IPv6 ADVANCED TRAINING COURSE

This two-day course provides an in-depth coverage of key topics regarding the deployment of IPv6 in different types of networks. The participant will gain insight into industry best practices and develop a high-level understanding of the different stages of IPv6 rollout.

Goals

- Get the green light from decision makers and develop a project plan for your IPv6 deployment project
- Identify the different stakeholders involved in the deployment of IPv6
- Recognise the needed technical skills and where to find training resources
- Develop an IPv6 transition strategy for the IPv6 deployment project
- Acquire a solid understanding of the different technical options available for each network type and how to choose from them
- Configure IPv6 with OSPF, BGP, DNS and test multihoming in a LAB network

Pre-Requisites

It is assumed you are familiar with:

- IPv4
- IPv6 Fundamental level: IPv6 notation, header information, DHCPv6, NDP
- IP networking basics: concepts and protocols TCP/IP, DNS, DHCP, OSPF, IS-IS, BGP),
- For the Labs: Basic understanding of CLI and command line tools.

Course Content

- Introduction
- Get Approval Phase
  - *Get the green light*
  - *Project Planning*
- Get Ready Phase
  - *Create an addressing Plan*
  - *Get IPv6 Training*
  - *Build a test environment*
  - *Adopt a transition strategy*
- Deploy Phase
  - *Backbone network*
  - *Access network*
  - *Enterprise network*
  - *Datacenter network*
  - *Cloud network*
- Manage
  - *Troubleshooting*
  - *Monitoring*
- Tips and Tricks

- Training Material: **https://www.ripe.net/training-material**
- Feedback Survey: **https://www.ripe.net/feedback/av6/**
- Contact us: **learning@ripe.net**
- Learning and Development Services: **www.ripe.net/support/training**

Training Material

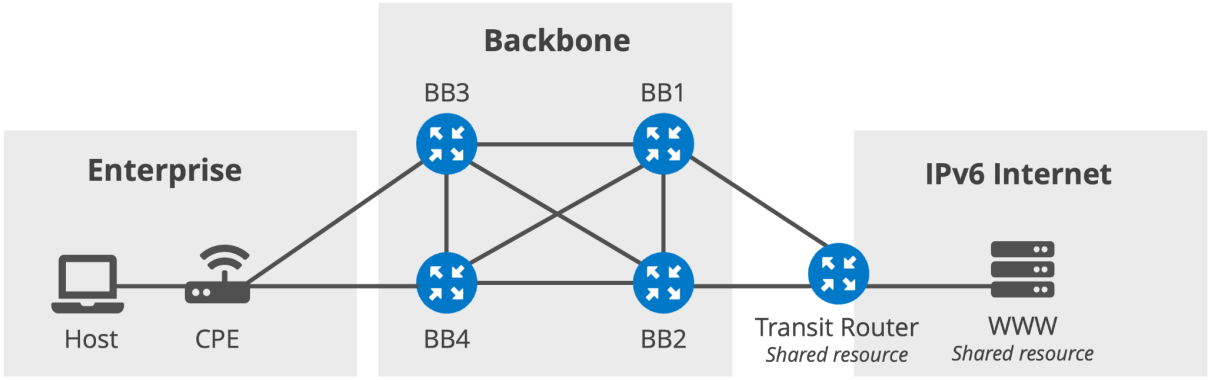
RIPE NCC

IPv6 Advanced Lab Guide

Introduction And Topology

You are hired as the new network engineer of ISP Sandbox Inc. where you'll be deploying IPv6 soon with your colleagues. As part of the deployment project you are tasked with preparing a lab environment as a small replica of your live network and testing your equipment for IPv6 readiness.

Here is the network topology that you are expected to work on.



Luckily lab technicians prepared the physical connections and the IPv4 related configurations for you.They also assigned the IPv6 addresses to the interfaces. You'll need to do the tasks, configure and test the protocols in the list below.

1. Test an IPAM tool for the registration of your IPv6 addressing plan.

2. Configure OSPFv3 in the backbone of the lab network.

3. Configure iBGP inside the backbone of the lab network, and eBGP over the connection to the transit router.

4. Configure multihomed connection for the CPE in the enterprise network for simulating the "Multihomed IPv6 connection using provider-assigned addresses" problem and seeking solution by using your own address space.

5. Configure DNS to support IPv6

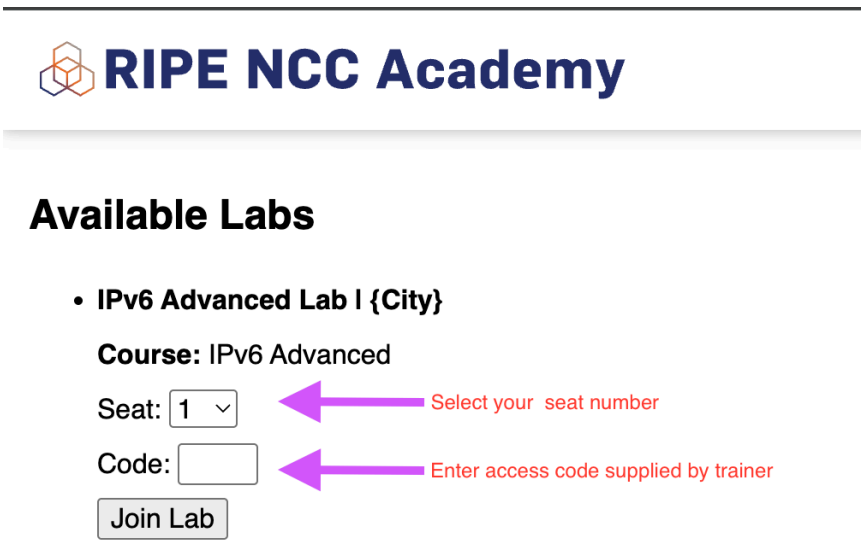
How to access the lab environment

The trainer will also assign you a number (Number **X**) which will be your **Seat** to access the lab environment. The trainer will also provide you with a **code** to access the IPv6 Advanced lab

Go to https://workbench.ripe.net, log in using your RIPE NCC access account. If you don't have one, please register on this page.

After logging in, find the **"IPv6 Advanced"** lab, then:

1. Select your **Seat** number from the dropdown menu
2. Enter the **Code** provided by the trainer.
3. Click on the **Join lab** button to access the lab.



You will see a landing page with the lab network diagram, and tabs to access the "**IPAM**", "**Backbone**", "**Enterprise Customer**" and "**Datacenter**" nodes in the lab.

The routers in the lab network are running an open-source routing software, FRR (Free Range Routing) version 8.4.2 (https://frrouting.org).

Lab Activity 1: IPv6 addressing plan using phpIPAM

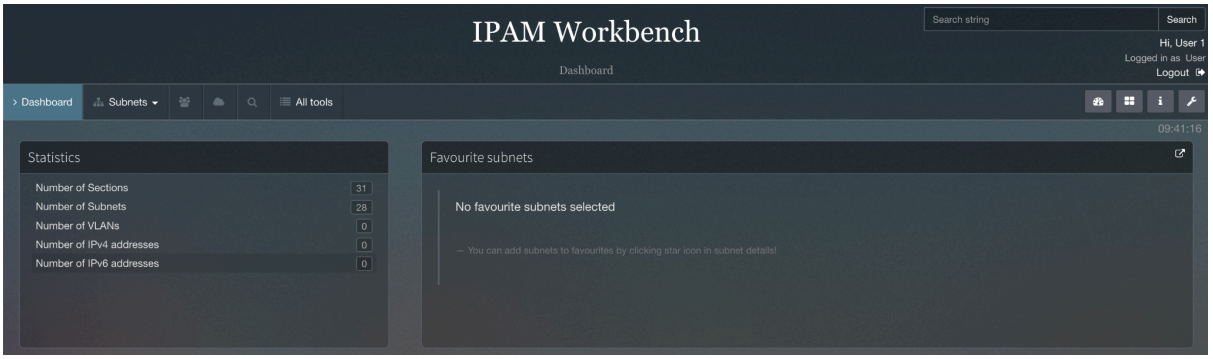
In this lab activity you'll create and register your IPv6 addressing plan in one of the most known and used IPAM tools; phpIPAM.

Activity 1.1: Creating a subnet in phpIPAM for your new allocation

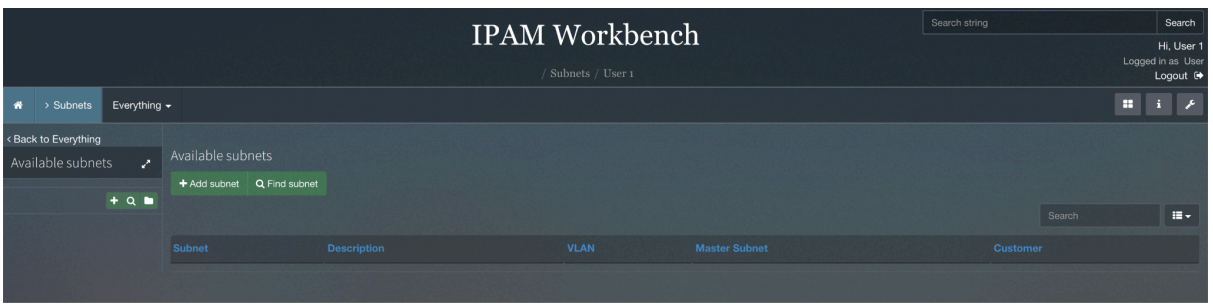
Your organisation has received a new /40 IPv6 allocation from the RIPE NCC. And, you've been tasked with adding this new allocation into your IPAM tool which is phpIPAM.
Important Note: Please keep in mind that minimum IPv6 allocation size in RIPE NCC region is /32. We use /40 just for the sake of simplicity here.

Go to the login page using your favorite browser. And enter the credentials.
Username: Your number provided by the trainer
Password: Password will be provided by the trainer

After logging in you'll see the landing page in the picture below

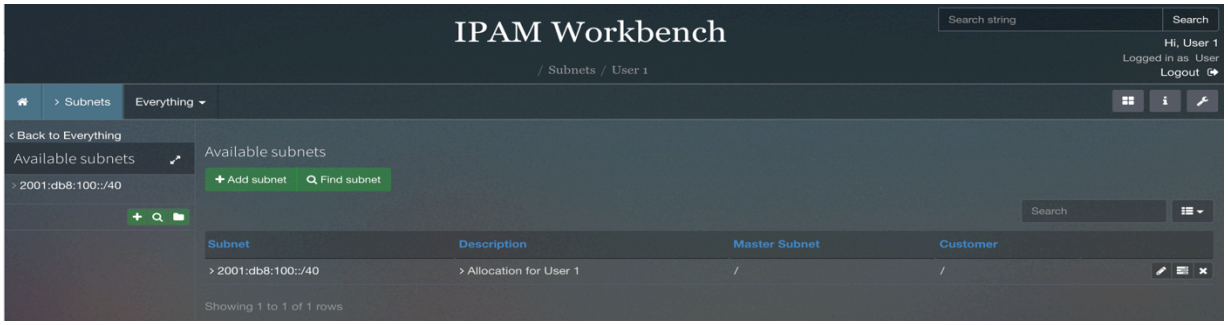


Click on the "Subnets" button and then click "All Sections"
Here in this page you will see your user name like "User X". Click on your username to reach your available resources. The page you see will be similar to the one below.



At this point, you don't have any registered IP resources yet. Your first task is registering your allocation here.

1. Click on the “**Add Subnet**” button.
2. Write the subnet allocated for your user number in the format:
   2001:db8:**U**00::/40
   U is your user number. If it is 3 your allocation is 2001:db8:**3**00::/40, and if it is 24 then your allocation is 2001:db8:**24**00::/40
3. **Description** is “Allocation for User **U**”
4. All the other fields can be left as default.
5. After a successful creation you’ll see a screen similar to the one below.
6.



