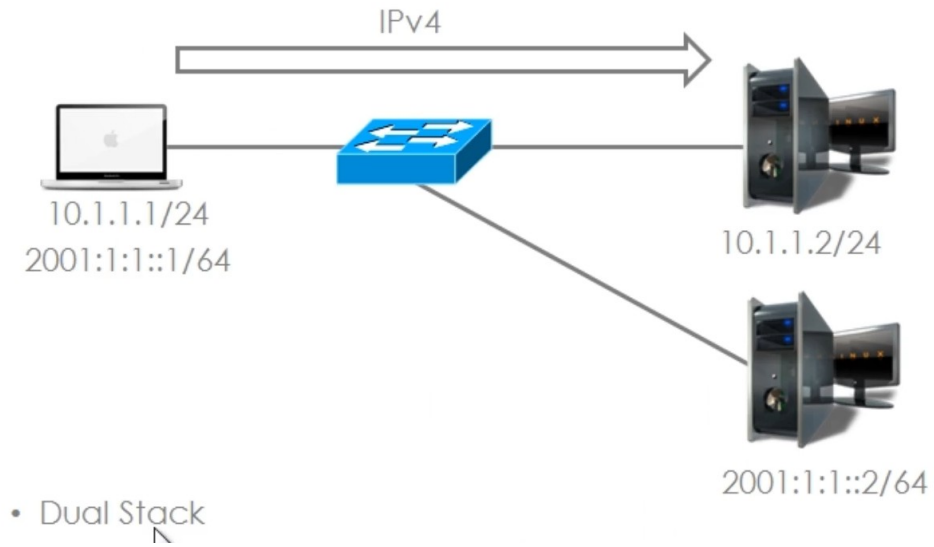
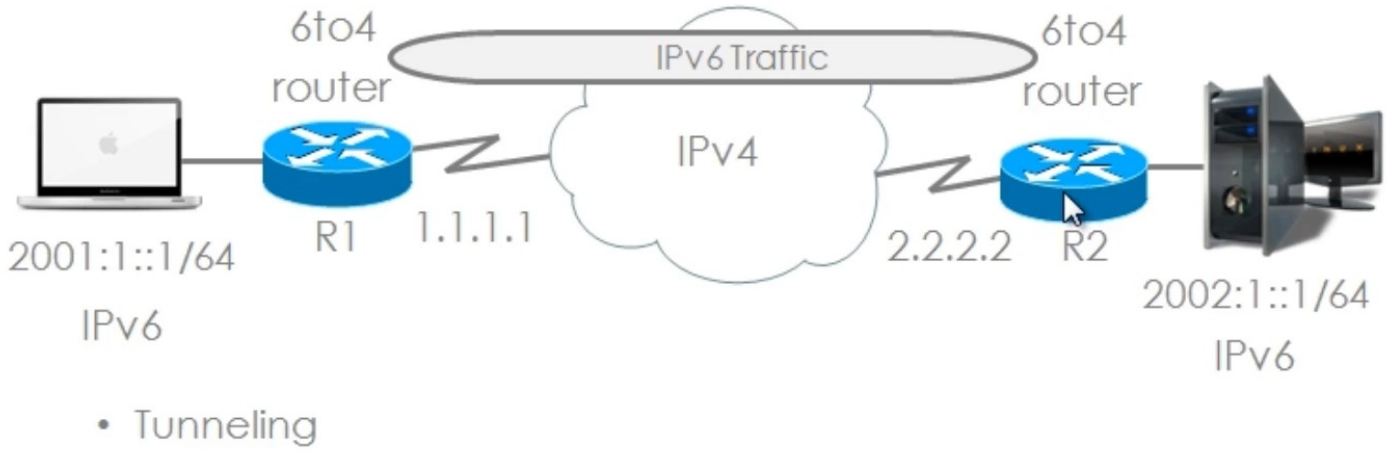
IPV4-to-IPv6 Transition Notes

DUAL Stack Method



Run both IPv4 and IPv6 at the same time on a single host
This allows the host to communicate over IPv4 to the server still running IPv4 and to Communicate over IPv6 to the server running IPv6



