Schedule

09:00 - 09:30  Coffee, Tea
11:00 - 11:15  Break
13:00 - 14:00  Lunch
15:30 - 15:45  Break
17:30  End
Introductions

• Name
• Number in the list
• Experience with IPv6, Cisco, OSPF, BGP
• Goals
Overview Day 1

• IPv6 Packets
• IPv6 Routing Basics
• Exercise: Enable IPv6
• OSPFv3
• Exercise: Configuring OSPFv3
• BGP
• Exercise: Configuring BGP
• Content
• Mobile Providers
Overview Day 2

- Host Configuration
- Exercise: SLAAC
- DHCPv6
- Exercise: DHCPv6
- Security
- Exercise: NAT64/DNS64
- IP Address Management
- Tips & Tricks
IPv6 Packets

1 - Section
IPv6 Header Format

• Fixed length
  - Optional headers are daisy-chained

• IPv6 header is twice as long (40 bytes) as IPv4 header without options (20 bytes)
# IPv6 Header

## IPv4 Header

<table>
<thead>
<tr>
<th>Field</th>
<th>IPv4</th>
<th>IPv6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>☀️</td>
<td>☀️</td>
</tr>
<tr>
<td>IHL</td>
<td>☀️</td>
<td>☀️</td>
</tr>
<tr>
<td>Type of Service</td>
<td>☀️</td>
<td>☀️</td>
</tr>
<tr>
<td>Total Length</td>
<td>☀️</td>
<td>☀️</td>
</tr>
<tr>
<td>Identification</td>
<td>☀️</td>
<td>☀️</td>
</tr>
<tr>
<td>Flags</td>
<td>☀️</td>
<td>☀️</td>
</tr>
<tr>
<td>Fragment Offset</td>
<td>☀️</td>
<td>☀️</td>
</tr>
<tr>
<td>Time to Live</td>
<td>☀️</td>
<td>☀️</td>
</tr>
<tr>
<td>Protocol</td>
<td>☀️</td>
<td>☀️</td>
</tr>
<tr>
<td>Header Checksum</td>
<td>☀️</td>
<td>☀️</td>
</tr>
<tr>
<td>Source Address</td>
<td>☀️</td>
<td>☀️</td>
</tr>
<tr>
<td>Destination Address</td>
<td>☀️</td>
<td>☀️</td>
</tr>
<tr>
<td>Options</td>
<td>☀️</td>
<td>☀️</td>
</tr>
<tr>
<td>Padding</td>
<td>☀️</td>
<td>☀️</td>
</tr>
</tbody>
</table>

## IPv6 Header

<table>
<thead>
<tr>
<th>Field</th>
<th>IPv6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>☀️</td>
</tr>
<tr>
<td>Traffic Class</td>
<td>☀️</td>
</tr>
<tr>
<td>Flow Label</td>
<td>☀️</td>
</tr>
<tr>
<td>Payload Length</td>
<td>☀️</td>
</tr>
<tr>
<td>Next Header</td>
<td>☀️</td>
</tr>
<tr>
<td>Hop Limit</td>
<td>☀️</td>
</tr>
<tr>
<td>Source Address</td>
<td>☀️</td>
</tr>
<tr>
<td>Destination Address</td>
<td>☀️</td>
</tr>
</tbody>
</table>

**LEGEND**

- ☀️ Field's name kept from IPv4 to IPv6
- ☀️ Field not kept in IPv6
- ☀️ Field not kept in IPv6
- ☀️ Name and position changed in IPv6
- ☀️ New field in IPv6
IPv6 Header

- Optional fields go into extension headers
- Daisy-chained after the main header

IPv6 Header

Next Header: TCP

TCP Header

Data

IPv6 Header

Next Header: Routing

Routing Header

Next Header: TCP

TCP Header

Data

IPv6 Header

Next Header: Routing

Routing Header

Next Header: Fragment

Fragment Header

Next Header: TCP

TCP Header

Data
Common Headers

• Common values of Next Header Fields:
  - 0  Hop-by-hop option (extension)
  - 6  TCP (payload)
  - 17 UDP (payload)
  - 43 Routing (extension)
  - 44 Fragmentation (extension)
  - 50 Encrypted Security Payload (extension)
  - 58 ICMPv6
Fragmentation

• Routers don’t fragment packets with IPv6
  - More efficient handling of packets in the core
  - Fragmentation is being done by host

• If a packet is too big for next hop:
  - “Packet too big” error message
  - This is an ICMPv6 message
  - Filtering ICMPv6 causes problems
A sender who gets this "message-too-big" ICMPv6 error tries again with a smaller packet

- A hint of size is in the error message
- This is called Path MTU Discovery
Ordering of Headers

• Order is important:

  - Only hop-by-hop header has to be processed by every node
  - Routing header needs to be processed by every router
  - Fragmentation has to be processed before others at the destination
IPv6 has no broadcast

There is an “all nodes” multicast group
- ff02::1

Disadvantages of broadcast:
- It wakes up all nodes
- Only a few devices are involved
- Can create broadcast storms
Neighbor Discovery

- IPv6 has no ARP
- Replacement is called Neighbor Discovery
  - Uses ICMPv6
  - Uses Multicast

- Every ARP request wakes up every node
- Each ND request only wakes up a few nodes
Neighbor Discovery

- ND is used by nodes:
  - For address resolution
  - To find neighboring routers
  - To track address changes
  - To check neighbor reachability
  - To do Duplicate Address Detection

- ND uses 5 different ICMPv6 packet types
Neighbor Discovery Protocol

- **Router Solicitation**
  - Hosts sends an ICMPv6 message to inquire if there is a router on the link
Neighbor Discovery Protocol

• Router Advertisement
  - Routers advertise their presence periodically or in response to a Router Solicitation message
  - Has a lot of important information for the host

Yes, I am here!
Neighbor Discovery Protocol

- Neighbor Solicitation
  - Sent by a node to find the MAC-address of the neighbor, or to check if the neighbor is still reachable

Are you still there?
Neighbor Discovery Protocol

- Neighbor Advertisement
  - A response to a neighbor solicitation message

Yes, I am still here!
Neighbor Discovery Protocol

• Redirect

- A router points the host to a better first hop router for a destination

You can better go see that guy over there!
Questions
IPv6 Routing Basics

2 - Section
IPv6 Routing Basics

- IPv6 routing is the same as IPv4 routing
  - Longest matching prefix
  - Same structure and concepts
  - Some technical differences
## Longest Matching Prefix

- Example routing table:

<table>
<thead>
<tr>
<th>Route</th>
<th>Next-Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>::/0</td>
<td>2001:db8:aaa:bbb::cdef:1</td>
</tr>
<tr>
<td>2001:db8::/32</td>
<td>2001:db8:bcd:aaa::1</td>
</tr>
<tr>
<td>2001:db8::/48</td>
<td>2001:db8:cde:bbb::1</td>
</tr>
<tr>
<td>2001:db8:1000::/36</td>
<td>2001:db8:fff:eeee::1</td>
</tr>
<tr>
<td>2001:db8:1000::/48</td>
<td>2001:db8:def:bbb::1</td>
</tr>
</tbody>
</table>
Longest Matching Prefix

- Matches for a packet with destination:
  2001:db8:2000:1a2b:02ab:9eff:fe01:f5b1

<table>
<thead>
<tr>
<th>Route</th>
<th>Next-Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>::/0</td>
<td>2001:db8:aaa:bbb::cdef:1</td>
</tr>
<tr>
<td>2001:db8::/32</td>
<td>2001:db8:bcd:aaa::1</td>
</tr>
<tr>
<td>2001:db8::/48</td>
<td>2001:db8:cde:bbb::1</td>
</tr>
<tr>
<td>2001:db8:1000::/36</td>
<td>2001:db8:ffff:eeee::1</td>
</tr>
<tr>
<td>2001:db8:1000::/48</td>
<td>2001:db8:def:bbb::1</td>
</tr>
</tbody>
</table>
# Longest Matching Prefix

- Matches for a packet with destination: `2001:db8:2000:1a2b:02ab:9eff:fe01:f5b1`

<table>
<thead>
<tr>
<th>Route</th>
<th>Next-Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>::/0</code></td>
<td><code>2001:db8:aaa:bbb::cdef:1</code></td>
</tr>
<tr>
<td><code>2001:db8::/32</code></td>
<td><code>2001:db8:bcd:aaa::1</code></td>
</tr>
<tr>
<td><code>2001:db8::/48</code></td>
<td><code>2001:db8:cde:bbb::1</code></td>
</tr>
<tr>
<td><code>2001:db8:1000::/36</code></td>
<td><code>2001:db8:ffff:eeee::1</code></td>
</tr>
<tr>
<td><code>2001:db8:1000::/48</code></td>
<td><code>2001:db8:def:bbb::1</code></td>
</tr>
</tbody>
</table>
Longest Matching prefix