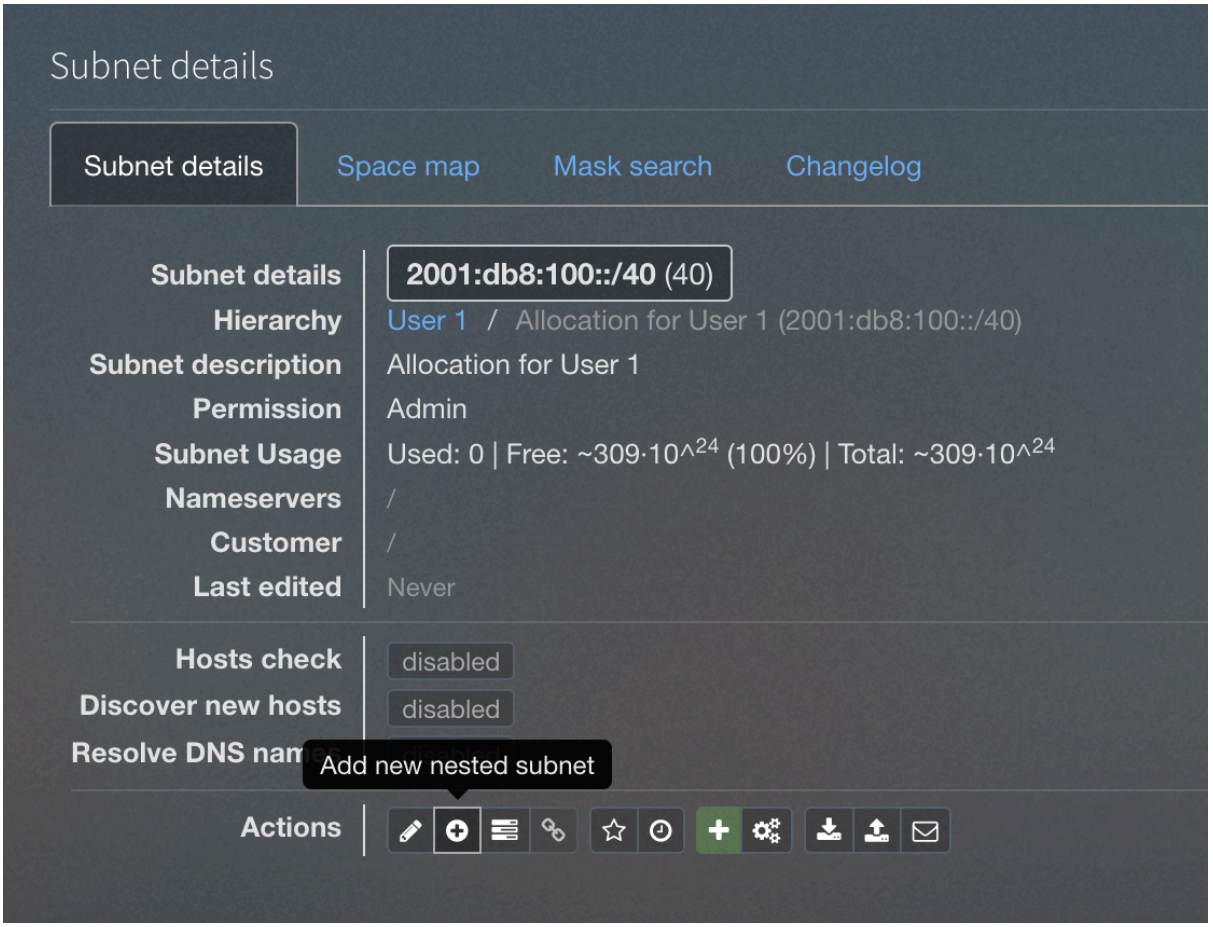
Congratulations! You’ve just registered your allocation in your IPAM tool.

Activity 1.2: Creating nested subnets for different parts of your network

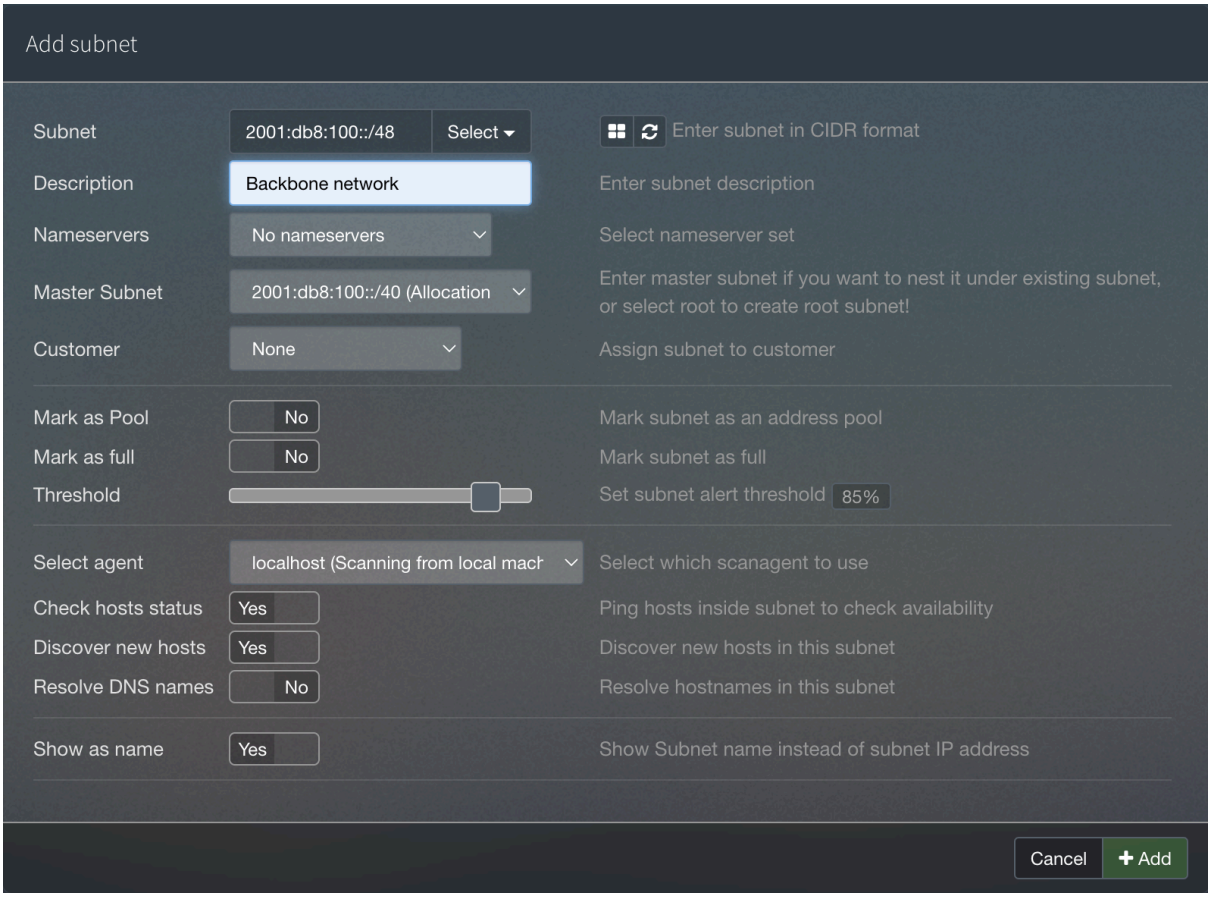
Now you need to create new subnets under your allocation to be used in different parts of your network such as: backbone network, fixed broadband network, etc.

Activity 1.2.1: Create a subnet for the backbone network

1. Click on the /40 subnet you’ve created in the previous exercise.
2. Find the “**Actions**” row and click on the plus sign in the circle. When you hover your mouse over it, the text appears saying “**Add new nested subnet**”. See the example below.



3. In our lab environment, we use /48 subnets for each part of the network. And our addressing template is:
   2001:0db8:**U**NN::/48
   Where "NN" represents different parts of your network. For backbone NN equals 00.
4. **Description** is "Backbone network"
5. You want to get notifications when this subnet reaches a certain threshold and you set this "**Threshold**" as 85%.
6. You also want to check the host status and discover the new hosts in this subnet. For these features you need to toggle the buttons: "**Check hosts status**", "**Discover new hosts**".
7. Lastly, toggle the button "**Show as name**" to view this subnet with the name instead of the subnet IP address.
   You see an example for User 1 below.



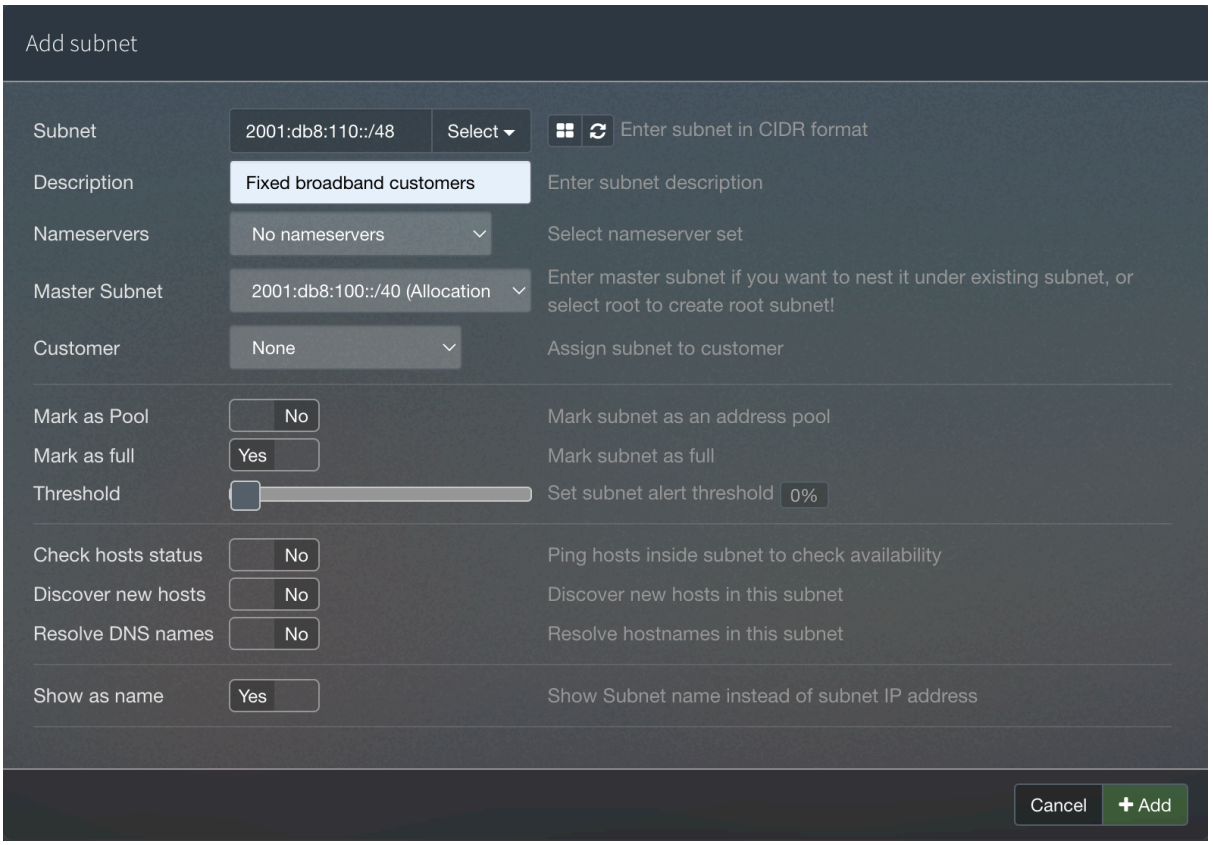
8. After a successful creation you'll see a screen similar to the one below.

Now, you've created your backbone subnet in your IPAM tool.

Activity 1.2.2: Create a subnet for the fixed broadband network

1. Click on the /40 subnet you've created in the first exercise.
2. Find the "**Actions**" row and click on the plus sign in the circle. When you hover your mouse over it, the text appears saying "**Add new nested subnet**".

3. In our lab environment, we use /48 subnets for each part of the network. And our addressing template is:
   2001:0db8:**U**NN::/48
   Where "NN" represents different parts of your network. For the fixed broadband network NN equals 10.
4. **Description** is "Fixed broadband customers"
5. You want to mark this subnet as full as you already assigned all of the available /64s inside this subnet to your broadband customers. For this toggle the button named "**Mark as full**"
6. Toggle the button "**Show as name**" to view this subnet with the name instead of the subnet IP address.



7. After a successful creation you need to see the "**Fixed broadband customers**" subnet in the column on the left side of the page.

Now, you've created your fixed broadband customers subnet in your IPAM tool and marked it as full.

Activity 1.2.3: Create a subnet for one of your customer networks

1. Click on the /40 subnet you've created in the first exercise.
2. Find the "**Actions**" row and click on the plus sign in the circle. When you hover your mouse over it, the text appears saying "**Add new nested subnet**".
3. In our lab environment, we use /48 subnets for each part of the network. And our addressing template is:
   2001:0db8:**U**NN::/48
   Where "NN" represents different parts of your network. For this customer network NN equals 21.

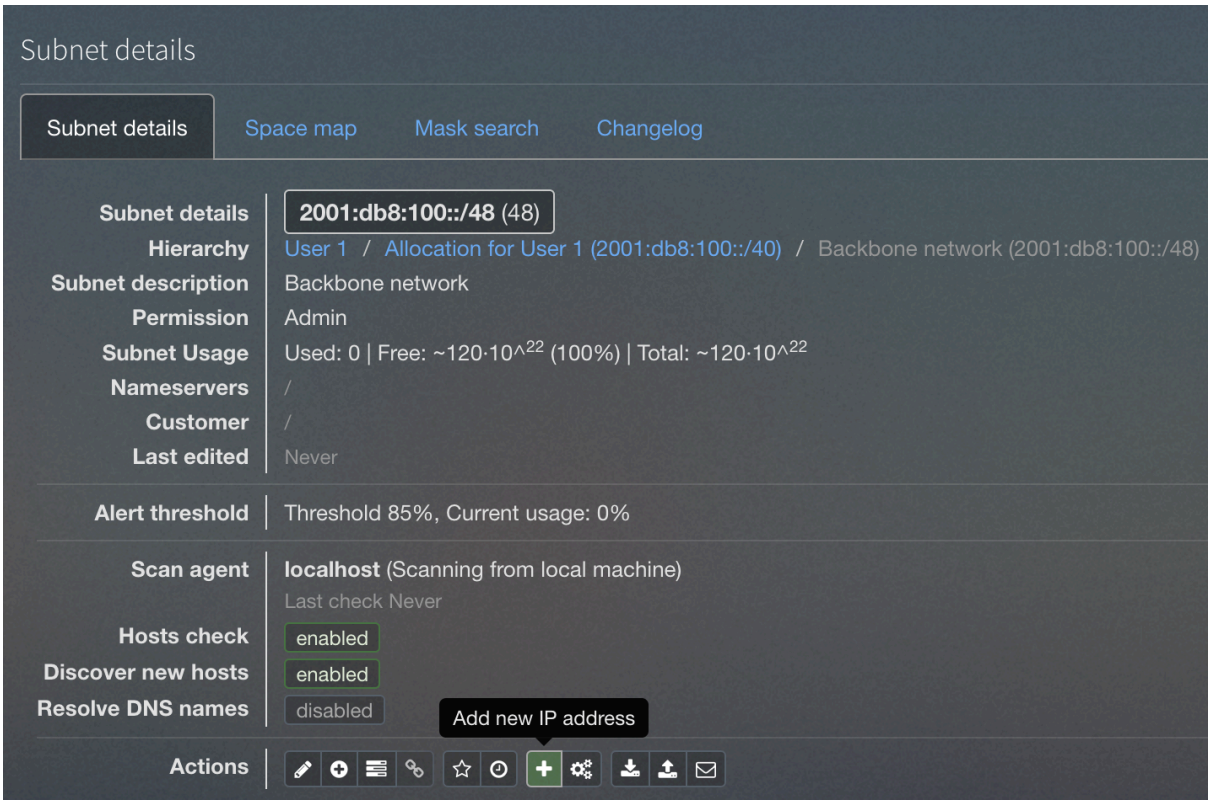
4. **Description** is “Customer-1 Network”
5. As you want to assign this subnet to one of your customers you’ll make this association by selecting your customer which was pre-created for you. In the “**Customer**” dropdown box select the customer name created for your user number: “**Customer of User U**”
6. Toggle the button “**Show as name**” to view this subnet with the name instead of the subnet IP address.
7. After a successful creation you need to see the “**Customer-1 Network**” subnet in the column on the left side of the page.