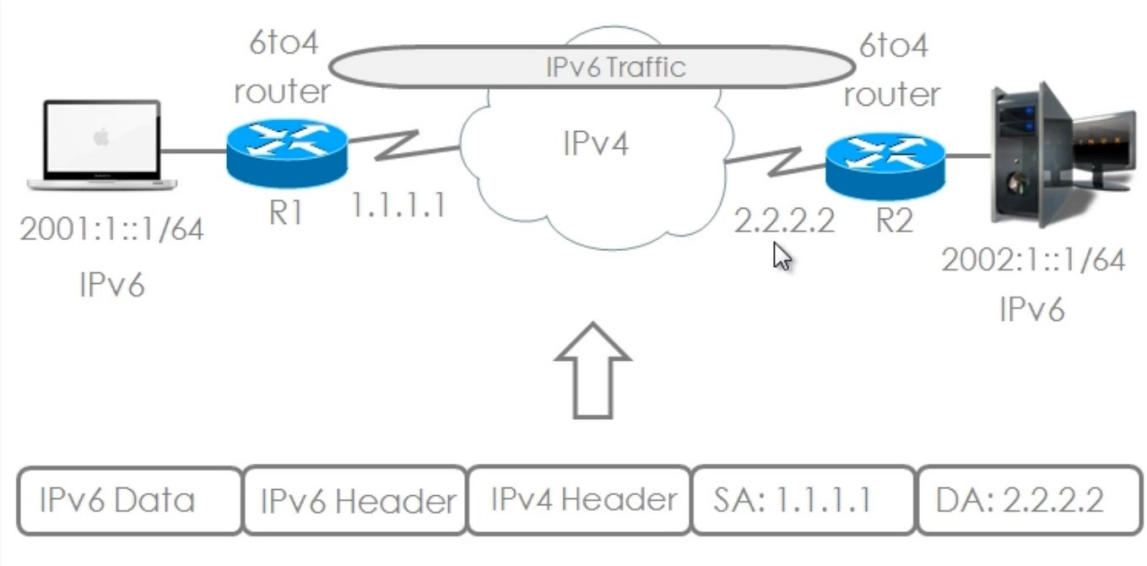
The Host on the left hand side is running IPv6 (only) and the Server to the Right is likewise running IPV6

However the Router's connecting them are only running IPv4, so IPv6 addresses will NOT be routed by this infrastructure.

So a Tunnel is setup between router 1 and Router 2 to "tunnel" IPv6 over IPv4.

There are multiple ways to setup this Tunnel

1. Manual IPv6 over IPv4 Tunnel
2. Dynamic 6to4 tunneling
3. Intra sire automatic tunnel addressing protocol (ISATAP)
4. Teredo Tunneling



The MacBook on the Left – sends IPv6 Data inside an IPv6 Header to its default gateway R1

R1 will then take the IPv6 information and encapsulate it inside an IPv4 Header

A tunnel is then setup from the local IPv4 address of R1 to the Destination IPv4 address for R2

Essentially an extra header (the IPv4 Header) is prepended to the existing IPV6 header, which encapsulates the IPV6 information/data.

The IPv4 infrastructure never sees the IPv6 header, they only see the IPv4 Header

When the Packet gets to Router 2, the IPv4 header is stripped off and the packet is sent on the remote LAN as a purely IPv6 packet.

When Setting up Tunneling

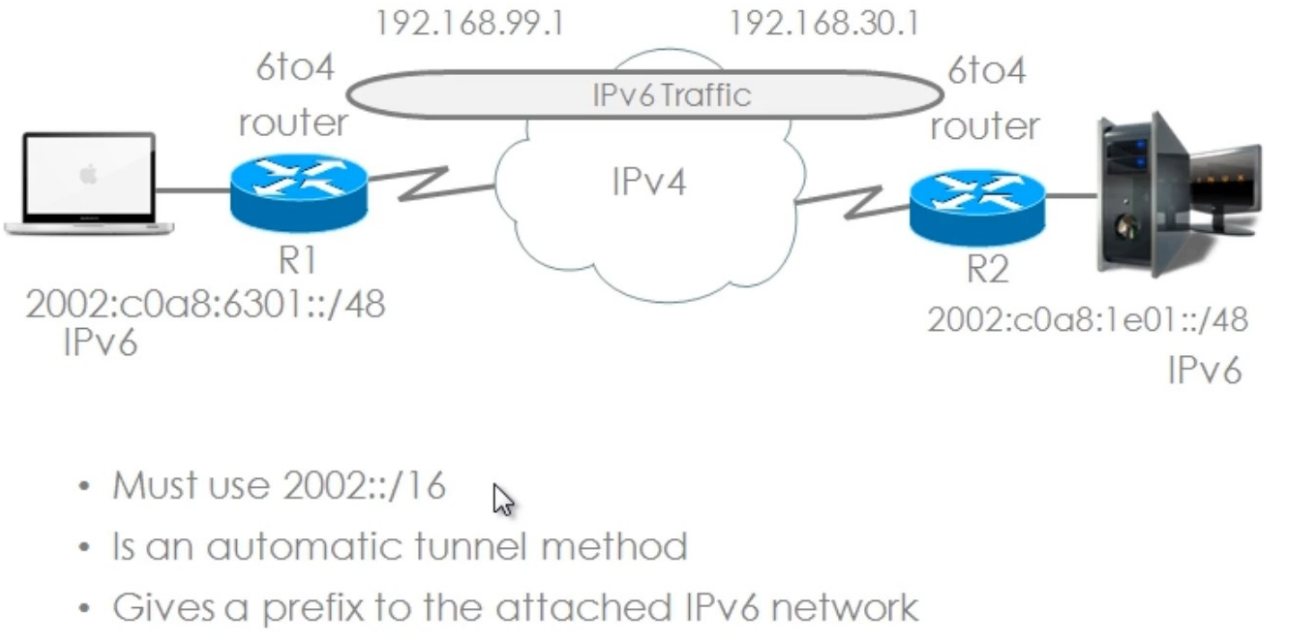
- Protocol Type 41 is specified in the IP header
- 20 byte IPv4 header with no options