- Matches for a packet with destination:
  2001:db8:2001:1a2b:02ab:9eff:fe01:f8b2

<table>
<thead>
<tr>
<th>Route</th>
<th>Next-Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>::/0</td>
<td>2001:db8:aaa:bbb:cdef:1</td>
</tr>
<tr>
<td>2001:db8::/32</td>
<td>2001:db8:bcd:aaa::1</td>
</tr>
<tr>
<td>2001:db8::/48</td>
<td>2001:db8:cde:bbb::1</td>
</tr>
<tr>
<td>2001:db8:1000::/36</td>
<td>2001:db8:ffff:eeee::1</td>
</tr>
<tr>
<td>2001:db8:1000::/48</td>
<td>2001:db8:def:bbb::1</td>
</tr>
</tbody>
</table>
Longest Matching prefix

- Matches for a packet with destination: 2001:db8:2001:1a2b:02ab:9eff:fe01:f8b2

<table>
<thead>
<tr>
<th>Route</th>
<th>Next-Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>::/0</td>
<td>2001:db8:aaa:bbb::cdef:1</td>
</tr>
<tr>
<td>2001:db8::/32</td>
<td>2001:db8:bcd:aaa::1</td>
</tr>
<tr>
<td>2001:db8::/48</td>
<td>2001:db8:cde:bbb::1</td>
</tr>
<tr>
<td>2001:db8:1000::/36</td>
<td>2001:db8:ffff:eeee::1</td>
</tr>
<tr>
<td>2001:db8:1000::/48</td>
<td>2001:db8:def:bbb::1</td>
</tr>
</tbody>
</table>
## Longest Matching Prefix

- Matches for a packet with destination: `2001:db8:1001:1a2b:02ab:92ff:fe01:f8b2`

<table>
<thead>
<tr>
<th>Route</th>
<th>Next-Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>::/0</td>
<td>2001:db8:aaa:bbb::cdef:1</td>
</tr>
<tr>
<td>2001:db8::/32</td>
<td>2001:db8:bcd:aaa::1</td>
</tr>
<tr>
<td>2001:db8::/48</td>
<td>2001:db8:cde:bbb::1</td>
</tr>
<tr>
<td>2001:db8:1000::/36</td>
<td>2001:db8:ffff:eeee::1</td>
</tr>
<tr>
<td>2001:db8:1000::/48</td>
<td>2001:db8:def:bbb::1</td>
</tr>
</tbody>
</table>
Longest Matching Prefix

- Matches for a packet with destination: 2001:db8:1001:1a2b:02ab:92ff:fe01:f8b2

<table>
<thead>
<tr>
<th>Route</th>
<th>Next-Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>::/0</td>
<td>2001:db8:aaa:bbb::cdef:1</td>
</tr>
<tr>
<td>2001:db8::/32</td>
<td>2001:db8:bcd:aaa::1</td>
</tr>
<tr>
<td>2001:db8::/48</td>
<td>2001:db8:cde:bbb::1</td>
</tr>
<tr>
<td>2001:db8:1000::/36</td>
<td>2001:db8:ffff:eeee::1</td>
</tr>
<tr>
<td>2001:db8:1000::/48</td>
<td>2001:db8:def:bbb::1</td>
</tr>
</tbody>
</table>
Summary

• IPv6 routing uses the same structure as IPv4 routing
  - Addresses are longer
  - Prefixes are longer
Add IPv6 to Loopback & Links

3 - Exercise
Discover the IPv4 Network

- Make sure you have connectivity
- Go to: workbench.ripe.net
- Your login is your number on the attendee list
- We will provide you with the password

- Read instructions carefully
- First discover, then configure
Discover the IPv4 Network

• Routing Protocol
  - IGP (OSPF) is used for loopback addresses and point-to-point links
  - EGP (BGP) is used for the edge core routers

• R1 announces a default route via OSPF
• Keeps routing tables in the area smaller
  - All inter-area traffic must pass R1
Adding IPv6 to the Network

• We will now add IPv6 to our existing network
• We will not change the network structure

• First step: Addressing Structure
• Find the addresses on the handout
Addressing with IPv6

Where X is your number on the attendee list!

- **Loopbacks:**
  - There is a /32 (IPv4): 172.X.255.Y (router number)
  - Use a /128 (IPv6) 2001:ffXX::Y (router number)/128

- **Point-to-point core:**
  - There is a /30 (IPv4) from 10.X.0.0/24
  - Use a /127 from 2001:ffXX::/60 for core links
  - Use a /64 from 2001:ffXX::/60 for the customer links
Interface IPv6 Settings Routers

- Disable Router Advertisements
  - On point-to-point interfaces
  - On LANs where unprepared devices are connected

- Otherwise they will suddenly be globally reachable over IPv6 without being configured, prepared and/or protected
Basic IPv6 Settings

• Before configuring IPv6 on your router interfaces, the basic IPv6 settings need to be set up on the router

• For R1, R2, R3 and the Customers

```plaintext
ipv6 unicast-routing
ipv6 cef
```
Interface IPv6 Settings Routers

- Use the information in the handouts
- Give the correct IPv6 addresses to the interfaces
- Start with the loopback interface
- Then configure the point-to-point links
- Fill in the appropriate interface name, IPv6 address and prefix length

```plaintext
interface xyz
  ipv6 address ...
  no ipv6 redirects
  ipv6 nd ra suppress all
```
Interface IPv6 Settings Customers

- Use the information in the handouts
- Give the correct IPv6 addresses to the interfaces
- We don’t configure loopback interfaces
- Then configure the point-to-point link
- Fill in the appropriate interface name, IPv6 address and prefix length (/64)
- We don’t disable router advertisements

```plaintext
interface xyz
ipv6 address ... no ipv6 redirects
```
Interface IPv6 Settings Customers

• We will set a default route for the customers
• This is a manual configuration
• This is not needed if you use SLAAC

```
ipv6 route ::/0 2001:ffXX:0:ff01::b (customer 1)
ipv6 route ::/0 2001:ffXX:0:ff02::b (customer 2)
```

• **XX** is your number on the list
Checking Your Configuration

• Check your own configuration
  - Can you ping your own IPv6 loopback address?
  - Can you ping your own side of the point-to-point link?
Questions
OSPF Characteristics

- OSPF = Open Shortest Path First
- Link State Protocol
- OSPFv3 is an implementation of OSPF for IPv6
- OSPFv2 (for IPv4) and OSPFv3 run independently on the router
- Most OSPFv3 functions are the same as OSPFv2
OSPF Refresher

• Link state protocol
  - Every router has full insight into network topology of the area
  - Routes are sent to other routers using Link State Advertisements (LSAs)

• Role of Area Border Routers:
  - Limit the flooding of LSAs to isolate topology changes within the area
OSPF Refresher

External LSA flooding across all areas

LSA flooding within the area
OSPF for IPv6

• Multiple instances of OSPFv3 can be run on a link
  - Unlike in OSPFv2

• OSPFv3 still uses 32-bit numbers as a router ID
  - If no IPv4 address is configured on any interface, the router ID command is required to set the 32-bit router ID (for IPv6-only networks)
**OSPF for IPv6**

- Router ID is a unique identifier for the router
  - Must be configured in the routing process
  - Is still a 32-bit number, written in 4 octets
  - It is used to sign routing updates

- But to make your life easy, you can use an IPv4 loopback address
OSPF for IPv6

• OSPF for IPv4 (OSPFv2) can be configured:
  - on each subnet or,
  - on each link

• OSPF for IPv6 (OSPFv3) can be configured:
  - on each link

• Interface mode configuration will automatically activate the OSPF process on your running config
• LSA types and functions in OSPF are almost the same as for OSPFv2
  - But there is no authentication in OSPFv3

• OSPFv3 uses multicast addresses:
  - ff02::5 for All OSPFv3 Routers
  - ff02::6 for All OSPFv3 Designated Routers

• All OSPFv3 adjacencies are formed using link-local addresses
  - From fe80::/10 IPv6’s link-local address scope
Configuration of OSPF as IGP

• Example of OSPF for IPv4 per-subnet configuration

```conf
router ospf 1
  log-adjacency-changes
  passive-interface default
  network 172.16.1.1 0.0.0.0 area 1
  no passive-interface e0/0
  network 172.16.11.8 0.0.0.3 area 1
  no passive-interface e0/1
  network 172.16.11.0 0.0.0.3 area 1
```
Configuration of OSPF as IGP

- Example of OSPF for IPv4 per-link configuration

```
router ospf 1
  log-adjacency-changes
  passive-interface e1/1
  passive-interface e1/0
!
interface loopback 0
  ip ospf 1 area 1
!
interface e0/0
  ip ospf 1 area 1
!
interface e0/1
  ip ospf 1 area 1
!
```
Configuration of OSPF as IGP