Now, you’ve created your customer’s network in your IPAM tool.

Activity 1.3: Creating IP addresses inside the subnets & searching them

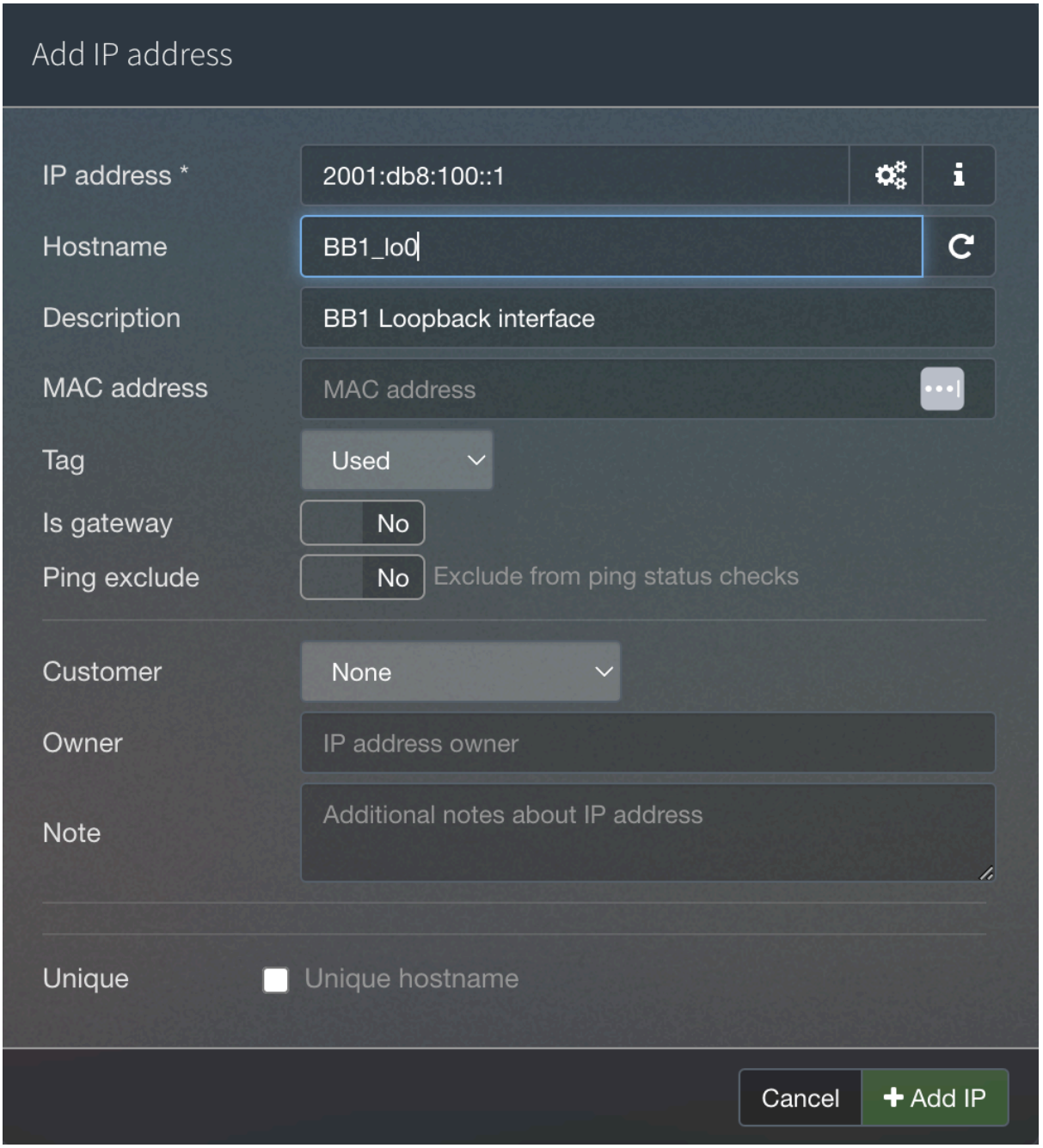
In this exercise we’ll create an IP address inside the backbone network and search for it.

1. Click on the “**Backbone network**” from the menu on the left side of the page.
2. Find the “**Actions**” row and click on the green plus sign. When you hover your mouse over it, the text appears saying “**Add new IP address**”. See the example below.



3. Fill in the “**IP address**” field with the address “2001:db8:U00::1”
4. Fill in the “Hostname” field with the text “BB1_lo0” representing the loopback interface name of backbone 1 router.
5. Fill in the “Description” field with the text “BB1 Loopback interface”.
6. All the other fields can be left as default.

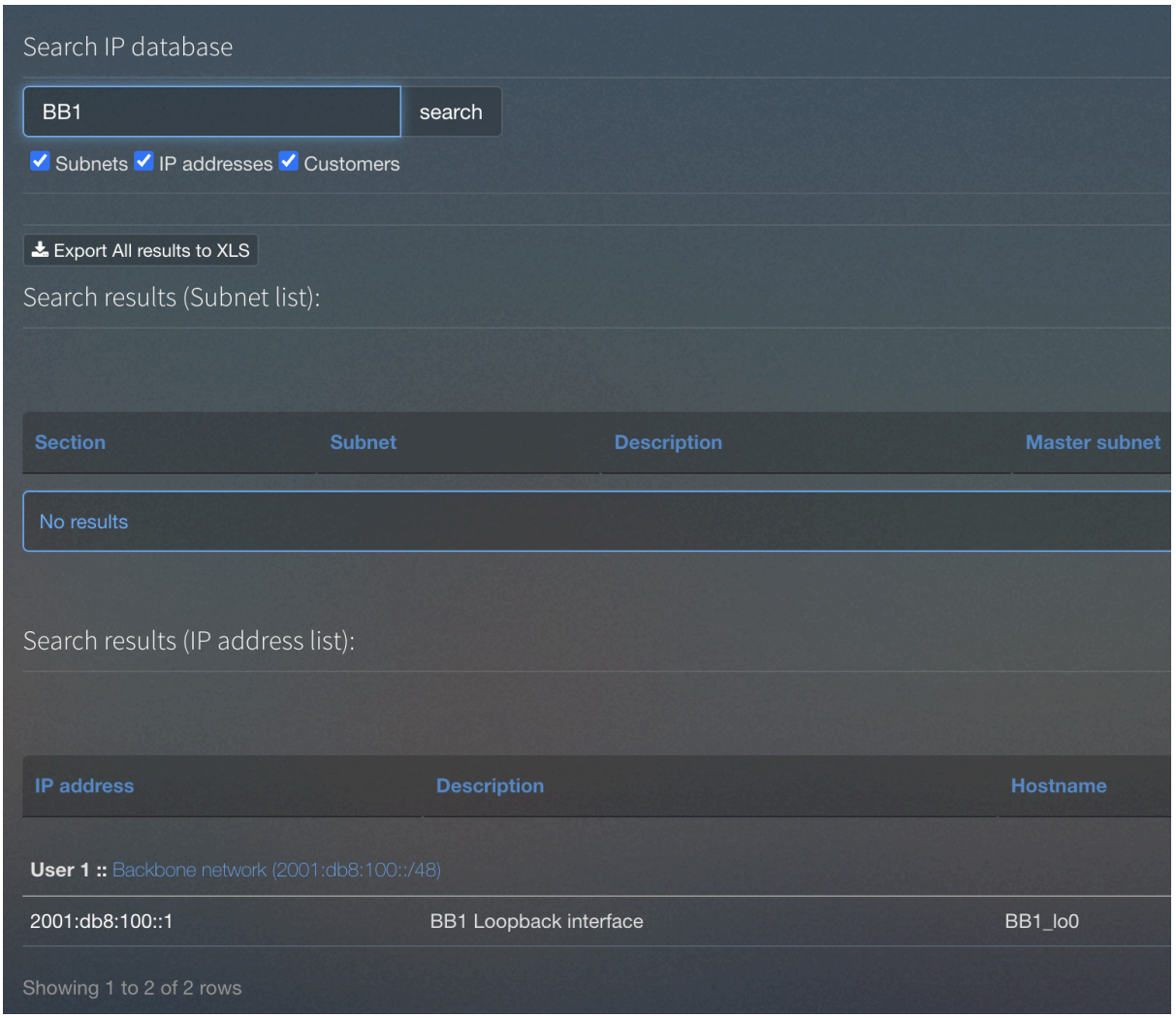
You see an example for User 1 below.



After a successful creation you need to see this address in the “**IP addresses in subnets**” field. And you can check the (offline/online) status of this address.

You can also search this address by its “IP”, “hostname” or “description”.

Now you can try searching it in the “Search bar” on the top right corner of the page using the search term “BB1”.
After the search, you’ll see the results similar to the one below.



Activity 1.4: Checking the availability of smaller subnets

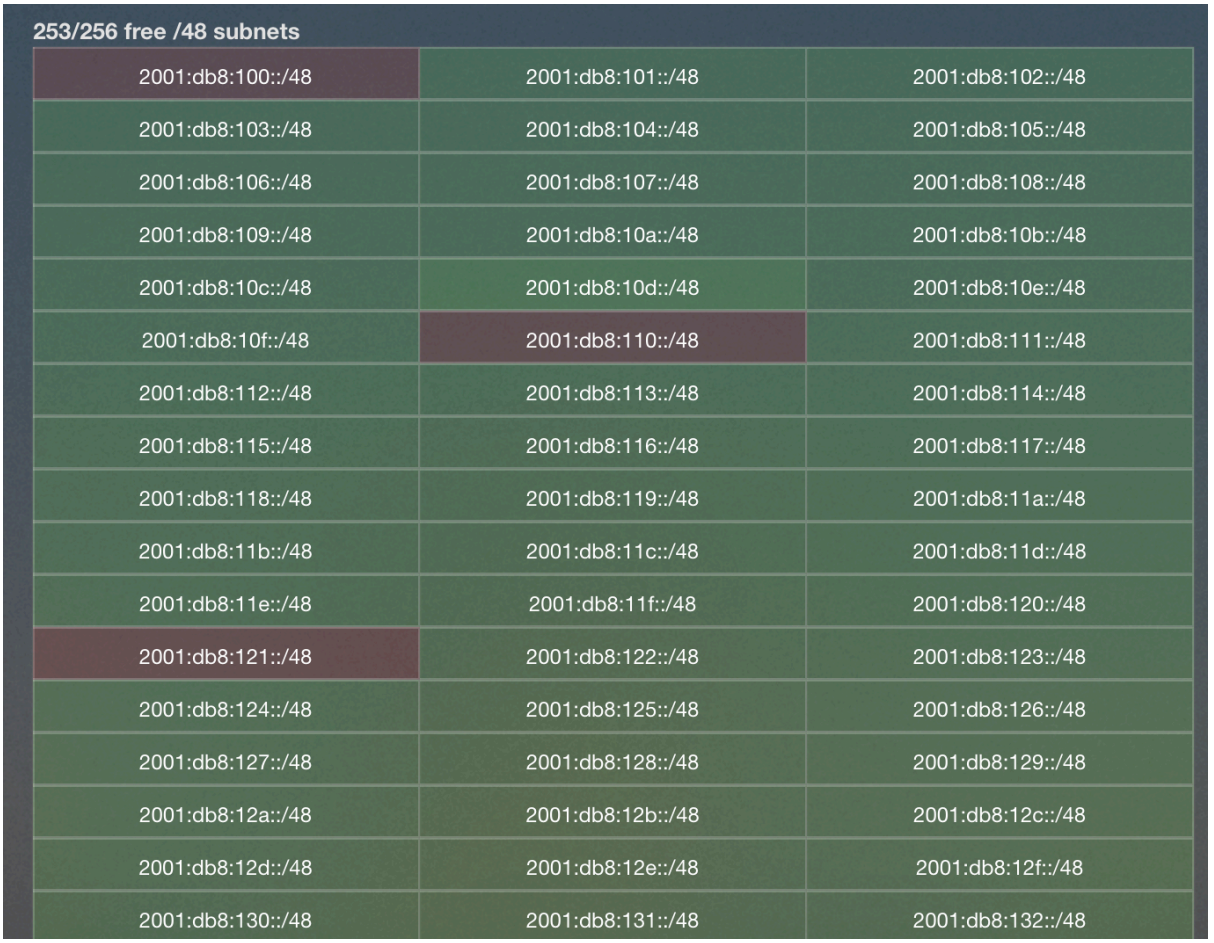
With the phpIPAM tool you can also easily check the availability of the subnets in your allocation. Now you’ll check the available /48 subnets in your organization’s allocation with a few clicks.

