IPv6

IPv6 is an important topic – and not just for the exam. The growth of web-based services and diminishing IPv4 addressing will continue to push organizations towards IPv6, especially on web-facing networks.

Basics

IPv4 addresses are 32 bits long and are represented in dotted-decimal format. IPv6 addresses are 128 bits and are in hexadecimal format.

The first 64 bits of an IPv6 address are reserved for the network portion and the last 64 bits are used for the host portion.

IPv6 Shorthand

The ability to shorten IPv6 addresses is very important to understand because it makes reading and writing them much easier.

There are two ways to condense an IPv6 address:

1. Leading zeros can be removed in any section.
For example, 0021:0001:0000:030A:0000:0000:0000:0987E can be abbreviated as: 21:1:0:30A:0:0:0:987E

2. Sequential sections of all zeros can be shortened to a single double colon.

This can only be used once per address. Using the same example address above, it can be further shortened to: 21:1:0:30A::987E

Unicast, Multicast, & Anycast

Unicast

Unicast is for sending traffic to a single interface. In IPv6 there are actually two different unicast types, *global unicast* and *link-local unicast*.

Multicast

Unlike IPv4, IPv6 addressing does not support broadcasts. Instead, it has replaced it with multicast (which is a more efficient variation). This is used for sending traffic to a group of devices.

Anycast

IPv6 supports another new packet type – anycast. Anycast allows the same address to be used on multiple devices for load balancing and redundancy. Technically, it is used for sending traffic to the nearest interface in a group. While multiple devices may be running the same anycast address, only one will be used per packet sent.

Address Assignment

There are three different ways devices are assigned an IPv6 address: manual configuration, stateless autoconfiguration, or DHCPv6.

Manual Address Configuration

The first thing to know about manual IPv6 address configuration is that addresses assigned to a router interface use the *address/prefix-length* notation instead of the*address mask* notation. This is so much easier than typing 255.255… after every IPaddress!

Also, make sure you first enable IPv6 routing with the ***ipv6 unicast-routing***global configuration command. Use the **ipv6 address *ipv6-address/prefix-length*** command to assign an address.

An example of an interface configured with an IPv6 address:

```
R1# conf t
R1(config)# ipv6 unicast-routing
R1(config)# int gig 1/1
R1(config-if)# ipv6 address 21:1:0:30A::987E/64
```

Manual Network Assignment

Another way to manually configure an IPv6 address is to configure the network and allow the host portion to be autopopulated based on the device's MAC

address. This can work well because MAC addresses are 64 bits long – the exact same length as the host portion of an IPv6 address!

An example configuration with the network portion defined:

R1(config)# int gig 1/1
R1(config-if)# ipv6 address 21:1:0:30A::/64

Note: Some systems have a 48 bit MAC address. In this case, it flips the 7th bit and inserts 0xFFEE into the middle of the MAC address. This modified version is called an *EUI-64 address*. To do this, add the keyword ***eui-64*** to the end of the ipv6 address statement.

Stateless Autoconfiguration

Stateless autoconfiguration allows a device to self-assign an IP address for use locally without any outside information. Remember that interfaces using IPv6 will often have more than one IPv6 address assigned, and in this case statelessautoconfiguraiton will generate a link-local address in addition to any other manually assigned addresses. Link-local addresses are created using the prefix FE80:: and appending the device's MAC address. Since every MAC address should be unique, it works well for auto-generated local IP addresses. Link-local addresses are not routable within packets and are used for administrative purposes within the local segment. For example, most IGPs use link-local addresses for neighbor relationships and the link-local address is listed as the next-hop address in the routing table.

Once a router has created an IPv6 link-local address using stateless autoconfiguration, it uses NDP to make sure it is actually unique within the local network. NDP stands for Neighbor Discovery Protocol and uses ICMP packets as part of the neighbor discovery process.

To configure stateless autoconfiguration, use the ***ipv6 address autoconfig*** command.Example:

R1(config)# int gig 1/1
R1(config-if)# ipv6 address autoconfig

IPv6 Routing

Static Routes

The configuration for IPv6 static routes is identical to IPv4, except for the *ipv6 route* keywords instead of *ip route*. Other than that, it is exactly the same.

An example of a static IPv6 default route:

R1(config)# ipv6 route ::/0 serial1/1

An example of an IPv6 static route with a next-hop address:

R1(config)# ipv6 route 2003:2:1:A::/64 2003:2:1:F::1

To view the IPv6 routes in the routing table, use the command **show ipv6 route**.

IPv6 EIGRP

There are many differences in the way EIGRP is configured for IPv6.

- It still sends hellos out every 5 seconds to its neighbors, but when running EIGRP with IPv6 addresses it uses the multicast address FF02::A.
- EIGRP messages are exchanged using the link-local address as the source address. Perhaps the biggest difference is that there is no *network* command. Instead, EIGRP routing is enabled on each participating interface.
- Also, EIGRP running IPv6 requires a router ID be configured. The format is that of an IPv4 address - 32 digits and it can be a private address (non-routable) with no issues.
- The last major change is that the EIGRP process starts in the shutdown state. You have to issue a no shut to bring it up on the router.

To configure IPv6 EIGRP:

R1(config)# ipv6 unicast-routing
R1(config)# ipv6 router eigrp *AS*
R1(config-rtr)# router-id *ipv4-address*
R1(config-rtr)# no shut
R1(config-rtr)# exit
R1(config)# interface *type number*
R1(config-if)# ipv6 eigrp *AS*

OSPFv3

OSPFv3 is an updated version of OSPF designed to accommodate IPv6 natively. Most of the configuration and function is identical to its predecessor, but a few changes were made starting with messaging.