- Example of OSPF for IPv6 per-link configuration

```
ipv6 router ospf 1
  log-adjacency-changes
  passive-interface e1/1
  passive-interface e1/0

interface loopback 0
  ipv6 ospf 1 area 1

interface e0/0
  ipv6 ospf 1 area 1

interface e0/1
  ipv6 ospf 1 area 1
```
Configuring OSPFv3

5 - Exercise
Overview of IGP Configuration

- You have to configure OSPFv3 as IGP for IPv6
- Dual Stack will be used to ensure both IPv4 and IPv6 operation
- OSPFv2 is already set up
Have a good look…

- At the IPv4 configuration…

```
show running-config | s router ospf
```
OSPFv3 Global Settings

- Tell the router to do OSPFv3 and the process-id
- Log adjacency changes
- Set a router ID
- Define passive interface

```plaintext
ipv6 router ospf 1
   log-adjacency-changes
   router-id 172.X.255.Y  (Y is router number)
   passive-interface Ethernet1/0
   passive-interface Ethernet1/1
   redistribute connected
```

- On router 1 also add:

```plaintext
ipv6 router ospf 1
   default-information originate always
```
OSPFv3 Interface Settings

- OSPFv3 interface settings

```plaintext
interface xyz
  ipv6 ospf network point-to-point
  ipv6 ospf 1 area 0
```

- Fill in the appropriate interface names and OSPF area
Checking Your Configuration

• Check your own configuration
  - Can you ping the loopback on R3 from C1?
  - Can you ping the loopback on R2 from C2?
Checking Your Configuration

• You should now have a running IPv6 core network!

• For every internal IPv4 route there should be a corresponding IPv6 route

• Try to ping and traceroute point-to-point connections and loopback addresses in your part of the network
BGP

6 - Section
BGP Overview

- Routing Protocol used to exchange routing information between networks
  - Exterior Gateway Protocol
- It is based on Path Vector Protocol
  - Similar to Distance Vector
- Each border router sends to its neighbors the full route to one destination, not just the distance
Autonomous System

• Collection of networks with the same routing policy

• Usually under single ownership and administrative control
  - Single routing policy

• Identified by 16 or 32 bit numbers
  - 16bit: 0 - 65,535
  - 32bit: 65,536 - 4,294,967,295
AS Path

- Sequence of ASes a route has traversed
  - Loop detection
  - Path selection (AS-PATH length)
BGP Modes

- **eBGP**: Between BGP speakers in a different AS

  ![eBGP Diagram]

- **iBGP**: Between BGP speakers within the same AS

  ![iBGP Diagram]
BGP Messages

- **OPEN**
  - opens the tcp session

- **KEEPALIVE**
  - keeps the session running

- **NOTIFICATION**
  - error handling

- **UPDATE**
  - actual route updates (NLRI, AS-path, AS-path attributes)
**NLRI**

- **Network Layer Reachability Information**
  - Used to advertise feasible routes
  - Composed of:
    - Network Prefix
    - Mask Length
BGP Path Attributes

• Well known
  - They are known by all the routers and passed to BGP neighbors
  - Mandatory and are included in the UPDATE messages

• Optional
  - May not be supported by all BGP implementations
  - The transitive bit determines if an optional attribute is passed to BGP neighbors
Multiprotocol BGP (MP-BGP)

- Extension to the BGP protocol
- Carries routing information about other protocols:
  - Multicast
  - MPLS VPN
  - IPv6
- Multi-Protocol NLRI exchange is negotiated at session set up (OPEN Message)
MP-BGP

• New features in OPEN Message:
  - BGP Capabilities Advertisement:
    - Address Family Identifier (AFI)
    - Subsequent Address Family Identifier (SAFI)
  - Multiprotocol Reachable Network Layer Reachability Information
AFI / SAFI

• Address Family Identifier (AFI)
  - Identifies Address Type
  - AFI = 1 (IPv4)
  - AFI = 2 (IPv6)

• Subsequent Address Family Identifier (SAFI)
  - Sub category for AFI Field
  - Address Family Identifier (AFI)
    • Sub-AFI = 1 (NLRI is used for unicast)
    • Sub-AFI = 2 (NLRI is used for multicast RPF check)
    • Sub-AFI = 3 (NLRI is used for both unicast and multicast RPF check)
    • Sub-AFI = 4 (label)
    • Sub-AFI = 128 (VPN)
MP-BGP Capabilities Negotiation

- BGP routers establish peering sessions through the OPEN message
- OPEN message contains optional parameters
- BGP session is terminated if OPEN parameters are not recognised
- A new optional parameter: CAPABILITIES containing its capabilities:
  - Multiprotocol extension (AFI/SAFI)
  - Route Refresh
  - Outbound Route Filtering
Managing Multiple Protocols

• Independent operation
  - One RIB per protocol
  - Distinct policies per protocol (IP address specific route maps and prefix lists must be adjusted)
  - Make separate route maps for IPv4 and IPv6
  - Prefix lists are always separate
  - It is common to use a _v4 and a _v6 suffix to names
Configuring MP-BGP & Customers

7 - Exercise
eBGP

7.1 - Exercise
BGP Configuration R1

- Cisco defaults to address-family IPv4
- This must be disabled before configuring IPv6
- Your AS Number is $1 + \text{your number on the participants list (e.g. 109)}$

```
router bgp 1XX
  no bgp default ipv4-unicast
```
Set the Route and Prefix list on R1

```
address-family ipv6
    network 2001:ffXX::/32
(exit)
ipv6 route 2001:ffXX::/32 Null0
ipv6 prefix-list filter_v6 seq 5 permit 2001:ffXX::/32
```
BGP Configuration R1

- Now we are going to set up BGP to our upstreams
- We use the same settings for IPv6 as we have for IPv4
- Only configure R1

```
router bgp 1xx
neighbor 2001:ff69::66 remote-as 66
neighbor 2001:ff69::99 remote-as 99
```
BGP Configuration R1

- And activate the external session in the correct address family

```conf
address-family ipv6
  redistribute static
neighbor 2001:ff69::66 activate
neighbor 2001:ff69::99 activate
neighbor 2001:ff69::66 prefix-list filter_v6 out
neighbor 2001:ff69::99 prefix-list filter_v6 out
```
Filtering

• We filter the routes we announce

  - Why?
  - Why in this way?
  - What are the differences in IPv4 and IPv6?
Now we are going to set up BGP on top of our IPv4 core
We use the same settings for IPv6 as we have for IPv4

```
neighbor 2001:ffXX::2 remote-as 1XX
neighbor 2001:ffXX::2 update-source lo0
neighbor 2001:ffXX::3 remote-as 1XX
neighbor 2001:ffXX::3 update-source lo0

address-family ipv6
  redistribute static
  neighbor 2001:ffXX::2 activate
  neighbor 2001:ffXX::3 activate
  neighbor 2001:ffXX::2 next-hop-self
  neighbor 2001:ffXX::3 next-hop-self
```
BGP Configuration R2

- Now we are going to set up BGP on top of our IPv4 core
- We use the same settings for IPv6 as we have for IPv4

```plaintext
router bgp 1XX
  no bgp default ipv4-unicast

neighbor 2001:ffXX::1 remote-as 1XX
neighbor 2001:ffXX::1 update-source lo0
neighbor 2001:ffXX::3 remote-as 1XX
neighbor 2001:ffXX::3 update-source lo0

address-family ipv6
  redistribute static
  neighbor 2001:ffXX::1 activate
  neighbor 2001:ffXX::3 activate
  neighbor 2001:ffXX::1 next-hop-self
  neighbor 2001:ffXX::3 next-hop-self
```
BGP Configuration R3

- Now we are going to set up BGP on top of our IPv4 core
- We use the same settings for IPv6 as we have for IPv4

```
router bgp 1xx
  no bgp default ipv4-unicast

neighbor 2001:ffxx::1 remote-as 1xx
neighbor 2001:ffxx::1 update-source lo0
neighbor 2001:ffxx::2 remote-as 1xx
neighbor 2001:ffxx::2 update-source lo0

address-family ipv6
  redistribute static
  neighbor 2001:ffxx::1 activate
  neighbor 2001:ffxx::2 activate
  neighbor 2001:ffxx::1 next-hop-self
  neighbor 2001:ffxx::2 next-hop-self
```
BGP Customer1

7.3 - Exercise
Network Diagram

CUSTOMER 1

CUSTOMER 2

R2

R1

R3

AS66

AS99

POP

e0/0  e0/1  e0/0  e0/0  e0/0  e1/0  e0/0  e0/1  e0/0  e0/0  e0/0

e1/0  e1/0  e1/0  e0/0  e0/0  e1/0
BGP Configuration Customer 1

- We will remove the default route for the customers

```bash
no ipv6 route ::/0 2001:ffXX:0:ff01::b
```
BGP Configuration Router 2

- The AS number for customer is 2 + your number on the participants list (e.g. 209)
- Add BGP session for Customer 1

```bash
router bgp 1XX
  neighbor 2001:ffXX:0:ff01::a remote-as 2XX

address-family ipv6
  neighbor 2001:ffXX:0:ff01::a activate
```
BGP Configuration Router 2