1. Click on the “Subnets” button and then click “All Sections”
2. Click on your username to reach your available resources.
3. Click on your parent allocation which is a /40 subnet.
4. Here in this view you can check “**Subnet Usage**” and see that 99.61% of your allocation is still free.

You can also check the individual status of /48s available in this /40.

5. Click on the “Space Map” tab and scroll down to /48 subnets.
6. Here you can see the free and used /48 subnets color coded. Greens free, reds used.

See an example output below.



253/256 free /48 subnets

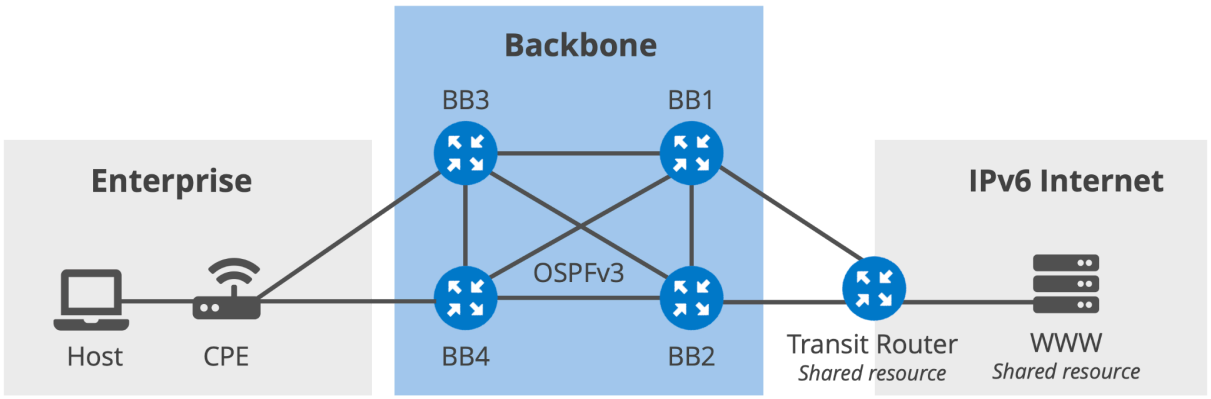
2001:db8:100::/48	2001:db8:101::/48	2001:db8:102::/48
2001:db8:103::/48	2001:db8:104::/48	2001:db8:105::/48
2001:db8:106::/48	2001:db8:107::/48	2001:db8:108::/48
2001:db8:109::/48	2001:db8:10a::/48	2001:db8:10b::/48
2001:db8:10c::/48	2001:db8:10d::/48	2001:db8:10e::/48
2001:db8:10f::/48	2001:db8:110::/48	2001:db8:111::/48
2001:db8:112::/48	2001:db8:113::/48	2001:db8:114::/48
2001:db8:115::/48	2001:db8:116::/48	2001:db8:117::/48
2001:db8:118::/48	2001:db8:119::/48	2001:db8:11a::/48
2001:db8:11b::/48	2001:db8:11c::/48	2001:db8:11d::/48
2001:db8:11e::/48	2001:db8:11f::/48	2001:db8:120::/48
2001:db8:121::/48	2001:db8:122::/48	2001:db8:123::/48
2001:db8:124::/48	2001:db8:125::/48	2001:db8:126::/48
2001:db8:127::/48	2001:db8:128::/48	2001:db8:129::/48
2001:db8:12a::/48	2001:db8:12b::/48	2001:db8:12c::/48
2001:db8:12d::/48	2001:db8:12e::/48	2001:db8:12f::/48
2001:db8:130::/48	2001:db8:131::/48	2001:db8:132::/48

Activity 1 Summary

In this activity:

- You created different types of subnets in the IPAM tool
- You registered IP addresses and learned to search them
- You learned how to check the availability of a certain size of subnets

Lab Activity 2: Configuring OSPFv3 inside the backbone network



Activity 2.1: Check existing OSPFv2 configuration

In this lab activity you'll be configuring OSPFv3 among the backbone routers BB1, BB2, BB3 and BB4. OSPFv2 for IPv4 addresses is already configured and ready for you.

First of all you'll start by checking the OSPFv2 configuration for IPv4. You'll do this with the command "`show run ospfd`". OSPFd is the name of the OSPFv2 process running for IPv4 on the FRR routers.
For example, if you type "show run ospfd" on router BB1, you will see a configuration similar to the one below:

```
interface lo
 ip ospf area 0
exit
!
interface to_bb2
 ip ospf area 0
 ip ospf network point-to-point
exit
!
interface to_bb3
 ip ospf area 0
 ip ospf network point-to-point
exit
!
interface to_bb4
 ip ospf area 0
 ip ospf network point-to-point
```

```
exit
!
interface to_dc
 ip ospf area 0
 ip ospf passive
exit
!
router ospf
 default-information originate
exit
```

As you can see in the running configuration, you have a very basic configuration for OSPFv2. The OSPF process is enabled with a single line command "`router ospf`" and interfaces are included in the OSPF Area 0 with the commands "`ip ospf area 0`".
As we don't want the routers to elect Designated Router (DR) / Backup Designated Router (BDR) on the point-to-point links (it is unnecessary for a link connecting two routers only) you mention the network type with the command "`ip ospf network point-to-point`".
You also need to add loopback interfaces into the OSPF topology as we want them to become reachable from the other routers. You'll use loopbacks to build the iBGP neighborship in the next lab activity.

For the interface on the BB1 router that is connected to the data center (which is an interface only on BB1 router), we configure it as passive. This is because you don't want to send hello packets out of this interface, as there are no other routers on this segment. However, you still want to advertise this network segment to all other backbone routers, so that it remains reachable. You'll do a similar IPv6 configuration for this interface.

In addition to this, you have the "`default-information originate`" command under ospf configuration on routers BB1 and BB2 where you have connections to the upstream provider. This command enables routers to generate default routes to the OSPF area they are in if they have a valid default route in their routing table (ie. it might be received via eBGP).

Note: *We don't use the "always" keyword in the default route origination as we don't want to create a routing loop between the BB1 and BB2 routers when we don't have a default route originating from the transit router. We do still use default-route origination in OSPF as a precautionary mechanism to limit the effect of a misoriginated eBGP default route from the customer networks. In our case, the misoriginated default route from the customer networks will only affect the router which receives it and will not take effect on the other backbone routers as OSPF has preferred over iBGP.*

Now you can check the neighbors on all of your backbone routers. Here is the example from **User1, Router1** (you should see a similar output when you check this on the other backbone routers and for the other users):

```
u0U-bb1# sh ip ospf neighbor
```

```
Neighbor ID     Pri State           Up Time         Dead Time Address         Interface
10.1.2.2          1 Full/-          5m06s             36.517s 10.1.21.2       to_bb2:10.1.21.1
10.1.3.3          1 Full/-          5m06s             39.226s 10.1.31.3       to_bb3:10.1.31.1
10.1.4.4          1 Full/-          5m06s             33.482s 10.1.41.4       to_bb4:10.1.41.1
```

Here you see that you have 3 neighbors on BB1 as expected and the state is “FULL”.
Are you sure that you have FULL OSPF neighborship on all of your backbone routers? You can check it with the same command.

Activity 2.2: Configure OSPFv3

After checking the initial configuration and the status of OSPFv2 you can start with the configuration of OSPFv3 which you’ll use for IPv6 addresses. You’ll configure OSPFv3 in a similar way.

As you’ll remember, OSPFv2 and OSPFv3 run in different processes on the routers. First you need to enable OSPFv3 process **on all BB routers** with the command:

```
configure terminal
router ospf6
end
```

And then you configure the default route origination on the routers connected to the transit network (**BB1 and BB2 only**) just like in OSPFv2.

```
configure terminal
router ospf6
default-information originate
end
```

The next step is enabling ospfv3 under the interfaces of **all the backbone routers**. You need to select only the interfaces facing towards other backbone routers.
X is the router number of which the interface is facing.

```
configure terminal
interface to_bbX
 ipv6 ospf6 area 0
 ipv6 ospf6 network point-to-point
end
```

And you’ll also need to configure OSPFv3 under the loopback interfaces on all four backbone routers (BB1, BB2, BB3 and BB4) as follows:

```
configure terminal
interface lo
 ipv6 ospf6 area 0
```

```
end
```

Note: We don't need the "`ipv6 ospf6 network point-to-point`" command under the loopback interfaces as the loopback interfaces are logical interfaces and not physically connected to any other device, so they are inherently considered point-to-point by OSPF.

Finally, **only on the BB1 router** you need to enable OSPFv3 for IPv6 in passive mode.

```
configure terminal
interface to_dc
 ipv6 ospf6 area 0
 ipv6 ospf6 passive
end
```

Remember that *you enable this interface under OSPFv2/OSPFv3 in passive mode because you don't want to send hello packets out of this interface, as there are no other routers on this segment. However, you still want to advertise this network segment to all other backbone routers, so that it remains reachable*

After configuring the interfaces you should be able to see an output like below on all backbone routers (BB1, BB2, BB3, BB4) with the command "`show ipv6 ospf6 neighbor`".

```
u0U-bb1# show ipv6 ospf6 neighbor
Neighbor ID     Pri    DeadTime    State/IfState         Duration I/F[State]
10.1.2.2          1    00:00:34     Full/PointToPoint    00:04:45 to_bb2[PointToPoint]
10.1.3.3          1    00:00:37     Full/PointToPoint    00:04:35 to_bb3[PointToPoint]
10.1.4.4          1    00:00:30     Full/PointToPoint    00:05:19 to_bb4[PointToPoint]
```

As you can see, all the backbone routers have OSPFv3 neighborship in "**Full**" state.
Did you notice that routers are still using IPv4 loopback addresses as the Router IDs? This is the default behavior of FRR software. And as we've learned in the course you can still use your IPv4 loopback addresses as the Router ID with OSPFv3 protocol.

And in the IPv6 routing table of User-1 BB1 router you'll see the populated OSPF routes indicated with the **O** letter. You'll have an output similar to the one below:

```
u0U-bb1# show ipv6 route
Codes: K - kernel route, C - connected, S - static, R - RIPng,
       O - OSPFv3, I - IS-IS, B - BGP, N - NHRP, T - Table,
       v - VNC, V - VNC-Direct, A - Babel, F - PBR,
       f - OpenFabric,
       > - selected route, * - FIB route, q - queued, r - rejected, b - backup
       t - trapped, o - offload failure

O   2001:db8:100::1/128 [110/10] is directly connected, lo, weight 1, 00:07:50
C>* 2001:db8:100::1/128 is directly connected, lo, 2d16h50m
O>* 2001:db8:100::2/128 [110/20] via fe80::a8c1:abff:fe33:5199, to_bb2, weight 1, 00:08:34
O>* 2001:db8:100::3/128 [110/20] via fe80::a8c1:abff:feb5:5c0b, to_bb3, weight 1, 00:08:19
O>* 2001:db8:100::4/128 [110/20] via fe80::a8c1:abff:fe7a:b810, to_bb4, weight 1, 00:08:09
O   2001:db8:100:12::/64 [110/10] is directly connected, to_bb2, weight 1, 00:07:50
C>* 2001:db8:100:12::/64 is directly connected, to_bb2, 2d16h50m
O   2001:db8:100:13::/64 [110/10] is directly connected, to_bb3, weight 1, 00:07:50
```

```
C>* 2001:db8:100:13::/64 is directly connected, to_bb3, 2d16h50m
O   2001:db8:100:14::/64 [110/10] is directly connected, to_bb4, weight 1, 00:07:50
C>* 2001:db8:100:14::/64 is directly connected, to_bb4, 2d16h50m
C>* 2001:db8:100:18::/64 is directly connected, to_transit, 2d16h50m
O>* 2001:db8:100:23::/64 [110/20] via fe80::a8c1:abff:fe33:5199, to_bb2, weight 1, 00:08:19
  *                               via fe80::a8c1:abff:feb5:5c0b, to_bb3, weight 1, 00:08:19
O>* 2001:db8:100:24::/64 [110/20] via fe80::a8c1:abff:fe33:5199, to_bb2, weight 1, 00:08:09
  *                               via fe80::a8c1:abff:fe7a:b810, to_bb4, weight 1, 00:08:09
O>* 2001:db8:100:34::/64 [110/20] via fe80::a8c1:abff:fe7a:b810, to_bb4, weight 1, 00:08:09
  *                               via fe80::a8c1:abff:feb5:5c0b, to_bb3, weight 1, 00:08:09
O   2001:db8:103::/64 [110/10] is directly connected, to_dc, weight 1, 00:07:50
C>* 2001:db8:103::/64 is directly connected, to_dc, 2d16h50m
C   fe80::/64 is directly connected, to_bb3, 2d16h50m
C * fe80::/64 is directly connected, to_bb4, 2d16h50m
C * fe80::/64 is directly connected, to_bb2, 2d16h50m
C * fe80::/64 is directly connected, to_transit, 2d16h50m
C>* fe80::/64 is directly connected, eth0, 2d16h50m
```

You can try to ping the other loopback addresses in the topology. Here you see the output of ping from BB1 to BB2 loopback address using the command `ping 2001:db8:U00::2`

```
u0U-bb1# ping 2001:db8:U00::2
PING 2001:db8:U00::2 (2001:db8:U00::2): 56 data bytes
64 bytes from 2001:db8:U00::2: seq=0 ttl=64 time=0.067 ms
64 bytes from 2001:db8:U00::2: seq=1 ttl=64 time=0.096 ms
64 bytes from 2001:db8:U00::2: seq=2 ttl=64 time=0.090 ms
64 bytes from 2001:db8:U00::2: seq=3 ttl=64 time=0.115 ms
^C
--- 2001:db8:100::2 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.067/0.092/0.115 ms
```

Activity 2.3: Compare OSPFv2 and OSPFv3 Databases

Now, let's check the details of LSAs advertised. As you'll remember in IPv6 we have physical link and IP address information separation. To see this in the database you can compare the **Router LSAs** in databases of OSPFv2 and OSPFv3.

