168.200.1
  LS Seq Number: 80000005
  Checksum: 0x1d44
  Length: 60
   Number of Links: 3
    Link connected to: Stub Network
     (Link ID) Network/subnet number: 10.1.1.0
     (Link Data) Network Mask: 255.255.255.0
      Number of TOS metrics: 0
       TOS 0 Metric: 1

    Link connected to: a Transit Network
     (Link ID) Designated Router address: 10.10.1.1
     (Link Data) Router Interface address: 10.10.1.1
      Number of TOS metrics: 0
       TOS 0 Metric: 1

    Link connected to: a Transit Network
     (Link ID) Designated Router address: 10.10.2.1
     (Link Data) Router Interface address: 10.10.2.1
      Number of TOS metrics: 0
       TOS 0 Metric: 1

vyatta@VY-A:~$
```

Taking the Test

After the Introduction Screen, once you click on **Next**, you will see the following non-disclosure agreement:

IMPORTANT: PLEASE READ THE FOLLOWING BROCADE NON-DISCLOSURE CONFIDENTIALITY AGREEMENT CAREFULLY BEFORE TAKING THIS EXAM.

The following Non-Disclosure Confidentiality Agreement (the "Agreement") sets forth the terms and conditions of your use of the exam materials as defined below.

The Disclosure to you of this Exam and any questions, answers, worksheets, computations, drawings, diagrams, or any communications, including verbal communication by any party, regarding or related to the Exam and such Exam Materials and any derivatives thereof is subject to the Terms and Conditions of this Agreement.

You understand, acknowledge and agree:

- That the questions and answers of the Exam are the exclusive and confidential property of Brocade and are protected by Brocade intellectual property rights;
- That you may not disclose the Exam questions or answers or discuss any of the content of the Exam Materials with any person, without prior approval from Brocade;
- Not to copy or attempt to make copies (written, photocopied, or otherwise) of any Exam Material, including, without limitation, any Exam questions or answers;
- Not to sell, license, distribute, or give away the Exam Materials, questions, or answers;
- You have not purchased, solicited or used unauthorized (non-Brocade sanctioned) Exam Materials, questions, or answers in preparation for this exam;
- That your obligations under this Agreement shall continue in effect after the Exam and, if applicable, after termination of your credential, regardless of the reason or reasons for terminations, and whether such termination is voluntary or involuntary.

Brocade reserves the right to take all appropriate actions to remedy or prevent disclosure or misuse, including, without limitation, obtaining an immediate injunction. Brocade reserves the right to validate all results and take any appropriate actions as needed. Brocade also reserves the right to use any technologies and methods for verifying the identity of candidates. Such technology may include, without limitation, personally identifiable information, challenge questions, identification numbers, photographic information, and other measures to protect against fraud and abuse.

Neither this Agreement nor any right granted hereunder shall be assignable or otherwise transferable by you.

By clicking on the "A" button ("YES, I AGREE"), you are consenting to be bound by the terms and conditions of this agreement and state that you have read this agreement carefully and you understand and accept the obligations which it imposes without reservation. You further state that no promises or representations have been made to induce agreement and that you accept this agreement voluntarily and freely.

A. YES, I AGREE

B. NO, I DO NOT AGREE