- Now add customer prefix to the prefix list
  - Customer 1 prefix: 2001:ffXX:ff01::/48

```
ipv6 prefix-list customer1_v6 seq 5 permit 2001:ffXX:ff01::/48
```

```
router bgp 1XX
  address-family ipv6
    neighbor 2001:ffXX:0:ff01::a prefix-list customer1_v6 in
```

```
clear bgp ipv6 unicast 2001:ffXX:0:ff01::a soft in
```
BGP Configuration Customer 1

- The AS number for customer is 2 + your number on the participants list (e.g. 209)
- Configure BGP session with R2

```
router bgp 2XX
    no bgp default ipv4-unicast
    neighbor 2001:ffXX:0:ff01::b remote-as 1XX

address-family ipv6
    redistribute static
    neighbor 2001:ffXX:0:ff01::b activate
```
BGP Configuration Customer 1

- Now add the prefix, prefix list and static route
  - Customer1 prefix: 2001:ffXX:ff01::/48

```
address-family ipv6
  network 2001:ffXX:ff01::/48
(exit)
ipv6 route 2001:ffXX:ff01::/48 Null0
ipv6 prefix-list my_v6 seq 5 permit 2001:ffXX:ff01::/48
```

```
router bgp 2XX
  address-family ipv6
    neighbor 2001:ffXX:0:ff01::b prefix-list my_v6 out
```
Challenge: BGP Customer 2

7.4 - Exercise
Network Diagram
BGP Configuration Customer 2

- Configure BGP session between Customer 2 router and provider R3
- The AS number for customer is 3 + your number on the participants list (e.g. 309)
- Add the prefix, prefix list and static route
  - Customer2 prefix: 2001:ffXX:ff02::/48
Summary

• We now added IPv6 to…
  - Links/interfaces
  - IGP (OSPF)
  - EGP (BGP)
  - Customers

• How difficult was it?

• Any surprises?
Questions
Content

8 - Section
Definition

• This section is mostly about websites but it can also apply to:
  - SMTP
  - POP3
  - IMAP4
  - SSH
  - Remote Desktops
  - Cloud Services
Options

- Multiple ways to make content dual stack
  - Native dual stack
  - Dual stack load balancer
  - IPv6-to-IPv4 (reverse) proxy
  - NAT64
Native Dual Stack

- If possible this is the preferred option

- Cleanest option: no mixing of IPv4 and IPv6
- Needs a fully dual stacked network
- All addresses fully visible where possible
Load Balancer with NAT or Proxy

- If web servers can’t handle IPv6
  - Needs a fully dual stacked network up to the load balancer
  - Web servers might not see IPv6 addresses
IPv6-to-IPv4 Proxy

- If the load balancer or part of the network can’t handle IPv6

- Web servers might not see IPv6 addresses
Proxy Protocol Level

- You can proxy on
  - Layer 4 (TCP)
  - Layer 7 (HTTP/HTTPS)

- Both have advantages and disadvantages
Proxy on Layer 4

• Very easy to configure
  - Doesn’t need to know about the protocol
  - Doesn’t need to be configured with host name
  - Don’t need SSL/TLS keys on the proxy server

• Just map an IPv6 address+port to an IPv4 address+port

• Cannot provide information to the servers
Proxy on Layer 4

- This example shows haproxy
  - Note the confusing notation in the config file
  - IPv6 address = 2001:db8:abc:123::cafe port 25

```plaintext
listen smtp1
  bind 2001:db8:abc:123::cafe:25
  mode tcp
  server smtp1 192.0.2.1:25
```
Proxy on Layer 7

• Bit more work to configure
  - Needs to know about the protocol or application
  - Might need to be configured with host name
  - Needs SSL/TLS keys on the proxy server

• Can provide information to the servers
  - Like X-Forwarded-For header
Proxy on Layer 7

- This example shows *haproxy*

```plaintext
listen website1
  bind 2001:db8:abc:123::cafe:80
  mode http
  option forwardfor
  server website1 192.0.2.1:80

listen website1-ssl
  bind 2001:db8:abc:123::cafe:443 ssl
  crt /etc/haproxy/website-ssl.pem
  mode http
  option forwardfor
  server website1 192.0.2.1:443 ssl
```

- With **SSL**
Happy Eyeballs

• Makes dual-stacked websites more responsive to users

• If there is both A and AAAA
  - First IPv6 is used with a 300 ms head start
  - If that fails, IPv4 is used

• Implemented by all browsers

• Instable connections can cause problems with cookies
IPv6 in the Root Servers and TLDs

• All 13 root servers have IPv6 AAAA records

• There are 1530 TLDs
  - 1501 of them are IPv6 capable (98.2%)

• Over 9 billion websites have AAAA records

Source: http://bgp.he.net/ipv6-progress-report.cgi
IPv6 in Mobile Networks

- IPv4 runout has a high and urgent impact on mobile internet providers
- Everyone has a smartphone
- Certain apps and protocols have problems with double NAT

- Do apps support IPv6?
Multiple Solutions

• Dual Stack users:
  - Public IPv4 and public IPv6
  - Private IPv4 and public IPv6

• IPv6 only users:
  - NAT64
  - IPv6 only
NAT64/DNS64

Mobile User

public IPv6

CUSTOMER

Infrastructure
public IPv6

PROVIDER

NAT64 Box

DNS64

IPv6 Internet

IPv4 Internet

INTERNET
464XLAT

- Extension to NAT64 to access IPv4-only applications (like Skype or Whatsapp)

- Handset pretends there is an IPv4 address (CLAT) and sends IPv4 packets in UDP over IPv6
  - Support from: Android 4.4 and Samsung Galaxy Note, Galaxy Light and Google Nexus
464XLAT

IPv6 UDP

464XLAT Client

Mobile User
IPv6 only

IPv6 UDP

GGSN

3G/4G Network
IPv6 only

PLAT Box

INTERNET
IPv6 Internet

IPv4 Internet

CUSTOMER

PROVIDER
Apple Approach

• Apple announced they will not support 464XLAT on the iPhone

• Instead they urge app developers to make apps work over IPv6-only

• That way, operators can use just NAT64 without 464XLAT
3G

- Works with Packet Data Protocol (PDP) Contexts
  - Initiated by the phone to establish a connection
  - IPv4, IPv6 and IPv4v6

- No requirement for always-on connection
- Only works with SLAAC
4G

- Works with Evolved Packet System (EPS) Bearer
  - Initiated by the phone to establish a connection
  - IPv4, IPv6 and IPv4v6

- Always-on Packet Data Network (PDN) Connection

- Due to the need for supporting VoIP
What Needs To Be Done

Source: Cameron Byrne T-Mobile USA
What Needs To Be Done

• Handset:
  - IPv6 capable

• Home Location Register (HLR)
  - Subscriber management needs to understand new PDP types (IPv6, IPv4v6)

• Serving GPRS Support Node (SGSN)
  - ‘IPv6 on user plane’ needs to be enabled
  - Fallback strategy
What Needs To Be Done

- Gateway GPRS Support Node (GGSN)
  - New PDP types (IPv6, IPv6v4)
  - IPv6 routing
  - DHCPv6
  - Neighbor Discovery Protocol
  - DNS Configuration
  - Fallback strategy
  - Billing
What Needs To Be Done

• And the usual....
  - Firewalls, servers, etc.
Tethering & IPv6

• A /64 prefix is received through an RA to the phone
• An /128 from that /64 is used for the own WAN
• The same /64 is used for the LAN (and for tethering)
  - Tethering is done through RA
  - DAD is used to avoid duplicate addresses
Challenges

• Only 1 IPv4 address and 1 IPv6 subnet on a handset

• Fallback from IPv4v6 to IPv4-only or IPv6-only is difficult in some cases
Overview Day 2

- Host Configuration
- Exercise: SLAAC
- DHCPv6
- Exercise: DHCPv6
- Security
- Exercise: NAT64/DNS64
- IP Address Management
- Tips & Tricks
Host Configuration

Section 10
Operating Systems

• We will look at Windows, Linux, OSX
• All of them support IPv6 natively

• Good news: it works automatically
• Bad news: it works automatically
Managing clients

• Users might not notice that their computer is using IPv6

• For management purposes, you want control over addresses
  • Disable SLAAC, Privacy extensions, other
Obtaining addresses

- Disabling SLAAC does not mean disabling Router Advertisements

- RAs are an important part of address distribution
  - They point clients at a DHCPv6 server
Windows 7

• By default, many services/protocols are enabled:
  • Privacy extensions
  • Teredo
  • 6to4
  • ISATAP

• You might want to disable some/all of them
Windows 7: Before
De-configuring Windows 7