You'll first check the database of OSPFv2 on BB1 for a special type of messages, you'll check **router LSA**s. (These example outputs are taken from user1 routers, so the addresses might be different from what you see in your own outputs).

```
u0U-bb1# show ip ospf database router adv-router 10.U.1.1

       OSPF Router with ID (10.1.1.1)


                Router Link States (Area 0.0.0.0)

  LS age: 1034
  Options: 0x2  : *|-|-|-|-|-|E|-
  LS Flags: 0x3
  Flags: 0x2 : ASBR
  LS Type: router-LSA
  Link State ID: 10.1.1.1
  Advertising Router: 10.1.1.1
```

```
  LS Seq Number: 80000092
  Checksum: 0xb68e
  Length: 120

   Number of Links: 8

    Link connected to: Stub Network
     (Link ID) Net: 10.1.1.1
     (Link Data) Network Mask: 255.255.255.255
      Number of TOS metrics: 0
       TOS 0 Metric: 0

    Link connected to: another Router (point-to-point)
     (Link ID) Neighboring Router ID: 10.1.2.2
     (Link Data) Router Interface address: 10.1.21.1
      Number of TOS metrics: 0
       TOS 0 Metric: 10

    Link connected to: Stub Network
     (Link ID) Net: 10.1.21.0
     (Link Data) Network Mask: 255.255.255.0
      Number of TOS metrics: 0
       TOS 0 Metric: 10


----The rest of the output is omitted------
```

As you can see in this output, in the OSPFv2 database, in the Router LSAs you see information for both physical interfaces connected to this router specified with the keywords "**`Router Interface address`**" and the IP address information specified with the "**`Net`**" and "**`Network Mask`**" fields.

Note: The first link information is about the loopback interface and there is no neighboring router on this logical interface. That's why we check the second and the third link information.

When you check the same for IPv6 with the command mentioned below, you don't see any IPv6 specific information and addresses. There is only physical information.

```
u0U-bb1# show ipv6 ospf6 database router adv-router 10.U.1.1 detail

        Area Scoped Link State Database (Area 0)

Age:  811 Type: Router
Link State ID: 0.0.0.0
Advertising Router: 10.1.1.1
LS Sequence Number: 0x80000004
CheckSum: 0xa8aa Length: 72
Duration: 00:13:30
    Bits: -------- Options: --|-|--|-|-|--|R|-|--|E|V6
    Type: Point-To-Point Metric: 10
    Interface ID: 0.0.21.195
    Neighbor Interface ID: 0.0.21.194
    Neighbor Router ID: 10.1.2.2
    Type: Point-To-Point Metric: 10
    Interface ID: 0.0.21.207
    Neighbor Interface ID: 0.0.21.206
    Neighbor Router ID: 10.1.3.3
    Type: Point-To-Point Metric: 10
    Interface ID: 0.0.21.210
    Neighbor Interface ID: 0.0.21.211
    Neighbor Router ID: 10.1.4.4
```

So how does the router learn IPv6 addresses connected to this router?
We know that this information is available in Intra-Area-Prefix LSAs (Type-9 LSAs). And you can check this with the command below.

```
u0U-bb1# show ipv6 ospf6 database intra-prefix adv-router 10.U.1.1 detail

        Area Scoped Link State Database (Area 0)

Age: 1523 Type: Intra-Prefix
Link State ID: 0.0.0.0
Advertising Router: 10.1.1.1
LS Sequence Number: 0x80000008
CheckSum: 0x42ee Length: 100
Duration: 00:25:23
     Number of Prefix: 5
     Reference: Router Id: 0.0.0.0 Adv: 10.1.1.1
     Prefix Options: --|--|--|--|--
     Prefix: 2001:db8:100::1/128
     Metric: 10
     Prefix Options: --|--|--|--|--
     Prefix: 2001:db8:100:12::/64
     Metric: 10
     Prefix Options: --|--|--|--|--
     Prefix: 2001:db8:100:13::/64
     Metric: 10
     Prefix Options: --|--|--|--|--
     Prefix: 2001:db8:100:14::/64
     Metric: 10
     Prefix Options: --|--|--|--|--
     Prefix: 2001:db8:103::/64
     Metric: 10
```

As you can see we have the IPv6 information next to the “Prefix” attributes of the LSA Type9s.

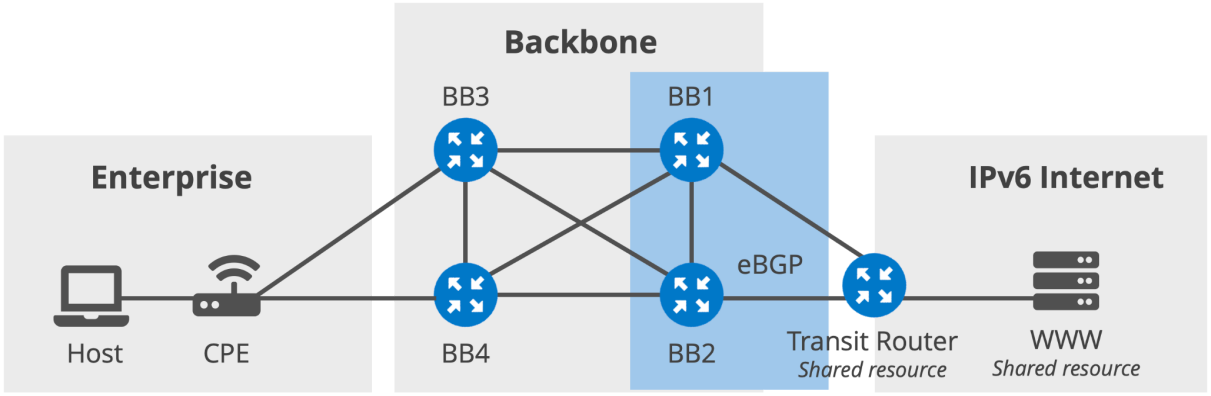
Activity 2 Summary

In this activity:

- You checked the existing lab setup for OSPFv2 in the backbone network
- You configured OSPFv3 between the backbone routers
- You checked the routing tables for IPv6 to see the populated OSPF routes
- You verified the connectivity with the ping tool
- And you compared the OSPFv2 and OSPFv3 databases to see the difference between the ways how IP information is distributed.

Lab Activity 3: Configuring BGP in the backbone network

Activity 3.1: Configuring eBGP with the transit router



In this part of the lab activities, you'll configure eBGP over IPv6 connections with the transit router which connects your network to the internet. You'll do this on the BB1 router (BB2's eBGP configuration is pre-loaded already) which is directly connected to the transit network. The transit router is already configured and you don't need to do anything for that.

You can check the existing IPv4 **eBGP** neighbors of BB1 to better understand our lab setup with the command below:

```
u0U-bb1# show bgp summary remote-as external

IPv4 Unicast Summary (VRF default):
BGP router identifier 10.1.1.1, local AS number 100 vrf-id 0
BGP table version 2
RIB entries 2, using 384 bytes of memory
Peers 4, using 2868 KiB of memory

Neighbor        V         AS   MsgRcvd   MsgSent   TblVer  InQ OutQ  Up/Down State/PfxRcd   PfxSnt Desc
10.1.81.8       4         99        63        63        0    0    0 00:56:57            1        1 N/A

Displayed neighbors 1
Total number of neighbors 4
```

This output is taken from BB1.
As you can see in the output, we only have one external IPv4 neighbor which is the transit router.

To begin, you'll need to modify the default BGP behavior of the FRR router. By default, it treats every configured BGP neighbor, including those with IPv6 addresses, as a neighbor for exchanging **IPv4 unicast routes**. We need to disable this on **all backbone routers** with the command below.

U is your user number assigned by the trainer.

```
configure terminal
router bgp U00
 no bgp default ipv4-unicast
end
```

Now we can start to configure neighbors for **BB1** under BGP with the following command: **U** is your user number assigned by the trainer and **R** is the router number you are working on (R=1 for router BB1).

```
configure terminal
router bgp U00
 neighbor 2001:db8:U00:R8::8 remote-as 99
```

After this configuration you'll enable the IPv6 unicast address family under the BGP configuration part on routers BB1 with the following commands:

```
 address-family ipv6 unicast
```

Now you can activate neighborship with the transit router.

```
 neighbor 2001:db8:U00:R8::8 activate
end
```

At this point, you can check the status of neighbors with the command:

```
show bgp ipv6 summary
```

```
u0U-bb1# show bgp ipv6 summary

IPv6 Unicast Summary (VRF default):
BGP router identifier 10.1.1.1, local AS number 100 vrf-id 0
BGP table version 0
RIB entries 0, using 0 bytes of memory
Peers 1, using 717 KiB of memory

Neighbor          V         AS   MsgRcvd   MsgSent   TblVer  InQ OutQ  Up/Down State/PfxRcd   PfxSnt Desc
2001:db8:100:18::8 4         99         4         3        0    0    0 00:00:41     (Policy) (Policy) N/A

Total number of neighbors 1
```

When you look at this specific output you can understand this neighborship is up for only 41 seconds. In addition to this you can understand that there is no prefix sent or received because of the **default policy**. As you are connected to the transit network to get access to the Internet, you need to allow some routes into your network. You'll do it with the route-maps.
Firstly, we'll configure a IPv6 prefix filter to allow IPv6 routes with a prefix length **less than or equal to 48**. /48 is the minimum prefix size that will be routed globally in the BGP. (There is no hard coded limit for this but this is the best-practice followed by most of the operators around the world)

```
configure terminal
ipv6 prefix-list ANY_v6 seq 10 permit 2000::/3 le 48
```

Then, you'll use this prefix list in your new route-map configuration.

```
route-map ANY_IN_v6 permit 100
 match ipv6 address prefix-list ANY_v6
 exit
```

And finally, you'll apply this newly configured route-map to the transit neighbor for the IN direction to accept the routes.

```
router bgp U00
 address-family ipv6 unicast
  neighbor 2001:db8:U00:R8::8 route-map ANY_IN_v6 in
  end
```

Now you should be able to see a **default route** in the BGP table. You can check it with the command:

```
u0U-bb1# show bgp
BGP table version is 1, local router ID is 10.1.1.1, vrf id 0
Default local pref 100, local AS 100
Status codes:  s suppressed, d damped, h history, * valid, > best, = multipath,
               i internal, r RIB-failure, S Stale, R Removed
Nexthop codes: @NNN nexthop's vrf id, < announce-nh-self
Origin codes:  i - IGP, e - EGP, ? - incomplete
RPKI validation codes: V valid, I invalid, N Not found

   Network          Next Hop            Metric LocPrf Weight Path
*> 2000::/3         fe80::a8c1:abff:fe55:c643
                                             0             0 99 ?

Displayed  1 routes and 1 total paths
```

With this configuration, you'll receive the routes from the transit network. However, **you still need to advertise your own network address** to the transit and the internet eventually. To do this you'll create another route-map to be applied in the **OUT** direction. Here we'll do this for two different prefixes. First for the prefixes belong to 2001:db8::/32 and second prefixes belong to 3fff:0::/20.
Note: *These are both documentation prefixes and we'll use 3fff:0::/20 range later in the "Multihoming" exercise.*

Without the exact match, this prefix will not be announced in the BGP so we need to create a Null0 route first.

```
configure terminal
ipv6 route 2001:0db8:U00::/40 Null0
ipv6 route 3fff:0:U00::/40 Null0
```

Then we'll create the prefix list. In our address plan in the lab network we have **/40 allocations for each user** so we'll announce these /40s to the transit router.

```
ipv6 prefix-list USER_U_NETW_v6 seq 10 permit 2001:db8:U00::/40
ipv6 prefix-list USER_U_NETW_v6 seq 20 permit 3fff:0:U00::/40
```

Then, you'll use this new prefix list in your new route-map configuration.

```
route-map USER_U_NETW_OUT_v6 permit 100
 match ipv6 address prefix-list USER_U_NETW_v6
 exit
```

Finally, you'll apply this newly configured route-map to the transit neighbor in the OUT direction and you'll redistribute the static route you configured for your user network (/40 for the user) into the BGP.