- OSPFv3 uses the multicast address FF02::5 and FF02::6, but like EIGRP it now uses its link-local address as the source address in advertisements.
- It is possible to run multiple instances of OSPFv3 on each link.
- Like the IPv6 implementation of EIGRP, a 32 bit router ID must be manually created. It will not automatically create one based on highest loopbackor interface address. The RID that is assigned will then be used to determine the DR and BDR on a segment (highest wins).
- OSPFv3 has dropped it's native authentication options. Instead, it relies on the underlying authentications built into IPv6, like IPSec.

Configuration

The configuration is now done on each individual interface. The following is an example configuration:

```
R2(config)# ipv6 unicast-routing
!
R2(config)# ipv6 router ospf 100
R2(config-rtr)# router-id 10.10.10.1
R2(config-rtr)# area 1 stub no-summary
R2(config-rtr)# exit
!
R2(config)# interface gig1/1
R2(config-if)# ipv6 address 2003:2:1:2::1/64
R2(config-if)# ipv6 ospf 100 area 0
R2(config)# interface gig1/2
R2(config-if)# ipv6 address 2003:2:1:A::1/64
R2(config-if)# ipv6 ospf 100 area 1
R2(config-if)# ipv6 ospf priority 30
```

MP-BGP

MP-BGP, or multiple protocol BGP, was outlined in RFC 2858 and includes extensions to the original BGP standard that allows support for other protocols – one of which is IPv6! The command *address-family* was added to specify which new protocol functionality is being configured and is used when applying IPv6 addressing.

Like EIGRP and OSPFv3, an IPv4 address must be configured as a router ID. The BGP configuration is not done at the interface level, it still is done in router configuration mode. The major difference is that neighbors must be first defined under router BGP configuration mode and then "activated" under IPv6 address-family mode submode. Networks and other parameters are also configured under IPv6 address-family mode submode.

Configuration

```
R3(config)# ipv6 unicast-routing
R3(config)# router bgp 600
R3(config-rtr)# router-id 10.10.10.10
R3(config-rtr)# neighbor 2003:76:1:1::10 remote-as 700
R3(config-rtr)# address-family ipv6 unicast
R3(config-rtr-af)# neighbor 2003:76:1:1::10 activate
R3(config-rtr-af)# network 2003:2:2::/48
R3(config-rtr-af)# exit
R3(config-rtr)# exit
```

Migrating to IPv6

Three options exist for transitioning from IPv4 to IPv6: dual stack, tunneling, or NAT.

Dual Stack

This involves running IPv4 alongside IPv6 on the same system.

Tunneling

This option allows you to encapsulate IPv6 traffic within an IPv4 header.

NAT

A new network translation extension, NAT-PT allows IPv6-to4 translation.

Dual Stack

Using a dual-stack transition allows servers, clients, and applications to be slowly moved to IPv6. Both protocols can run concurrently and neither communicating with the other. If both IPv4 and IPv6 are running on a server for example, IPv6 will be used.

Configuration Example

```
R1# config t
R1(config)# ipv6 unicast-routing
R1(config)# ipv6 cef
!
R1(config)# interface serial1/0/1
R1(config-if)# ip address 192.168.1.1
R1(config-if)# ipv6 address 2001:1:3:1::1/64
```

IPv6 Tunneling

Dual-stacking IPv4 alongside IPv6 on systems works well, but it requires most of your infrastructure to support IPv6. In many cases, the network core does not support IPv6 or it has not been implemented. IPv6 tunnels solve this problem by allowing IPv6 islands to exist and bridges them over IPv4 systems.

Because IPv6 tunnels provide virtual IPv6 connectivity through an IPv4 transport, it does not matter what specific IPv4 transport is used. The only requirement is that there is end-to-end IPv4 connectivity between both ends.

Manual Tunnels

The tunnels discussed here are from one router to another. The source router (RouterA) encapsulates the IPv6 traffic in IPv4 headers, then forwards it to the

other end of the tunnel (Router B). Router B then decapsulates the packets and forwards them on to their destination using native IPv6.
Manual IPv6 tunnels are easy to configure using the ***tunnel mode ipv6ip***command. Using the Router A/B example above, the configuration on Router A would look something like this:

```
RouterA(config)# interface tunnel0
RouterA(config-if)# ipv6 address 2001:2:0:7::/64
RouterA(config-if)# tunnel source 10.1.1.1
RouterA(config-if)# tunnel destination 10.3.3.1
RouterA(config-if)# tunnel mode ipv6ip
RouterA(config-if)# exit
```

GRE Tunnels

First, GRE tunnels are the default tunnel method on Cisco routers. GRE tunnels are very flexible and work over most protocols.
The configuration is exactly the same as the manual configuration example above, but you do not have to specify the tunnel mode. Also, routing protocols can be enabled on GRE tunnel interfaces just as if they were physical interfaces.

6to4 Tunnels

6to4 tunnels are similar to the manual tunnel, but set up the tunnel dynamically.

6to4 tunnels use 2002::/16 IPv6 addresses in front of the 32 bit IPv4 address of the edge router – creating a 48 bit prefix. Each router on both sides of the tunnel needs a route to its peer. They only support static and BGP routes, so be careful.

Configure the tunnel as if it was a manual tunnel, using the IPv4 address as the source, but don't enter a destination.The tunnel requires an IPv6 address using the method just described. Finally, use the command ***tunnel mode ipv6ip 6to4***.