The IPv6 header is encapsulated in IPv4 and when IPv4 encapsulates that IPv6 packet, a protocol type of “41” is specified in the IPv4 header.

Manual Tunneling

- You are manually establishing the tunnel between Router 1 and Router 2

Dynamic 6to4 Tunneling



Dynamic 6to4 Tunneling

- The Tunnel is automatically established between the IPv6 Networks through the IPv4 Network
- Presetting of Source and Destination IPv4 addresses is not required
- Automatic prefix assignment is where 1 Aggregateable global unicast IPv6 prefix is assigned to each 6to4 site
- These automatic prefix assignments are based on the specific address 2002::/16 (assigned by IANA)
- Must use the 2002::/16 address range
- Provides for an automatic tunnel
- Gives a prefix to the attached IPv6 network

2002::/16
The ipv4 address is converted to Hexadecimal and add to the 2002::/16 address space

2002 is 16 bits (remember each Hexadeimal is 4 bits)
Plus the 32 bits for the IPv4 address equals 48 bits, thus a /48 mask

192.168.99.1 is the IPv4 address for R1's interface that connects to the IPV4 infrastructure between the two IPv6 networks

192.168.99.1 in hex is c0.a8.63.01
All together the IPv6 address is
2002:C0A8:6301::/48

ISATAP - Intra Site Automatic Tunnel Addressing Protocol

An automatic overlay tunneling network mechanism that uses IPv4 as a link-layer for IPv6
These tunnels allow individual "IPv4/IPv6 dual stack hosts" within a site to communicate with other hosts on a virtual link, creating an IPv6 network using the IPv4 infrastructure
For example, this allows a host to setup a "dynamic IPv6 tunnel" to a Cisco Router across an IPv4 infrastructure

Teredo Tunneling

Allows for Host-to-Host Automatic Tunneling, instead of Gateway Tunneling
It can be used to pass "Unicast IPv6 Traffic" when "Dual Stack Hosts" are located behind one or multiple "IPv4 Network Address Translators"

IOS LAB – 6to4 Tunneling

R1 Se 0/0/0 and R2 Se 0/0/0 are IPv4 Only on the 10.1.2.0 /24 subnet

R1 Gig 0/0 is IPv6 network 2001:1:1:1::1/64
R2 Gig 0/0 is IPv6 network 2001:1:1:3::1/64

A 6to4 Tunnel over WAN (aka the Serial interfaces) needs to be created

Step 1: Create an Interface for Tunnel 0

R1(config)#int tun 0

Step 2: Define Tunnel Parameters

a) Ipv6 address
b) Tunnel Source
c) Tunnel Destination
d) Tunnel Mode

R1(config-if)#ipv6 address 2003::1/64
R1(config-if)#tunnel source 10.1.2.1

R1(config-if)#tunnel destination 10.1.2.2

R1(config-if)#tunnel mode ipv6ip

Step 3: Create a Static IPV6 Route pointing to the network on R2 you are tunneling to
R1(config)#ipv6 route 2001:1:1:3::64 tun 0
This essentially tells the router that the IPv6 route for 2001:1:1:3::/64 is available thru Tunnel 0 (on R1)