- First, disable all the transition methods

- On the command prompt, as administrator:

```plaintext
netsh interface ipv6 6to4 set state state=disabled
netsh interface ipv6 isatap set state state=disabled
netsh interface ipv6 set teredo disable
```
Windows 7 Privacy Extensions

- Disable privacy extensions

```
netsh interface ipv6 set privacy state=disable
netsh interface ipv6 set global randomizeidentifier=disabled
```
Windows 7: After
Windows 10

• By default, many services/protocols are disabled:
  • Privacy extensions
  • Teredo
  • 6to4
  • ISATAP

• Unfortunately, Windows 10 does not do DHCP out-of-the-box
DHCPv6 client on Windows

- First, get the interface ID:

```cmd
netsh interface ipv6 show interfaces
```
DHCPv6 client on Windows

• With the interface ID instead of the red XX, run the command:

```
netsh interface ipv6 set interface ipv6 XX advertise=enabled managed=enabled
```

• (This has to be run as an Administrator)
Check configuration

- To check the configuration:
  
  ```
  netsh interface ipv6 show interface XX
  ```

- (This has to be run as an Administrator)
Activating DHCPv6

- Without DHCPv6:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neighbor Discovery</td>
<td>enabled</td>
</tr>
<tr>
<td>Neighbor Unreachability Detection</td>
<td>enabled</td>
</tr>
<tr>
<td>Router Discovery</td>
<td>enabled</td>
</tr>
<tr>
<td>Managed Address Configuration</td>
<td>disabled</td>
</tr>
<tr>
<td>Other Stateful Configuration</td>
<td>disabled</td>
</tr>
<tr>
<td>Weak Host Sends</td>
<td>disabled</td>
</tr>
<tr>
<td>Weak Host Receives</td>
<td>disabled</td>
</tr>
</tbody>
</table>

- With DHCPv6:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neighbor Discovery</td>
<td>enabled</td>
</tr>
<tr>
<td>Neighbor Unreachability Detection</td>
<td>enabled</td>
</tr>
<tr>
<td>Router Discovery</td>
<td>enabled</td>
</tr>
<tr>
<td>Managed Address Configuration</td>
<td>enabled</td>
</tr>
<tr>
<td>Other Stateful Configuration</td>
<td>disabled</td>
</tr>
<tr>
<td>Weak Host Sends</td>
<td>disabled</td>
</tr>
<tr>
<td>Weak Host Receives</td>
<td>disabled</td>
</tr>
</tbody>
</table>
Windows and DHCPv6

- You can either:
  - Configure a router to supply the “M” flag
    - But with no prefix announced
  - Or disable router discovery
    - And let other clients pick up addresses from SLAAC
OSX

- It will automatically configure IPv6

- It will look in the RA messages to check M flag
  - If present, it will check with DHCPv6
OSX Configuration

Configure IPv4: Using DHCP
IPv4 Address: 193.0.10.84
Subnet Mask: 255.255.255.0
Router: 193.0.10.1
Configure IPv6: Automatically
Router: fe80::13:0:0:1
IPv6 Address

en0: flags=8863<UP,BROADCAST,SMART,_RUNNING,_SIMPLEX, MULTICAST> mtu 1500
ether 64:76:ba:98:3d:d6
inet6 fe80::6676:baff:fe98:3dd6%en0 prefixlen 64 scopeid 0x4
inet 193.0.10.84 netmask 0xffffff00 broadcast 193.0.10.255
nd6 options=1<PERFORMNUD>
media: autoselect
status: active
Linux

• As client, same behaviour as OSX
  • Everything works out of the box
  • IPv6 is enabled automatically

• As server, static configuration is required
Linux Static configuration

- For CentOS/Red Hat:
- `/etc/sysconfig/network`

- Add:

```
NETWORKING_IPV6=yes
```
Linux Static configuration

- In /etc/sysconfig/network-scripts/ifcfg-eth\(X\)

- Add:

```
IPV6INIT=yes
IPV6ADDR=2001:0db8:aaaa:bbbb:0000:0000:0000:0002/64
IPV6_DEFAULTGW=2001:db8:aaaa:bbbb:0000:0000:0000:0001
IPV6_AUTOCONF=no
```

- Where \(X\) is the number of the interface, then:

```
service network restart
```
On C1

- Now we will enable SLAAC

```bash
interface e0/0
ipv6 address autoconfig default
no shutdown
```

- Leave configuration mode
- Enable debug ND

```bash
debug ipv6 nd
```
On R2

• Now we will remove the suppression

```bash
interface e1/0
    ipv6 address 2001:ffXX:0:ff01::b/64
    no ipv6 nd ra suppress all
    no shutdown
```

• Leave configuration mode

• To stop debug messages on C1

```bash
undebug all
```
Debugging SLAAC

• Can you find the new IPv6 address?
• Look at the routing…
• Do you see any interesting debug messages?
SLAAC: Router Messages

Link-local: fe80::a390:45ff:fe14:3f0f
Global unicast: 2001:db8:a:b::1

Multicast Address

1 2 3 4 5 6 7 8
NS NA RA NA RA RA RA FF02::2
FF02::1 FF02::1:FF14:3F0F
FF02::1:FF00:1 FF02::1:FF05:1C9E

Time

Link-local: fe80::ba8d:12ff:fe05:1c9e
SLAAC: Client Messages

Link-local: fe80::a390:45ff:fe14:3f0f
Global unicast: 2001:db8:a:b::1

Link-local: fe80::ba8d:12ff:fe05:1c9e
Questions
About DHCPv6

• New protocol
• Requires IPv6 transport
• Offer similar functionality to DHCPv4 but for IPv6
• Allows more control than SLAAC
  - Routers and servers can have static or dynamic assignments

• Is supported by Cisco IOS, Microsoft, etc.
Information provided by DHCPv6

- No routing information is distributed
  - no default route (routers in IPv6 have different roles in the network)

- Only host configuration protocol

- Other configuration parameters
  - includes DNS, NTP etc
DHCPv6 Fundamentals

- Client driven via DHCPv6 request message

- Solely layer 3 protocol unlike DHCPv4:
  - port 546 for clients
  - port 547 for server

- DHCPv6 options are similar to those in DHCPv4
DHCPv6 Operation

• Client first detects the presence of routers on link

• Client examines router advertisements to check if DHCP can be used (managed flag)

• If no router is found or if DHCP can be used, the client:
  - sends DHCP solicit message to “all-DHCP-agents” multicast address (ff02::1:2)
  - uses link-local address as source address
DUID

• DHCP Unique IDentifier

• A globally unique identifier used to identify the single machine/device
  - One DUID per DHCPv6 client

• DHCPv6 does not use only MAC address as identifier

• Variable length between 96 - 160 bits
  - Example Client DUID:
    00030001001A2F875602
DHCPv6 Modes

• Stateful
  - Also requesting an address
  - M flag

• Stateless
  - Only other configuration parameters
  - O flag

• Prefix Delegation
Stateful DHCPv6

• Similar to DHCPv4 today
• A router can act as a DHCP server
• Configuration parameters include:
  - DHCP pool name
  - Prefix information
  - List of DNS servers
  - Addresses for clients
Stateful DHCPv6 Server

- Responds to requests from clients to:
  - Offer IPv6 addresses
  - Other configuration parameters (DNS servers...)

- Listens on the following multicast addresses:
  - All_DHCP_Relay_Agents_and_Servers (FF02::1:2)
  - All_DHCP_Servers (FF05::1:3)

- Usually stores client’s state
Stateful DHCPv6 Client and Relay

- **Client**
  - Initiates requests on a link to obtain configuration
  - Uses its link local address to connect the server
  - Sends requests to FF02::1:2 multicast address

- **Relay agent**
  - A node that acts as an intermediary to deliver DHCP messages between clients and servers
  - On the same link as the client
  - Listens on FF02::1:2 multicast address
Stateful DHCPv6 Messages

SERVER

CLIENT

SOLICIT

ADVERTISE

REQUEST

REPLY
Stateless DHCPv6

- Complements SLAAC configuration:
  - I.e: host obtain the address using SLAAC and the DNS server address from DHCPv6
  - In dual-stack networks we can obtain IPv4 DNS server addresses from DHCPv4

- Configure a DHCP pool with additional parameters:
  - DNS Server
  - Domain name
  - NTP

- Activated by “other configuration” flag in ND
Stateless DHCPv6 Messages

SERVER

CLIENT

INFORMATION-REQUEST

REPLY
IPv6 Prefix Delegation

• IPv4 deployments:
  - ISP only has to deliver a public IPv4 address
  - NAT is used for translation using RFC1918

• IPv6 deployments:
  - IPv6 end-to-end reachability:
  - Home network gets its own IPv6 prefix (public address)
  - No NAT
DHCPv6 Prefix Delegation

- ISP assigns a block of addresses for delegation to customers (e.g. /48)
- Customer assigns /64 prefixes to LAN interfaces
DHCPv6 Prefix Delegation