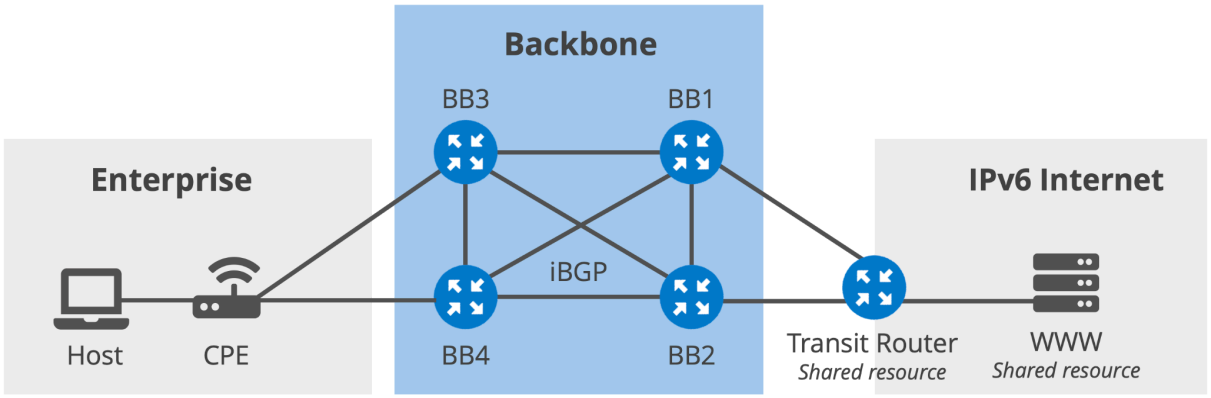
```
router bgp U00
 address-family ipv6 unicast
  redistribute static
  neighbor 2001:db8:U00:R8::8 route-map USER_U_NETW_OUT_v6 out
  end
```

Now you can check the status of IPv6 neighbors again:

```
u0U-bb1# show bgp ipv6 summary

IPv6 Unicast Summary (VRF default):
BGP router identifier 10.1.1.1, local AS number 100 vrf-id 0
BGP table version 2
RIB entries 2, using 384 bytes of memory
Peers 1, using 717 KiB of memory

Neighbor           V         AS   MsgRcvd   MsgSent   TblVer  InQ OutQ  Up/Down State/PfxRcd   PfxSnt Desc
2001:db8:100:18::8 4         99        23        20        0    0    0 00:07:53            1        2 N/A

Total number of neighbors 1
```

As you can see now you have sent and received prefixes. You send your network prefixes (including both backbone and customer networks).

Activity 3.2: Configuring iBGP within the backbone



Now you'll configure BGP within your backbone network which will form iBGP neighborship between the backbone routers.

You can start to configure neighbors on BB1 and BB3, (BB2 and BB4 are already pre-configured) under BGP with the following command (**X** is the number of one of the **other routers** in the backbone, for example if you're configuring BB3, **X** will be 1, 2 and 4 respectively):

```
configure terminal
router bgp U00
 neighbor 2001:db8:U00::X remote-as U00
 neighbor 2001:db8:U00::X update-source lo
```

Here we use the update-source keyword to be able to use the loopback addresses for the protocol messages.
Now we can activate neighborship under the IPv6 address family.

```
 address-family ipv6 unicast
  neighbor 2001:db8:U00::X activate
  neighbor 2001:db8:U00::X next-hop-self
end
```

Now you can check the status of the neighbors again (on router BB1 and BB2 you will see 4 neighbors and on routers BB3 and BB4 you will see 3 neighbors):

```
u0U-bb1# show bgp ipv6 summary

IPv6 Unicast Summary (VRF default):
BGP router identifier 10.1.1.1, local AS number 100 vrf-id 0
BGP table version 2
RIB entries 2, using 384 bytes of memory
Peers 4, using 2868 KiB of memory

Neighbor           V         AS   MsgRcvd   MsgSent   TblVer  InQ OutQ  Up/Down State/PfxRcd   PfxSnt Desc
2001:db8:100::2    4        100         5         6        0    0    0 00:00:56            2        2 N/A
2001:db8:100::3    4        100         3         6        0    0    0 00:00:39            0        2 N/A
2001:db8:100::4    4        100         3         5        0    0    0 00:00:25            0        2 N/A
2001:db8:100:18::8 4         99        28        25        0    0    0 00:12:36            1        1 N/A

Total number of neighbors 4
```

At this step, you should be able to ping the WEB server behind our transit router from BB3 (or BB4).

```
u0U-bb3# ping 2001:db8:99:78::7
PING 2001:db8:99:78::7 (2001:db8:99:78::7): 56 data bytes
64 bytes from 2001:db8:99:78::7: seq=0 ttl=62 time=0.196 ms
64 bytes from 2001:db8:99:78::7: seq=1 ttl=62 time=0.129 ms
64 bytes from 2001:db8:99:78::7: seq=2 ttl=62 time=0.125 ms
64 bytes from 2001:db8:99:78::7: seq=3 ttl=62 time=0.153 ms
64 bytes from 2001:db8:99:78::7: seq=4 ttl=62 time=0.157 ms
^C
--- 2001:db8:99:78::7 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.125/0.152/0.196 ms
```

To identify the individual network layer protocols (such as IPv6), a combination of Address Family Identifier (AFI) and Subsequent Address Family Identifier (SAFI) is used. While exchanging capabilities, the same address family identifiers are also advertised within the OPEN messages. And you can see the address families of the specific routes with the following commands **on router BB4**.

```
u0U-bb4# show  ip route 10.U.0.0/16 json | include afi
          "afi":"ipv4",
          "afi":"ipv4",
          "afi":"ipv4",
          "afi":"ipv4",
u0U-bb4#
u0U-bb4# show ipv6 route 2001:db8:U00::/40 json | include afi
          "afi":"ipv6",
          "afi":"ipv6",
          "afi":"ipv6",
          "afi":"ipv6",
```

Note: We see them four times because they are coming from two different routers (BB1 and BB2), once for recursive next-hops and once for resolved next-hops.

Activity 3 Summary

In this activity:

- You checked the existing eBGP connections between the transit router and the 2 of the backbone routers; BB1 and BB2.
- You configured eBGP over IPv6 between the transit router and the 1 of the backbone routers; BB1
- You configured the required policies to receive and send the routes
- You configured iBGP over IPv6 within the backbone network.
- You compared the address family attributes (AFIs) of IPv4 and IPv6 routes

Lab Activity 4: Multihoming for SOHO Networks (with and without own address space)

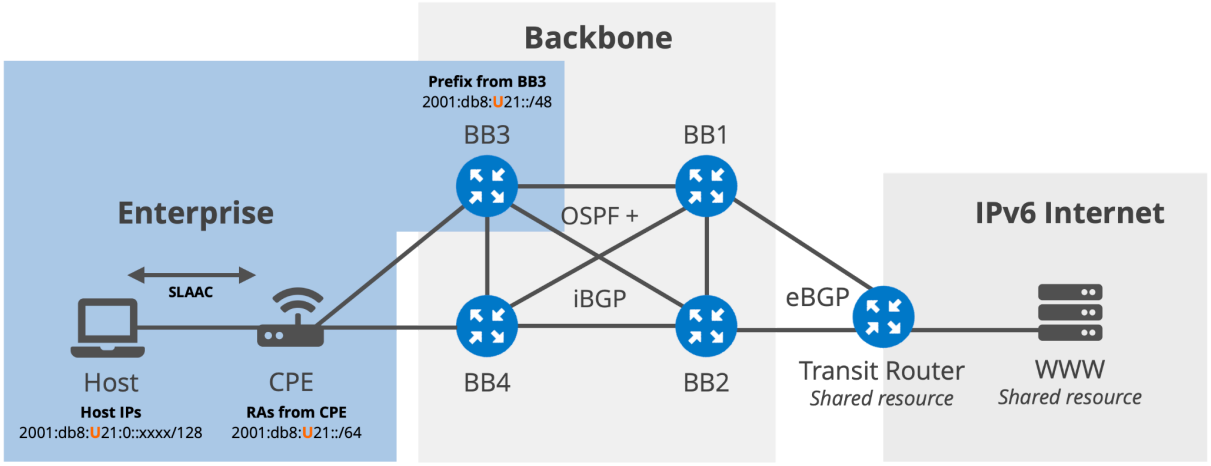
The multihoming problem in IPv6 SOHO networks arises when a small network connects to multiple ISPs, each assigning its own IPv6 address prefix. The CPE router advertises these prefixes on its LAN interface, causing IPv6 hosts to generate addresses from both. This results in hosts having multiple global IPv6 addresses with different prefixes. When initiating new sessions, hosts select source addresses based on destination matching rather than network topology, potentially leading to arbitrary address usage for upstream traffic. This can cause asymmetrical routing or packet dropping due to ISPs' Reverse Path Forwarding checks. Link failures further complicate the issue, as revoking invalid prefixes can take up to two hours due to IPv6 prefix lifetime constraints. During this period, clients may use invalid addresses without knowing it, resulting in dropped return traffic.

In this exercise, you'll first simulate this multihoming problem for SOHOs which do not have their own IPv6 address allocation and use ISPs addresses and then you'll solve this problem with one of the most viable methods. Your setup will look like what we've seen in the course slides previously.

Important Note: Generally, assigning prefixes is not a manual process and this is achieved by the prefix delegation (PD), however our router software, FRR, does not support this and we simulate it by adding prefixes to be passed to the host manually. The idea is still the same.

You'll start with a main uplink scenario (the link to the BB3 router) and you'll add a backup link (the link to the BB4 router) to simulate the multihoming for the CPE device. We'll assign two different GUA IPv6 addresses, from two different ISPs, to the host behind with the RA messages from the CPE.

Activity 4.1: Configuring a single homed SOHO network without own address space



You'll start by checking the interfaces and the IP addresses on the CPE device to see the interfaces already configured. As you'll see, the CPE device is physically connected to the BB3 and the BB4 routers and also to the host behind. The IPv6 addresses are already assigned to the interfaces.

```
u0U-cpe# show interface brief
Interface       Status  VRF             Addresses
---------       ------  ---             ---------
eth0            down    default         172.20.20.9/24
lo              up      default
to_bb3          up      default         + 2001:db8:U02:36::6/64
to_bb4          up      default         + 2001:db8:U02:46::6/64
to_host2        up      default         + 2001:db8:U21::1/64
```

You'll also see that we have a default route configured on the CPE router and it is pointing to the BB3 router.

```
u0U-cpe# show ipv6 route
Codes: K - kernel route, C - connected, S - static, R - RIPng,
       O - OSPFv3, I - IS-IS, B - BGP, N - NHRP, T - Table,
       v - VNC, V - VNC-Direct, A - Babel, F - PBR,
       f - OpenFabric,
       > - selected route, * - FIB route, q - queued, r -
rejected, b - backup
       t - trapped, o - offload failure

S>* ::/0 [1/0] via 2001:db8:102:36::3, to_bb3, weight 1, 3d16h33m
C>* 2001:db8:102:36::/64 is directly connected, to_bb3, 3d16h33m
C>* 2001:db8:102:46::/64 is directly connected, to_bb4, 3d16h33m
C>* 2001:db8:121::/64 is directly connected, to_host2, 00:00:43
```

```
C * fe80::/64 is directly connected, to_host2, 00:00:44
C * fe80::/64 is directly connected, eth0, 3d16h33m
C>* fe80::/64 is directly connected, to_bb4, 3d16h33m
```

Now you are ready to add more configurations to devices to make the host reachable over the Internet.

You'll need to add a prefix (we'll call it enterprise network prefix) to the interface facing the host together with the command that enables RA messages to be sent. This prefix will be advertised to the hosts behind the CPE in the RA messages.

On the CPE:

```
configure terminal
interface to_host2
 ipv6 nd prefix 2001:db8:U21::/64
 no ipv6 nd suppress-ra
end
```

The host is configured to get its IPv6 address automatically using the prefix in the RA messages by default. Now, it is time to check the IPv6 address of the host. Later you'll use this address in the ping tests. You'll use the "`ip addr show`" command to check the interface IP addresses on the Linux host.

```
bash-5.0# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
1520: to_cpe@if1519: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9500 qdisc noqueue state UP group
default
    link/ether aa:c1:ab:9c:20:3b brd ff:ff:ff:ff:ff:ff link-netnsid 1
    inet6 2001:db8:121:0:a8c1:abff:fe9c:203b/64 scope global dynamic mngtmpaddr
       valid_lft 2591995sec preferred_lft 604795sec
    inet6 fe80::a8c1:abff:fe9c:203b/64 scope link
       valid_lft forever preferred_lft forever
```

There is still one more thing to do to make this host reachable over the Internet (or in our case from the transit network and the hosts there, ie. WWW server). You need to configure a route **on BB3** to let the router know where the enterprise network prefix lies and announce this in BGP.