Do the same thing on R2

Step 1: Create an Interface for Tunnel 0

R2(config)#int tun 0

Step 2: Define Tunnel Parameters

a) Ipv6 address
b) Tunnel Source
c) Tunnel Destination
d) Tunnel Mode

R2(config-if)#ipv6 address 2003::2/64
R2(config-if)#tunnel source 10.1.2.2

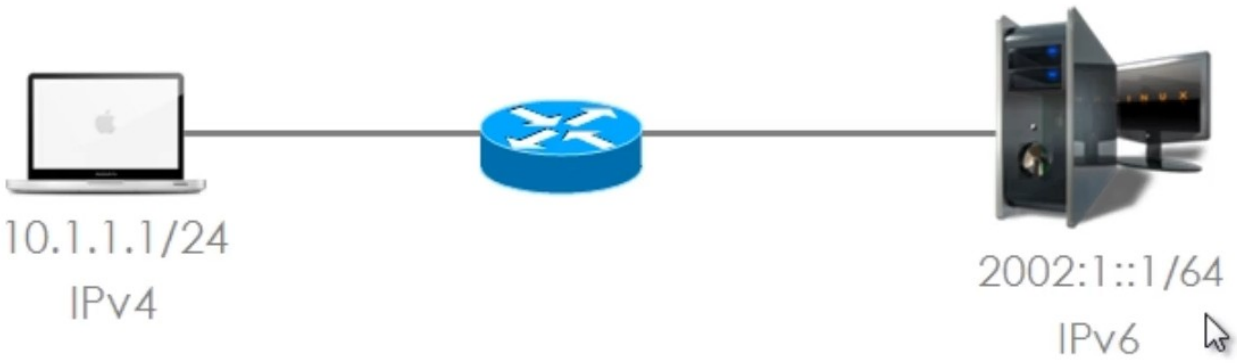
R2(config-if)#tunnel destination 10.1.2.1

R2(config-if)#tunnel mode ipv6ip (stands for IPv6 over IP encapsulation)

Step 3: Create a Static IPV6 Route pointing to the network on R1 you are tunneling to
R2(config)#ipv6 route 2001:1:1:1::64 tun 0
This essentially tells the router that the IPv6 route for 2001:1:1:1::/64 is available thru Tunnel 0 (on R2)

Within the IPv6 Routing Table on Router 1 you will now see a Route to 2001:1:1:3/64 via Tunnel 0
And on Router 2 you will now see a Route to 2001:1:1:1/64 via Tunnel 0

Proxying and Translation (NAT-PT) Network Address Translation – Protocol Translation



Allows for the translation of both IP Addressees and Protocols

The host on the left only communicates over IPv4 and the host on the right only communicates over IPv6

NAT-PT Example

R1 – only IPV6 interfaces Fa
R2 – Both IPv6 and Ipv4 interfaces
R3 - Only IPv4 interface F0/0 = 10.1.1.2 /24

On router 2
IPv6 NAT is configured on se0/0 and Fa0/0

The following commands were entered on R2 to set the IPv6 NAT parameters
ipv6 nat prefix 3FFE:B00:FFFF::1:0:0/96
ipv6 nat v4v6 source 10.1.1.2 3FFE:B00:FFFF::1:0:A
ipv6 nat v6v4 source 3FFE:B00:FFFF:1::2 10.1.1.3

ipv6 nat v4v6 source 10.1.1.2 3FFE:B00:FFFF::1:0:A
this command creates a static mapping this IPV4 address “10.1.1.2” to this IPV6 add “3FFE:B00:FFFF::1:0:A” maps a valid IPV4 address to a fake IPv6 address (“3FFE:B00:FFFF::1:0:A”)

ipv6 nat v6v4 source 3FFE:B00:FFFF:1::2 10.1.1.3
this command creates a static mapping this IPV6 add “3FFE:B00:FFFF:1::2” to this IPv4 Add “10.1.1.3” maps a valid IPv6 address (“3FFE:B00:FFFF:1::2” on R1) to a fake IPv4 Address – (“10.1.1.3” from the 10.1.1.0/24 subnet)

```
R2#sh ip nat translations

R2#sh ipv6 nat tr
R2#sh ipv6 nat translations
Prot  IPv4 source              IPv6 source
      IPv4 destination         IPv6 destination
---   ---                      ---
      10.1.1.2                 3FFE:B00:FFFF::1:0:A

---   10.1.1.3                 3FFE:B00:FFFF:1::2
      10.1.1.2                 3FFE:B00:FFFF::1:0:A

---   10.1.1.3                 3FFE:B00:FFFF:1::2
      ---                      ---
```