- Provider edge as delegating DHCP server
- CPE as DHCP client and IPv6 router
DHCPv6 PD Messages

SERVER

CLIENT

SOLICIT

ADVERTISE

REQUEST

REPLY
Questions
DHCPv6-PD

12 - Exercise
DHCPv6
**DHCPv6-PD Router Configuration**

- DHCP pool named “DHCP_CUSTOMERS” references local pool “DHCP_POOL”
- DHCP_POOL details about the address pool

```plaintext
ipv6 dhcp pool DHCP_CUSTOMERS
  prefix-delegation pool DHCP_POOL
!
interface e1/0
  ipv6 address 2001:ffXX:0::ff02::b/64
  ipv6 dhcp server DHCP_CUSTOMERS
  no shutdown
!
ipv6 local pool DHCP_POOL 2001:ffXX:ff02::/48 56
```
DHCPv6-PD C2 Configuration

- ISP facing interface is the DHCP client
- LAN facing interface is the IPv6 router sending RA message

```bash
interface e0/0
  ipv6 address 2001:ffXX:0:ff02::a/64
  ipv6 dhcp client pd PREFIX
  no shutdown
!
interface e0/1
  ipv6 address PREFIX ::1:b00c:caf3:bab3:1/64
  no ipv6 redirects
  no shutdown
```
On H1

- Enable IPv6 on the router
  - You know how

- And enable SLAAC

```bash
interface e0/0
  ipv6 address autoconfig default
  no shutdown
```
Summary

• We have now distributed an IPv6 prefix to Customer 2

• Customer has distributed prefixes to LAN interfaces automatically and distributed SLAAC to the host

• Can you find the DUID address of C2?
DHCPv6-PD with static assignment

12 - Exercise
DHCPv6-PD with Static Assignment

• Assign to Customer 2 static prefix:
  - 2001:ffXX:ff02:AB00::/56
On R3

- Find the DUID of the customer

```bash
#show ipv6 dhcp binding
```

- We have to create a pool, set-up prefix delegation and specify the DUID

```bash
(config)#ipv6 dhcp pool DHCP_CUSTOMERS
(config-dhcpv6)#prefix-delegation 2001:ffXX:ff02:AB00::/56 DUID
```
On C2

- To make sure the changes we made are propagated, we shut and no shut the interface towards R2

```
(config)# interface e0/0
(config-if) shut
(config-if) no shut
```

- It can take some time to propagate, up to 4 minutes

```
# show ipv6 interface brief
```
Security

13 - Section
IPv6 Security - Why Does It Matter?

- Most operating systems have IPv6 enabled by default nowadays
- IPv6 is present in your IPv4-only networks
  - tunnels
  - autoconfiguration on hosts
- The default IPv6 policies will not be what you need
- Often everything open
Subnet Scanning

• In IPv6, scanning the whole address space is not possible anymore, but people use:
  - words (dead, beef, babe, cafe)
  - lower numbers (::1, ::2, ::3)
Subnet Scanning

• Scanning multicast addresses
  - ff02::1 - all hosts
  - ff05::5 - all DHCP servers
  - ff05::2 - all routers

• You can use easy to remember addresses, but remember that scanning will work the same as in IPv4
ICMPv6

- ICMPv6 is used to report errors, ping and discover others (Neighbor Discovery)

- ICMPv6 is an integral part of IPv6

- Disabling ICMPv6 will break your network
Firewall Filtering and ICMPv6

- IPv6 border filter example

<table>
<thead>
<tr>
<th>Action</th>
<th>Src</th>
<th>Dst</th>
<th>ICMPv6 type</th>
<th>ICMPv6 Code</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permit</td>
<td>Any</td>
<td>A</td>
<td>128</td>
<td>0</td>
<td>echo reply</td>
</tr>
<tr>
<td>Permit</td>
<td>Any</td>
<td>A</td>
<td>129</td>
<td>0</td>
<td>echo request</td>
</tr>
<tr>
<td>Permit</td>
<td>Any</td>
<td>A</td>
<td>1</td>
<td>0</td>
<td>no route to dest</td>
</tr>
<tr>
<td>Permit</td>
<td>Any</td>
<td>A</td>
<td>2</td>
<td>0</td>
<td>packet too big</td>
</tr>
<tr>
<td>Permit</td>
<td>Any</td>
<td>A</td>
<td>3</td>
<td>0</td>
<td>TTL exceeded</td>
</tr>
<tr>
<td>Permit</td>
<td>Any</td>
<td>A</td>
<td>4</td>
<td>0</td>
<td>parameter problem</td>
</tr>
</tbody>
</table>
IPv6 Headers

• In IPv6, the header of a packet can be extended

• Extension headers are used for routing, fragmentation, IPSEC, etc.

• Some Intrusion Detection Systems find it hard to figure out where layer 4 starts and the extension header ends
IPSec

- IPSec in IPv6 is the same as in IPv4
- There is nothing automatically secure in IPv6
- IPSec should be supported in IPv6
  - PKI infrastructure costs time and money
RA Guard

- RFC6105
- Implement on a L2 switch, so they can filter out rogue or misconfigured routers sending router advertisements
- Filtering based on:
  - MAC address
  - Port where the RA was received
  - IP source address
Hosts

- Hosts can get an IPv6 address unnoticed
- Hosts can set up tunnels
- Keep software up-to-date
- Host security controls should inspect IPv4 and IPv6
  - Firewalls
  - VPN clients
Routers

• Protect vty lines

```plaintext
ipv6 access-list line-vty-in
   remark company management prefix
   permit ipv6 2001:db8:0:1::/64 any
```

```plaintext
line vty 0 15
ipv6 access-class line-vty-in in
```

• Use a /127 for point-to-point links if possible
IPv6 Bogons

• Documentation prefix
  - 2001:db8::/32

• 6bone
  - 3ffe::/16
  - Returned to the IANA pool

• Cymru bogon list (very long!)
  - Also available as BGP feed
  - https://www.team-cymru.org/Services/Bogons/fullbogons-ipv6.txt
Questions
Configuring NAT64

14 - Section
NAT64 / DNS64

• Single-stack clients will only have IPv6
• Translator box will strip all headers and replace them with IPv4
• Requires some DNS “magic”
  - Capture responses and replace A with AAAA
  - Response is crafted based on target IPv4 address
• Usually implies address sharing on IPv4
NAT64/DNS64

Mobile User
public IPv6

Infrastructure
public IPv6

NAT64 Box

DNS64

IPv6 Internet

IPv4 Internet

CUSTOMER

PROVIDER

INTERNET
Well Known Prefix

- **64:ff9b::/96**
  - Algorithmic translation from an IPv4 address to an IPv6 address and vice versa

<table>
<thead>
<tr>
<th>Well-Known Prefix</th>
<th>IPv4 address</th>
<th>IPv4-Embedded IPv6 address</th>
</tr>
</thead>
<tbody>
<tr>
<td>64:ff9b::/96</td>
<td>192.0.2.33</td>
<td>64:ff9b::192.0.2.33</td>
</tr>
</tbody>
</table>

- Can not be used for private IPv4 addresses
- Only used within the operator, not announced in global routing table
- Or use your own prefix, but be careful
NAT64 Lab

• The problem is: We will have IPv6-only customers that cannot reach IPv4-only content
  • Step 1: Configure C1 as an IPv6-only customer
  • Step 2: Ping from C1 to www.example.com
  • Step 3: Ping from C1 to ipv4.example.com
  • Step 4: Setup R4 as NAT64 box
  • Step 5: Set static route on R2
  • Step 6: Ping from customer to dns64.example.com
NAT64 Lab

CUSTOMER 1

CUSTOMER 2

R2

R1

R4

POP

IPv6 Internet

IPv4 Internet

DNS64
Step 1: Configure C1

- The customer will only have an IPv6 address
  
  ```
  ipv6 unicast-routing
  ipv6 cef
  ```

- Set up a static route to R2
  
  ```
  interface e0/0
  ipv6 address 2001:ffXX:0:ff01::a/64
  no ipv6 redirects
  ipv6 nd ra suppress all
  no shut
  ```

- Set up a name server
  
  ```
  ip domain lookup
  ip name-server 2001:ff53::53
  ```
Step 2: Ping from C1 to dual-stack

- Ping from the customer to a dual-stacked website

```
ping www.example.com
```
Step 3: Ping from C1 to IPv4-only

- Ping from the customer to an IPv4-only website

```
ping ipv4.example.com
```
Step 4: Setup R4 as NAT64 box

- Enable IPv6

```
ipv6 unicast-routing
ipv6 cef
```

- Configure the loopback interface for IPv4 & IPv6

```
interface lo0
  ipv6 address 2001:ffxx::4/128
  no ipv6 redirects
```
Step 4: Setup R4 as NAT64 box