To **add this static route and redistribute it** over the BGP, **on the router BB3** you need to run the following commands:

```
configure terminal
ipv6 route 2001:db8:U21::/48 2001:db8:U02:36::6 to_cpe
router bgp U00
 address-family ipv6 unicast
```

```
  redistribute static
end
```

By now you should be able to ping the WWW server (`2001:db8:99:78::7`) behind our transit router from the host. To make sure that the host uses the correct source address we'll do the ping with the source address parameter as follows;

```
ping6 -I 2001:db8:U21:0:xxxx:xxxx:xxxx:xxxx 2001:db8:99:78::7
```

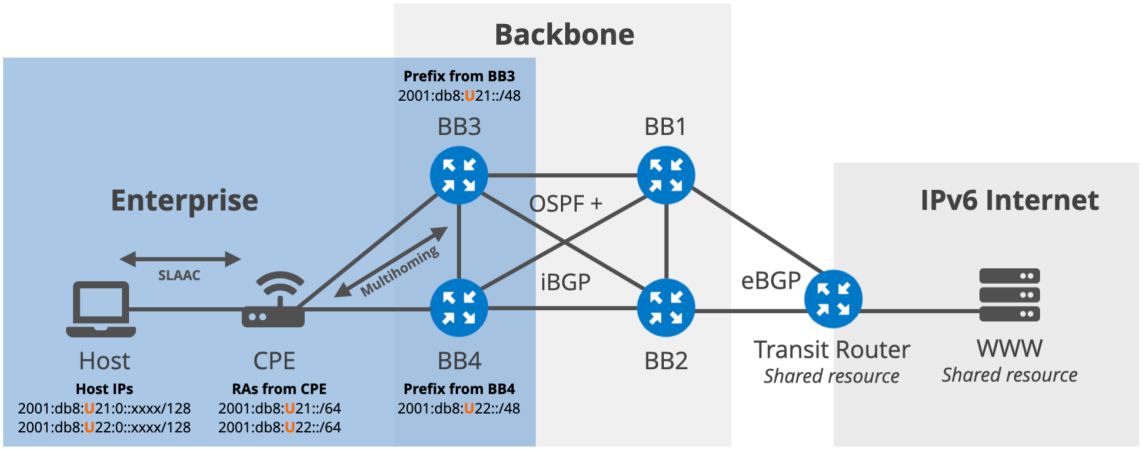
Please note that "**xxxx:xxxx:xxxx:xxxx**" will be a randomly generated value for the interface ID in your lab environment

The output should look like:

```
bash-5.0# ping6 -I 2001:db8:U21:0:xxxx:xxxx:xxxx:xxxx 2001:db8:99:78::7
PING 2001:db8:99:78::7(2001:db8:99:78::7) from
2001:db8:121:0:a8c1:abff:fe9c:203b : 56 data bytes
64 bytes from 2001:db8:99:78::7: icmp_seq=1 ttl=61 time=0.053 ms
64 bytes from 2001:db8:99:78::7: icmp_seq=2 ttl=61 time=0.066 ms
64 bytes from 2001:db8:99:78::7: icmp_seq=3 ttl=61 time=0.063 ms
64 bytes from 2001:db8:99:78::7: icmp_seq=4 ttl=61 time=0.076 ms
^C
--- 2001:db8:99:78::8 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
rtt min/avg/max/mdev = 0.053/0.064/0.076/0.008 ms
```

Now you verified that the WWW server is reachable from the host behind the CPE with a single uplink to the ISPs.

Activity 4.2: Configuring the redundant uplink



After successfully configuring the main uplink of the CPE you'll add a redundant link into our SOHO scenario.
To start, you'll check the IPv6 addresses on the link between the router BB4 and the CPE, and the link between the CPE and the host.
First start with the **BB4 interface**:

```
u0U-bb4# show interface brief
Interface       Status  VRF             Addresses
---------       ------  ---             ---------
eth0            down    default         172.20.20.7/24
lo              up      default         10.1.4.4/32
                                        2001:db8:100::4/128
to_bb1          up      default         10.1.41.4/24
                                        + 2001:db8:100:14::4/64
to_bb2          up      default         10.1.42.4/24
                                        + 2001:db8:100:24::4/64
to_bb3          up      default         10.1.43.4/24
                                        + 2001:db8:100:34::4/64
to_cpe          up      default         + 2001:db8:102:46::4/64
```

And **on the CPE** you'll run the same command to check the interface IPv6 addresses.

```
u0U-cpe# show interface brief
Interface       Status  VRF             Addresses
---------       ------  ---             ---------
eth0            down    default         172.20.20.8/24
lo              up      default
to_bb3          up      default         + 2001:db8:102:36::6/64
to_bb4          up      default         + 2001:db8:102:46::6/64
to_host2        up      default         + 2001:db8:121::1/64
```

Then, on the CPE you'll add another default route pointing to the BB4 interface.
Note: In real life deployments, depending on the software and the configuration on the CPE this might be enabling equal-cost multipath feature (ECMP) or the CPE might prioritize any of the default routes you configured.

```
configure terminal
ipv6 route ::/0 2001:db8:U02:46::4 to_bb4
end
```

Now, we assume that another prefix delegation from the second ISP (represented by the BB4 router) happens and the CPE device needs to pass this prefix to the host behind it. You'll configure another IPv6 address on the interface facing the host by using this prefix. (As we discussed before, in real-life scenarios, this happens automatically after the prefix is delegated from the ISP). After configuring the CPE's interface with the new IPv6 address, you'll add this new prefix into the RA messages sent by the CPE.

On the CPE device:

```
configure terminal
interface to_host2
 ipv6 address 2001:db8:U22::1/64
 ipv6 nd prefix 2001:db8:U22::/64
end
```

Now, it is time again to check the IPv6 addresses configured on the host to see the effect of the change you made in the CPE configuration.

```
bash-5.0# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
1520: to_cpe@if1519: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9500 qdisc noqueue state
UP group default
    link/ether aa:c1:ab:9c:20:3b brd ff:ff:ff:ff:ff:ff link-netnsid 1
    inet6 2001:db8:122:0:a8c1:abff:fe9c:203b/64 scope global dynamic mngtmpaddr
       valid_lft 2591992sec preferred_lft 604792sec
    inet6 2001:db8:121:0:a8c1:abff:fe9c:203b/64 scope global dynamic mngtmpaddr
       valid_lft 2591992sec preferred_lft 604792sec
    inet6 fe80::a8c1:abff:fe9c:203b/64 scope link
       valid_lft forever preferred_lft forever
```

After validating that both IPv6 addresses are present on the host, you need to add and redistribute another static route into the BB4 router's configuration to be able to reach this new prefix from BB4 and from the rest of the network.

On router BB4:

```
configure terminal
ipv6 route 2001:db8:U22::/48 2001:db8:U02:46::6 to_cpe
router bgp U00
 address-family ipv6 unicast
  redistribute static
end
```

With this, you should be able to ping the WWW server (2001:db8:99:78::7) behind our transit router from the host **with the new source address** by specifying it in the ping command as shown below.

```
ping6 -I 2001:db8:U22:0:xxxx:xxxx:xxxx:xxxx 2001:db8:99:78::7
```

The output should look like:

```
bash-5.0# ping6 -I 2001:db8:U22:0:xxxx:xxxx:xxxx:xxxx 2001:db8:99:78::7
PING 2001:db8:99:78::7(2001:db8:99:78::7) from 2001:db8:122:0:a8c1:abff:fe9c:203b :
56 data bytes
64 bytes from 2001:db8:99:78::7: icmp_seq=1 ttl=61 time=0.150 ms
```

```
64 bytes from 2001:db8:99:78::7: icmp_seq=2 ttl=61 time=0.073 ms
64 bytes from 2001:db8:99:78::7: icmp_seq=3 ttl=61 time=0.072 ms
64 bytes from 2001:db8:99:78::7: icmp_seq=4 ttl=61 time=0.073 ms
^C
--- 2001:db8:99:78::8 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3000ms
rtt min/avg/max/mdev = 0.072/0.092/0.150/0.033 ms
```

This output indicates that the WWW server is reachable with the new source address as well. At this point, we are simulating a scenario where you (as the enterprise) are connected to two different ISPs and have received two different IPv6 address delegations from them. Your host accepts both prefixes and configures an IPv6 address for each. Both addresses are valid and usable. It is up to the host and the applications on the host to decide which one to use. The CPE device uses both ISPs as uplinks, each with a default route.

Activity 4.3: Shutting down the main uplink and simulating the problem

In this lab setup, we are not applying “reverse path forwarding” feature (RPF/uRPF) on the routers, but in real scenarios, ISPs are very likely to apply RPF to be able to prevent spoofing, which means that even in this scenario, there is a high possibility of encountering problems. Let's consider that your host is using the IP provided by the second ISP as the source, and the CPE tries to forward this packet through the first ISP. As the first ISP does not expect this source from the CPE (thanks to RPF), it will perceive this packet as a spoof and will drop it. Or, in other cases, where RPF is not in use, there might be asymmetrical routing which means sending and receiving packets of the same session through different uplinks.

In our lab scenario, we will make the issue more dramatically visible. First, you’ll shut down the link between the CPE and the BB3 router simulating a failure on this uplink. Then, you’ll send a ping packet from the host to the WWW server again using the IPv6 address configured from the prefix received from the first ISP.

Now, **on the BB3 router, shut down the interface to the CPE**.

```
configure terminal
interface to_cpe
 shut
 end
```

You can check the addresses on the host and you’ll see that both of them are still valid.

```
bash-5.0# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
```

```
       valid_lft forever preferred_lft forever
1520: to_cpe@if1519: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9500 qdisc noqueue state
UP group default
    link/ether aa:c1:ab:9c:20:3b brd ff:ff:ff:ff:ff:ff link-netnsid 1
    inet6 2001:db8:U22:0:xxxx:xxxx:xxxx:xxxx/64 scope global dynamic mngtmpaddr
       valid_lft 2591993sec preferred_lft 604793sec
    inet6 2001:db8:U21:0:xxxx:xxxx:xxxx:xxxx/64 scope global dynamic mngtmpaddr
       valid_lft 2591993sec preferred_lft 604793sec
    inet6 fe80::a8c1:abff:fe9c:203b/64 scope link
       valid_lft forever preferred_lft forever
```

Note: This is also true when you use "prefix delegation" in real-life scenarios. If the CPE router loses an uplink, it can detect the link loss and loss of the delegated prefix, and revoke the IPv6 prefix from router advertisements sent to the LAN interfaces. However, this process might be slow — the minimum valid lifetime of an IPv6 prefix in ND messages used for stateless autoconfiguration is two hours, so in this case it could take up to two hours for clients to recognise that their IPv6 address from the failed link is no longer valid. During this window, the client continues to use an address that has become invalid without realising it. Although outbound traffic would be forwarded over the remaining link, the return traffic might end up in the wrong Autonomous System (AS) associated with the failed link to your site and be dropped.

And from the host try to ping the WWW server (2001:db8:99:78::7) again with the new source address by specifying it in the ping command as shown below.

```
ping6 -I 2001:db8:U21:0:xxxx:xxxx:xxxx:xxxx 2001:db8:99:78::7
```

What do you see? Is it still reachable?

The output should look like this (you can use “Ctrl+C” or “Command+C” key combinations to stop the command trying)

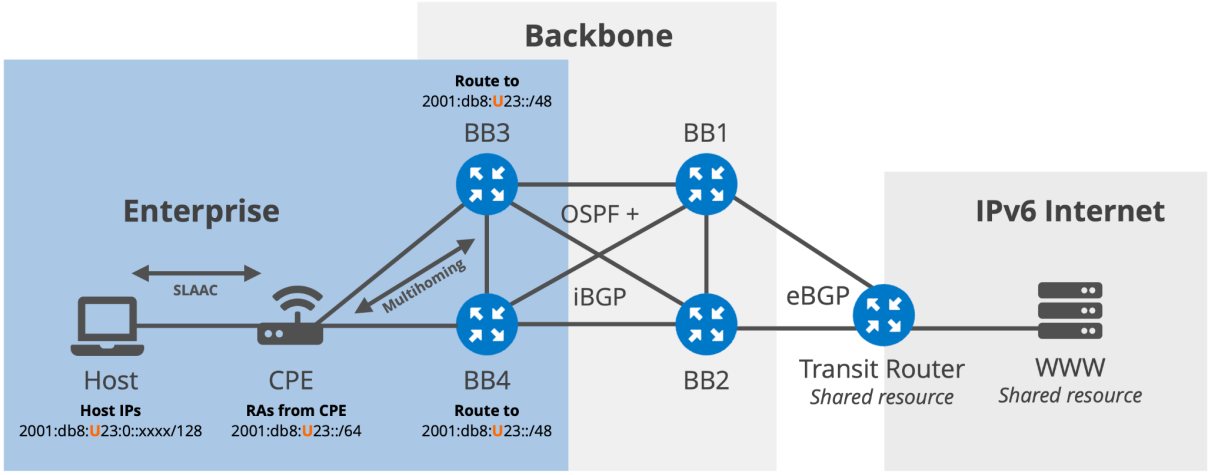
```
bash-5.0# ping6 -I 2001:db8:U21:0:xxxx:xxxx:xxxx:xxxx 2001:db8:99:78::7
PING 2001:db8:99:78::7(2001:db8:99:78::7) from
2001:db8:121:100:a8c1:abff:fe17:eb42 : 56 data bytes
^C
--- 2001:db8:99:78::7 ping statistics ---
13 packets transmitted, 0 received, 100% packet loss, time 11999ms
```

You can clearly see that it is not reachable by looking at the packet loss percentage in the output.
By doing this you simulated the problem in the multihomed networks of SOHOs which use ISP assigned addresses.

Activity 4.4: Using your own address space

We have observed the problems caused by this setup. Now, let’s concentrate on one of the most viable solutions. If you have your own address space, it may simplify the issue for you. You can configure your hosts with your own single address prefix, and you can request your ISPs to announce this prefix for you to the outside world. You’ll simulate this approach in the

lab topology now, by removing the ISP assigned IPv6 prefixes in the previous steps and adding a new prefix which is allocated to this enterprise company. After using this new prefix on the CPE device you'll also announce this to the outside world by redistributing them in the BGP on the ISP owned backbone devices (namely BB3 and BB4) In this scenario, your network topology will look like the one shown below.



First re-enable the interface facing the CPE **on the router BB3**.

```
configure terminal
interface to_cpe
 no shut
end
```

You'll remove the addresses and the prefixes assigned by the ISPs from the interface facing the host.