- Configure the 3 internal interfaces

```
interface e0/0
  ipv6 address 2001:ffXX:0:04::a/127
  no ipv6 redirects
  ipv6 nd ra suppress all
```

```
interface e0/1
  ipv6 address 2001:ffXX:0:05::a/127
  no ipv6 redirects
  ipv6 nd ra suppress all
```

```
interface e1/0
  ipv6 address 2001:ffXX:0:06::a/127
  no ipv6 redirects
  ipv6 nd ra suppress all
```
Step 4: Setup R4 as NAT64 box

- Setup OSPFv3 on the router

```plaintext
ipv6 router ospf 1
    log-adjacency-changes
    router-id 172.X.255.Y  (Y is router number)
    passive-interface Ethernet2/0
    redistribute connected
```

- Setup OSPFv3 on the three interfaces: (e0/0, e0/1, e1/0)

```plaintext
interface xyz
    ipv6 ospf network point-to-point
    ipv6 ospf 1 area 0
    no shut
```
Step 4: Setup R4 as NAT64 box

- Configure the transit interface

  ```
  (config)# interface Ethernet2/0
  (config-if)# ip address 10.132.X.2 255.255.255.252
  (config-if)# no shut
  ```

- Test if transit provider router is reachable

  ```
  # ping 10.132.X.1
  ```
Step 4: Setup R4 as NAT64 box

• Create a filter

```
(config)# ip prefix-list transit-out-v4 seq 5 permit 10.X.0.0/22
```

• Configure the BGP session

```
(config)# router bgp 1XX
(config-router)# bgp log-neighbor-changes
(config-router)# neighbor 10.132.X.1 remote-as 44
(config-router)# neighbor 10.132.X.1 prefix-list transit-out-v4 out
```

• Set the network statement

```
(config-router)# network 10.X.0.0 mask 255.255.252.0
```
Step 4: Setup R4 as NAT64 box

• Insert static Null route
  
  - Before BGP advertised its network, it checks for an exact match of network number and mask on router’s routing table

  (config)# ip route 10.0.0.0 255.255.252.0 null0
Step 4: Setup R4 as NAT64 box

- From R4 ping the IPv4-only host behind the transit

  ping 193.0.21.80
enable NAT64 on the interfaces

- Interface lo0, e0/0, e0/1, e1/0 and e2/0

```bash
interface xyz
nat64 enable
```

- On the router define the prefix used for the translations

```bash
(config)# nat64 prefix stateful 64:ff9b::/96
```
Step 4: Setup R4 as NAT64 box

- Set up an access list

```
(config)# ipv6 access-list allow-nat64
(config-acl)# permit ipv6 2001:ffXX::/32 any
(config-acl)# exit
```
Step 4: Setup R4 as NAT64 box

- Define the pool of IPv4 addresses used for the translation

```
(config)# nat64 v4 pool nat64-v4-pool 10.X.3.0 10.X.3.255
```

- Map pool and access list

```
(config)#nat64 v6v4 list allow-nat64 pool nat64-v4-pool overload
```
Step 5: Setup a static route on R2

• We need a static route from R2 to R4

(config)# ipv6 route 64::ff9b::/96 2001::ffXX::4
ping dns64.example.com from C1

# ping dns64.example.com
Questions
IP Address Management

15 - Section
Why IP Address Management?

- How do you currently keep track?
  - There are many subnets in IPv6
  - Your spreadsheet might not scale
  - And you want to take care of DNS/reverse DNS

- There are 524288 /48s in a /29
- That is 34359738368 /64s!
Address Management

• There are many open source IPv6 IPAM tools
  - NetDot
  - GestióIP
  - phpIPAM
  - Netbox
  - NIPAP
  - NOCProject

• And of course our own IP Analyser
NetDot

- Device discovery via SNMP
- DNS and DHCP config management
- MAC address tracking
- BGP and AS Number tracking
- Export scripts for
  - Nagios
  - Smokeping
  - Cacti
  - RANCID
GestiólIP

- Web based IPAM software
- Structure based on Surfnet document
- Shows free ranges
- Incorporated VLAN management system
- Host discovery via SNMP and DNS
- Multi lingual (Russian, Italian, French, Catalan, etc.)
- DNS zone file generator for forward and reverse zones
  - Supporting BIND and tinydns zone files
<table>
<thead>
<tr>
<th>network</th>
<th>BM</th>
<th>description</th>
<th>site</th>
<th>category</th>
<th>comment</th>
<th>sync</th>
<th>vlan</th>
</tr>
</thead>
<tbody>
<tr>
<td>2001:db8:12ef::</td>
<td>52</td>
<td></td>
<td>HH1</td>
<td>prod</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2001:db8:12ef::</td>
<td>64</td>
<td>frontends I</td>
<td>HH1</td>
<td>prod</td>
<td>new frontends</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2001:db8:12ef::</td>
<td>64</td>
<td>backends</td>
<td>HH1</td>
<td>prod</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2001:db8:12ef::</td>
<td>64</td>
<td>sync LBs</td>
<td>HH1</td>
<td>prod</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2001:db8:12ef::</td>
<td>64</td>
<td>sync FWs</td>
<td>HH1</td>
<td>prod</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2001:db8:12ef::</td>
<td>64</td>
<td>management network devices</td>
<td>HH1</td>
<td>prod</td>
<td>400 - management pro</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2001:db8:12ef::</td>
<td>64</td>
<td>backup</td>
<td>HH1</td>
<td>prod</td>
<td>121 - backup pro</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2001:db8:12ef::</td>
<td>64</td>
<td>admins</td>
<td>HH1</td>
<td>corp</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2001:db8:12ef::</td>
<td>64</td>
<td>developers</td>
<td>HH1</td>
<td>corp</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2001:db8:12ef::</td>
<td>64</td>
<td>Virt</td>
<td>HH1</td>
<td>prod</td>
<td>Pool virtual addresses</td>
<td>125 - virt pro</td>
<td></td>
</tr>
<tr>
<td>2001:db8:12ef::</td>
<td>64</td>
<td></td>
<td>HH2</td>
<td>pre</td>
<td>601 - frontends pre</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2001:db8:12ef::</td>
<td>64</td>
<td>management pro</td>
<td>HH2</td>
<td>pre</td>
<td>602 - backends pre</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2001:db8:12ef::</td>
<td>64</td>
<td>backup</td>
<td>HH2</td>
<td>pre</td>
<td>607 - management pro</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2001:db8:12ef::</td>
<td>64</td>
<td></td>
<td>HH2</td>
<td>pre</td>
<td>610 - backup pre</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2001:db8:12ef::</td>
<td>64</td>
<td>virt pre</td>
<td>HH2</td>
<td>pre</td>
<td>688 - virt pre</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2001:db8:12ef::</td>
<td>64</td>
<td>frontends</td>
<td>HH2</td>
<td>dev</td>
<td>801 - frontends dev</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2001:db8:12ef::</td>
<td>64</td>
<td>backends</td>
<td>HH2</td>
<td>dev</td>
<td>902 - backends dev</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2001:db8:12ef::</td>
<td>64</td>
<td>admins</td>
<td>HH2</td>
<td>corp</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
phpIPAM

- AJAX based using jQuery libraries
  - PHP script, javascript and some HTML5/CSS3
  - Modern browser is preferred
- E-mail notifications
- Displays free ranges and numbers of clients
- Import and export to XLS files
- Can pull info from the RIPE DB
- Does not update DNS server
phpIPAM IP address management

Customer IPv6 segment (3a54:22:100::/48) has 4 directly nested subnets:

<table>
<thead>
<tr>
<th>VLAN</th>
<th>Subnet description</th>
<th>Subnet</th>
<th>Used</th>
<th>% Free</th>
<th>Requests</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>Free space</td>
<td>3a54:22:100:: - 3a54:22:100::1</td>
<td>0</td>
<td>184</td>
<td>477/407/737/035/1614</td>
</tr>
<tr>
<td>5</td>
<td>Customer 1</td>
<td>3a54:22:100:1::/64</td>
<td>0</td>
<td>184</td>
<td>477/407/737/035/1614</td>
</tr>
<tr>
<td>5</td>
<td>Customer 2</td>
<td>3a54:22:100:2::/64</td>
<td>0</td>
<td>184</td>
<td>477/407/737/035/1614</td>
</tr>
<tr>
<td>5</td>
<td>Customer 3</td>
<td>3a54:22:100:3::/64</td>
<td>0</td>
<td>184</td>
<td>477/407/737/035/1614</td>
</tr>
<tr>
<td>5</td>
<td>Customer 4</td>
<td>3a54:22:100:4::/64</td>
<td>0</td>
<td>184</td>
<td>477/407/737/035/1614</td>
</tr>
<tr>
<td>-</td>
<td>Free space</td>
<td>3a54:22:100:: - 3a54:22:100::1</td>
<td>0</td>
<td>184</td>
<td>477/407/737/035/1614</td>
</tr>
</tbody>
</table>