On the CPE device:

```
configure terminal
interface to_host2
 no ipv6 address 2001:db8:U21::1/64
 no ipv6 address 2001:db8:U22::1/64
 no ipv6 nd prefix 2001:db8:U21::/64
 no ipv6 nd prefix 2001:db8:U22::/64
```

And configure a new address with a new prefix (this new prefix is reserved for documentation purposes in 2024 in RFC9637).

```
 ipv6 address 3fff:0:U23::1/64
 ipv6 nd prefix 3fff:0:U23::/64
end
```

You need to route this new prefix to the enterprise network on the ISP routers you are connected to.

On BB3:

```
configure terminal
no ipv6 route 2001:db8:U21::/48 2001:db8:U02:36::6 to_cpe
ipv6 route 3fff:0:U23::/48 2001:db8:U02:36::6 to_cpe
end
```

On BB4:

```
configure terminal
no ipv6 route 2001:db8:U22::/48 2001:db8:U02:46::6 to_cpe
ipv6 route 3fff:0:U23::/48 2001:db8:U02:46::6 to_cpe
end
```

Finally, you need to reset the interface configuration on the host and trigger the RA on CPE.

On the host:

```
ip -6 addr flush dev to_cpe scope global
```

Now check the IPv6 addresses on the host interface (it can take up to 20 seconds as the RA interval on the CPE is 20 seconds). The output will look like:

```
bash-5.0# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
1520: to_cpe@if1519: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9500 qdisc
noqueue state UP group default
    link/ether aa:c1:ab:9c:20:3b brd ff:ff:ff:ff:ff:ff link-netnsid 1
    inet6 3fff:0:U23:0:xxxx:xxxx:xxxx:xxxx/64 scope global tentative
dynamic mngtmpaddr
       valid_lft 2592000sec preferred_lft 604800sec
    inet6 fe80::a8c1:abff:fe9c:203b/64 scope link
       valid_lft forever preferred_lft forever
```

From the host start pinging the WWW server (2001:db8:99:78::8) again with the new source address by specifying it in the ping command as shown below and leave it on.

```
ping6 -I 3fff:0:U23:0:xxxx:xxxx:xxxx:xxxx 2001:db8:99:78::8
```

And now you'll try to test the link failure again. You can shut down the link between the CPE and the BB3 router.

On the BB3 router:

```
configure terminal
interface to_cpe
 shut
```

```
end
```

Check the ping output on the host. Do you see any failed packets? Is the server still reachable?

```
u0U-host2 / # ping6 -I 3fff:0:U23:0:xxxx:xxxx:xxxx:xxxx 2001:db8:99:78::8
PING 2001:db8:99:78::8 (2001:db8:99:78::8) from
3fff:0:123:0:xxxx:xxxx:xxxx:xxxx : 56 data bytes
64 bytes from 2001:db8:99:78::8: icmp_seq=1 ttl=61 time=0.108 ms
64 bytes from 2001:db8:99:78::8: icmp_seq=2 ttl=61 time=0.152 ms
64 bytes from 2001:db8:99:78::8: icmp_seq=3 ttl=61 time=0.117 ms
64 bytes from 2001:db8:99:78::8: icmp_seq=4 ttl=61 time=0.083 ms
^C
--- 2001:db8:99:78::8 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3000ms
rtt min/avg/max/mdev = 0.083/0.115/0.152/0.024 ms
```

As you'll see there is no packet loss and the WWW server is still reachable. You can also try this with the second uplink (don't forget to enable the link on BB3 first before you shutdown the BB4 link to the CPE device)

Note: If you were trying to test this on a live network, you'd probably have some active equipment between the CPE and the ISP routers (BB3 and BB4 in our case). Therefore, a failure on the link would be harder to detect, as the active equipment would keep the link **up**. In such cases, you can employ the bi-directional forward detection (BFD)***** feature and associate it with the static route you configure. This would allow the router to detect a failure over the link, even if there is active L2 (switch) or L1 (DWDM) equipment between the CPE and the routers.
****Bi-directional forward detection (BFD):*** *Bidirectional Forwarding Detection (BFD) is a network protocol designed to quickly detect faults in the communication path between two routers or switches, often within milliseconds. It works on various types of connections, including Ethernet, virtual circuits, tunnels, and MPLS paths, even if the underlying media doesn't inherently support failure detection. BFD establishes a session between two endpoints and can detect faults in as little as a few milliseconds, enabling rapid response to network issues.*

Activity 4 Summary

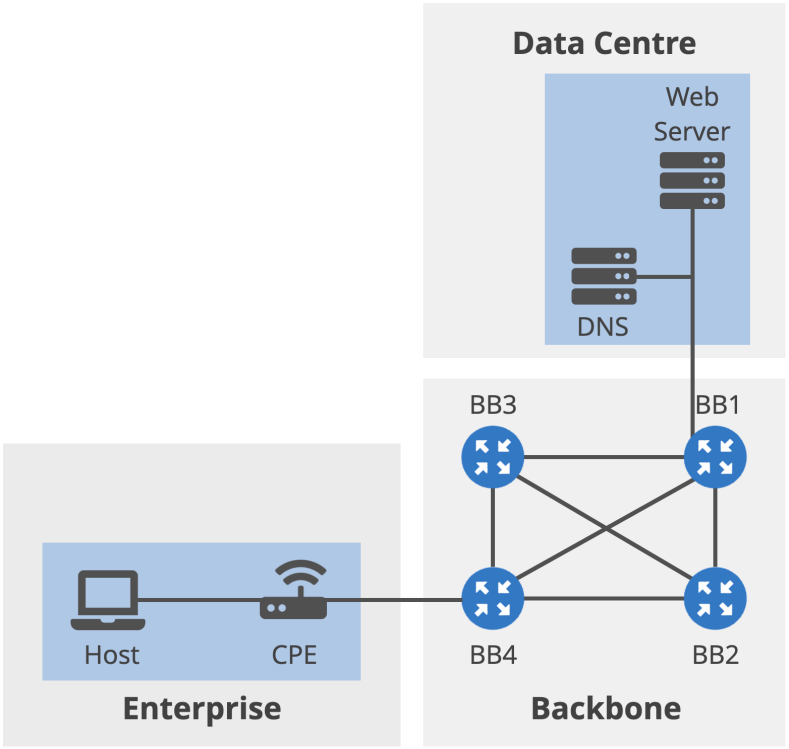
In this activity:

- You configured a single homed SOHO network with a single host behind a CPE.
- You configured the redundant uplink for the CPE simulating a connection to another ISP and making CPE multihomed.
- You checked and verified the connectivity from the host over both uplinks of CPE with two different IPv6 prefixes assumed to be assigned by the ISPs.
- You triggered and verified the problem by shutting down one of the uplinks

- You configured the same physical setup and solved the problem by using a single IPv6 prefix allocated to SOHO company instead of ISPs.

Lab Activity 5: DNS Configuration

In this lab activity, you will configure the DNS server on Host2 by sending it in the RA message from the CPE. You'll check the DNS zone file on the DNS server and then you'll configure a AAAA record in the zone file. You'll test web server reachability with the curl command on the linux host to verify that DNS server works as expected.



Activity 5.1: Configuring RDNSS option on CPE

First, you need to verify that there is no DNS server configured on the host device. To do this you'll use the following command on "host2".

```
u0U-host2 / # more etc/resolv.conf
# Generated by dhcpcd
# /etc/resolv.conf.head can replace this line
# /etc/resolv.conf.tail can replace this line
```

As you can see, in the output, youdon't have any dns or "nameserver" information.

Now, let's start to configure the RDNSS option on the CPE. You'll do it under the interface facing the host. Your DNS server IP address is:
`2001:db8:U03::e`

On the CPE router:

```
configure terminal
interface to_host2
 ipv6 nd rdnss 2001:db8:U03::e
end
```

Now you can check the DNS server information on the host again. If everything is correct you should see an output similar to the one below.

```
u0U-host2 / # more etc/resolv.conf
# Generated by dhcpcd from to_cpe.ra
# /etc/resolv.conf.head can replace this line
nameserver 2001:db8:U03::e
# /etc/resolv.conf.tail can replace this line
```

Activity 5.2: Configuring DNS Server for AAAA Record

In this activity, first you need to check WEB server reachability.

On Host2 check the assigned GUA address again and try pinging the WEB server with that GUA IPv6 address.

```
u0U-host2 / # ping -I 3fff:0:U23:0:XXXX:XXXX:XXXX:XXXX 2001:db8:U03::b
PING 2001:db8:U03::b (2001:db8:203::b) from 3fff:0:U23:0:xxxx:xxxx:xxxx:xxxx
: 56 data bytes
64 bytes from 2001:db8:U03::b: icmp_seq=1 ttl=61 time=0.928 ms
64 bytes from 2001:db8:U03::b: icmp_seq=2 ttl=61 time=0.147 ms
64 bytes from 2001:db8:U03::b: icmp_seq=3 ttl=61 time=0.147 ms
64 bytes from 2001:db8:U03::b: icmp_seq=4 ttl=61 time=0.144 ms
64 bytes from 2001:db8:U03::b: icmp_seq=5 ttl=61 time=0.148 ms
^C
--- 2001:db8:203::b ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4002ms
rtt min/avg/max/mdev = 0.144/0.302/0.928/0.312 ms
```

As you'll see, the WEB server is reachable with its IPv6 address. Now try to see if there is any DNS record for "localweb.example". This is a domain that should be pointing to the same WEB server.

You can use **host** and **dig** commands for this. Try both as shown below.

First try "**host**".

```
u0U-host2 / # host localweb.example
Host localweb.example not found: 3(NXDOMAIN)
```

Now with “**dig”**.

```
u0U-host2 / # dig AAAA localweb.example

; <<>> DiG 9.18.27 <<>> AAAA localweb.example
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 46754
;; flags: qr aa rd; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: fecbaffe1359858301000000671f6cd8fec5bba6f82c7cbc (good)
;; QUESTION SECTION:
;localweb.example.              IN      AAAA

;; AUTHORITY SECTION:
example.                300     IN      SOA     ns.example. admin.example.
2024090100 7200 3600 1209600 3600

;; Query time: 1 msec
;; SERVER: 2001:db8:U03::e#53(2001:db8:203::e) (UDP)
;; WHEN: Mon Oct 28 10:52:08 UTC 2024
;; MSG SIZE  rcvd: 118
```

As you can see, there are no AAAA records for this domain. Let’s create it on the DNS server!

On the DNS server, first check the zone file for the domain “example”. You can use “**more”** command.

```
u0U-dns / # more etc/bind/example.zone
$ORIGIN example.
$TTL 300

@       300     IN      SOA ns.example. admin.example. (
                                2024090100 ; serial
                                7200       ; refresh (2 hours)
                                3600       ; retry (1 hour)
                                1209600    ; expire (2 weeks)
                                3600       ; minimum (1 hour)
                                )

@       300     IN      NS      ns.example.
ns      300     IN      A       127.0.0.1

www     300     IN      A       10.0.87.7
```

Here we verify again that there is no AAAA record for this domain.

To be able to create it, you'll use nano text editor on this linux server.

1. Run the command "**nano /etc/bind/example.zone**"
2. Hit the key "**G**" on your keyboard. This should bring your cursor to the last line of the file.
3. Hit "**yy**" on your keyboard. This will copy the last line of the file.
4. Hit "**p**" on the keyboard. This will paste the copied line below the current line.
5. Now hit "**i**" to enable the insert mode and change the last line (which you created recently) according to the example entry below. The IPv6 address in the entry is the IPv6 address of your web server. You can check it with the command "**ip addr**" on the WEB server.

```
localweb        300     IN      AAAA    2001:db8:U03::b
```

6. Now hit the "**Esc**" character on your keyboard to exit insert mode.
7. Write **:wq!** characters to save and exit the file.
8. You can check the zone file again by the command "more etc/bind/example.zone" to verify your new record.
9. As the last step, you need to reload the dns service on the server for the changes to take effect. Use the command below.

```
u0U-dns / # rndc reload
server reload successful
```

After creating and verifying your new domain record on the DNS server, you can check this on Host2 again.

On Host2, run the commands "dig" and "host" again.

First try "host".

```
u0U-host2 / # host localweb.example
localweb.example has IPv6 address 2001:db8:U03::b
```

Now try "dig".

```
u0U-host2 / # dig AAAA localweb.example

; <<>> DiG 9.18.27 <<>> AAAA localweb.example
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 7124
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
```

```
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 11bfcca8191f31f001000000671f6c6484823fe3d3f81c6d (good)
;; QUESTION SECTION:
;localweb.example.              IN      AAAA

;; ANSWER SECTION:
localweb.example.       300     IN      AAAA    2001:db8:U03::b

;; Query time: 0 msec
;; SERVER: 2001:db8:203::e#53(2001:db8:203::e) (UDP)
;; WHEN: Mon Oct 28 10:50:12 UTC 2024
;; MSG SIZE  rcvd: 101
```

And finally you can try to reach the WEB server over the http protocol by using the “**curl**” command. You should see your IPv6 address as a response.

```
u0U-host2 / # curl localweb.example
Your IP address is: 3fff:0:U23:0:xxxx:xxxx:xxxx:xxxx
```

Activity 5 Summary

In this activity:

- You configured RDNSS option on the CPE device and passed the DNS information to the host via RA messages
- You checked the DNS records for a specific domain
- You configured the AAAA record on the DNS server
- You verified the same DNS information on the host and was able to reach the WEB server with this DNS record available.