All IP addresses belonging to ALL nested subnets

<table>
<thead>
<tr>
<th>IP address</th>
<th>Hostname</th>
<th>Description</th>
<th>Device</th>
<th>Owner</th>
</tr>
</thead>
<tbody>
<tr>
<td>3a54:22:100:1</td>
<td>-</td>
<td>3a54:22:100:: - 3a54:22:100::1</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

phpIPAM IP address management [v0.67] rev008 | in case of problems please contact Miha Petkovic | Donate
Netbox

- Written in Python
- Web application with PostgreSQL database
- Also a Datacenter Infrastructure Management Tool
- No network monitoring, DNS, RADIUS, Config management
# Prefixes

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Status</th>
<th>VRF</th>
<th>Site</th>
<th>Role</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0.0.0/8</td>
<td>Active</td>
<td>Global</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>5.0.0.0/24</td>
<td>Active</td>
<td>Global</td>
<td>big site</td>
<td>Infrastructure</td>
<td>—</td>
</tr>
<tr>
<td>5.0.0.0/25</td>
<td>Active</td>
<td>Global</td>
<td>big site</td>
<td>VoIP</td>
<td>vop network</td>
</tr>
<tr>
<td>9.0.0.0/8</td>
<td>Active</td>
<td>Global</td>
<td>All</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>9.0.0.0/24</td>
<td>Active</td>
<td>Global</td>
<td>All</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>10.0.0.0/24</td>
<td>Container</td>
<td>Global</td>
<td>—</td>
<td>Infrastructure</td>
<td>Point-to-point links</td>
</tr>
<tr>
<td>10.0.0.0/31</td>
<td>Active</td>
<td>Global</td>
<td>Main Office</td>
<td>Infrastructure</td>
<td>Office MPLS</td>
</tr>
<tr>
<td>10.0.0.128/31</td>
<td>Active</td>
<td>Global</td>
<td>Branch 1</td>
<td>Infrastructure</td>
<td>Branch 1 MPLS circuit</td>
</tr>
<tr>
<td>10.0.0.130/31</td>
<td>Active</td>
<td>Global</td>
<td>Branch 2</td>
<td>Infrastructure</td>
<td>Branch 2 MPLS circuit</td>
</tr>
<tr>
<td>10.0.0.132/31</td>
<td>Active</td>
<td>Global</td>
<td>Branch 3</td>
<td>Infrastructure</td>
<td>Branch 3 MPLS circuit</td>
</tr>
<tr>
<td>10.0.0.134/31</td>
<td>Active</td>
<td>Global</td>
<td>Branch 4</td>
<td>Infrastructure</td>
<td>Branch 4 MPLS circuit</td>
</tr>
<tr>
<td>10.0.0.136/31</td>
<td>Active</td>
<td>Global</td>
<td>Branch 4</td>
<td>Infrastructure</td>
<td>Branch 5 MPLS circuit</td>
</tr>
<tr>
<td>10.0.0.138/31</td>
<td>Active</td>
<td>Global</td>
<td>Branch 1</td>
<td>Infrastructure</td>
<td>Backup MPLS link</td>
</tr>
<tr>
<td>10.0.100.0/24</td>
<td>Active</td>
<td>Global</td>
<td>London Data Center</td>
<td>Infrastructure</td>
<td>London Data Center - Server Network</td>
</tr>
<tr>
<td>10.1.0.0/16</td>
<td>Container</td>
<td>Global</td>
<td>Branch 1</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>10.1.0.0/24</td>
<td>Container</td>
<td>Global</td>
<td>Branch 1</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>10.1.0.0/25</td>
<td>Active</td>
<td>Global</td>
<td>Branch 1</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>10.1.0.0/26</td>
<td>Active</td>
<td>Global</td>
<td>Branch 1</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>
NIPAP

- Written in Python
- Web and CLI
- Integrated audit log
- IP request system
- XML-RPC middleware
  - Easy integration with other applications
<table>
<thead>
<tr>
<th>VRF</th>
<th>Prefix</th>
<th>Node</th>
<th>Order</th>
<th>Customer</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>94.142.240.0/21</td>
<td></td>
<td></td>
<td></td>
<td>Coliadus PA</td>
</tr>
<tr>
<td></td>
<td>94.142.240.0/24</td>
<td></td>
<td></td>
<td></td>
<td>EU Networks shared subnet</td>
</tr>
<tr>
<td></td>
<td>94.142.241.0/24</td>
<td></td>
<td></td>
<td></td>
<td>member subnets</td>
</tr>
<tr>
<td></td>
<td>94.142.242.0/24</td>
<td></td>
<td></td>
<td></td>
<td>member subnets</td>
</tr>
<tr>
<td></td>
<td>94.142.244.0/24</td>
<td></td>
<td></td>
<td></td>
<td>DCG Shared subnet</td>
</tr>
<tr>
<td></td>
<td>94.142.245.0/24</td>
<td></td>
<td></td>
<td></td>
<td>member subnets</td>
</tr>
<tr>
<td></td>
<td>94.142.246.0/24</td>
<td></td>
<td></td>
<td></td>
<td>Sileus</td>
</tr>
<tr>
<td></td>
<td>94.142.247.0/24</td>
<td></td>
<td></td>
<td></td>
<td>Coliadus infra</td>
</tr>
<tr>
<td></td>
<td>195.52.224.0/22</td>
<td></td>
<td></td>
<td></td>
<td>Coliadus PA from final #8</td>
</tr>
<tr>
<td></td>
<td>195.52.224.0/24</td>
<td></td>
<td></td>
<td></td>
<td>Member subnets</td>
</tr>
<tr>
<td></td>
<td>194.1.193.0/24</td>
<td></td>
<td></td>
<td></td>
<td>Pim van Peet</td>
</tr>
<tr>
<td></td>
<td>195.114.12.0/24</td>
<td></td>
<td>146</td>
<td></td>
<td>Meikldeb Aemans</td>
</tr>
<tr>
<td></td>
<td>2a02:898:0/32</td>
<td></td>
<td></td>
<td></td>
<td>Coliadus PA</td>
</tr>
<tr>
<td></td>
<td>2a02:898:0/64</td>
<td></td>
<td></td>
<td></td>
<td>Coliadus infrastructure</td>
</tr>
<tr>
<td></td>
<td>2a02:898:0/80/64</td>
<td></td>
<td></td>
<td></td>
<td>AMS-IX Out of band EU Networks</td>
</tr>
<tr>
<td></td>
<td>2a02:898:0/120/64</td>
<td></td>
<td></td>
<td></td>
<td>TransIP OOB</td>
</tr>
</tbody>
</table>
NOC Project (mostly CIS region)

- BSD licensed
- Complete OSS system
- Clean web interface
- DNS integration
- Reporting tools
- Quick view options (free space)
- Hierarchical user groups
- Large developer team
NOC Project
IP Analyser

- Available through the LIR Portal
- Get a visual insight into your RIPE Database objects
  - Hierarchical view of **used** address space
- Create new objects using an easy to use wizard
  - Interface seamlessly with the RIPE Database
  - Explain the different options well
  - Use sensible defaults
  - Delete redundant objects directly from the UI
IP Analyser

More specific inet6nums

<table>
<thead>
<tr>
<th>Inet6num</th>
<th>Status</th>
<th>Date</th>
<th>Size</th>
<th>AsgSize</th>
<th>Netname</th>
</tr>
</thead>
<tbody>
<tr>
<td>2a01:9e00:4000::/34</td>
<td>ALLOCATED_BY_LIR</td>
<td>03-02-2011</td>
<td>/34</td>
<td></td>
<td>UK-FAELIX-CUSTOMER</td>
</tr>
<tr>
<td>2a01:9e00:ac00::/38</td>
<td>ALLOCATED_BY_LIR</td>
<td>04-02-2011</td>
<td>/38</td>
<td></td>
<td>UK-FAELIX-TUNNEL</td>
</tr>
<tr>
<td>2a01:9e00:a217::/48</td>
<td>ALLOCATED_BY_LIR</td>
<td>03-02-2011</td>
<td>/48</td>
<td></td>
<td>UK-FAELIX-FAELIX</td>
</tr>
<tr>
<td>2a01:9e00:2ee3c800::/53</td>
<td>ALLOCATED_BY_LIR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2a01:9e00:4000::/53</td>
<td>ALLOCATED_BY_LIR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2a01:9e00:4000::/64</td>
<td>ALLOCATED_BY_LIR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2a01:9e00:7fff::/48</td>
<td>ALLOCATED_BY_LIR</td>
<td>23-06-2012</td>
<td>/48</td>
<td></td>
<td>UK-FAELIX-CROSSCONNECT</td>
</tr>
</tbody>
</table>
Tips and Tools

16 - Section
Graduate to the next level!

http://academy.ripe.net
Feedback!

https://www.ripe.net/training/advanced-ipv6/survey
Follow us!

@TrainingRIPENCC
Questions

Email address
@Twitter handle
The End!