

# Tui

## Deployment Guide

Author: Nathan Ward  
Version: 0.6 - DRAFT  
Date: 5/06/2008



# Contents

---

Background	5
Peering vs. non-peering networks	5
IPv4 and IPv6	5
6to4	5
Teredo	6
Tui	7
Technical Overview	8
Route servers	8
Tui appliance	8
Route Server Deployment	10
Tui Node Deployment	12
Configuration basics	12
Building an image	12
IPv4 connectivity	14
IPv6 connectivity	19
Joining an existing IPv6 network	19
Creating a new IPv6 network	20
6to4 configuration	21
Teredo configuration	21
Connecting Tui to the Tui route servers	22
Routing Policy	22
Resources	23

---

## Document History

Version	Date	Author	Comments
0.1	17/12/2007	Nathan Ward	Initial document creation
0.2	28/01/2008	Nathan Ward	First draft released

Version	Date	Author	Comments
0.3	4/02/2008	Nathan Ward	Added CF card build instructions Added DNAT 6to4 information Added NTP and SNMP information
0.4	29/03/2008	Nathan Ward	Added APNIC support Added IPv4 /24s and ASNs
0.5	29/03/2008	Nathan Ward	Updated connectivity diagram to show ISPs transiting /24s
0.6	5/06/2008	Nathan Ward	Updated ISP v4 and v6 connectivity sections to show several v4 examples and simplify v6 examples

## Diagram Conventions



Teredo

Public Teredo or 6to4 Relay

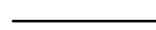


Traffic Flow



Tui

Tui Appliance



Network Connectivity

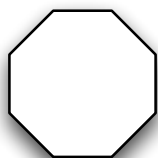


Tui RS

Tui Route Server



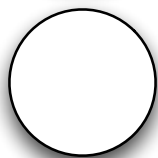
IPv4 only network



Internet Exchange



IPv4 and IPv6 network



Host



Non-free transit network

## Documentation Conventions

RFC3330 defines TEST-NET, 192.0.2.0/24, for documentation. APNIC defines IPV6-DOC-AP, 2001:0db8::/32 for documentation. Both these prefixes are used in this document as examples. Network operators should use their own IP numbers as appropriate.

Code, commands and variables are always in fixed-width fonts. Line breaks are signified by a backslash (\) at the end of the line.

# Background

---

## Peering vs. non-peering networks

We define two concepts for simplicity; Peering networks and non-peering networks. This is a technical document so does not make any political statements, it simply accepts that both exist.

A *peering network* is defined as a network that has its IPv6 prefixes advertised openly at the APE and/or WIX. Consequently, a *non-peering network* is defined as a network that does not have its prefixes advertised at the APE and/or WIX.

## IPv4 and IPv6

Much documentation already exists for IPv4 and IPv6, and some basic knowledge is assumed. The reader should understand IP version agnostic inter-networking concepts such as prefixes, BGP4+, next-hops, interfaces, tunnels, and so on.

This document does not make any political statement regarding IPv6, however Tui is designed to encourage its update in both New Zealand, and around the world.

## 6to4

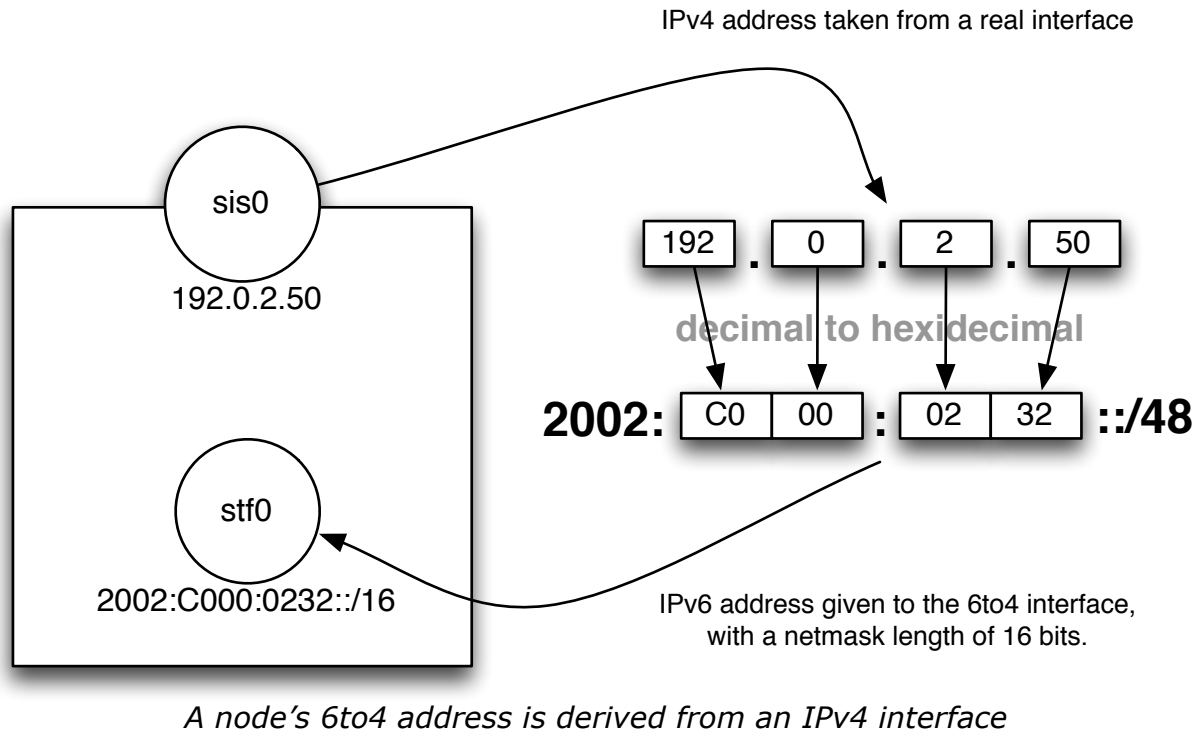
This tunneling protocol is described in RFC3056<sup>1</sup>. It gives an IPv6 /48 prefix to a node with a globally reachable IPv4 address. 6to4 encapsulates IPv6 traffic in IPv4 protocol 41. 6to4 addresses are all in the IPv6 prefix 2002::/16.

While 6to4 does not work behind NAT, if the NAT router is dual-stack and supports 6to4, hosts behind the router can use 6to4 addresses as though they had native IPv6 connectivity; the NAT router is configured to perform 6to4 encapsulation and decapsulation.

An interface is created on a node with an IPv6 address mapped from one of the node's IPv4 addresses. For example, a node with an IPv4 address 192.0.2.0 will have a 6to4 address in the prefix 2002:c000:0200::/48. The netmask length on the interface is 16 bits, and as such all addresses in the prefix 2002::/16 are considered to be connected. For the purposes of this document, a *6to4 node* is a node with a 6to4 interface.

---

<sup>1</sup> <http://www.ietf.org/rfc/rfc3056.txt>



6to4 is used in two ways:

1. When a 6to4 node attempts to send a packet to an address in the 6to4 prefix, it will take bits 17 through 48 and use these for the destination IPv4 address when encapsulating the packet.
2. When a 6to4 node attempts to send a packet to an address with a next-hop in the local routing table of a 6to4 address, bits 17 through 48 in the next-hop address are used for the destination IPv4 address.

Typically, a 6to4 relay is deployed with a well known IPv4 address of 192.88.99.1. 6to4 hosts wishing to access the global IPv6 network set their default IPv6 route to 2002:c058:6301:: - which tells the host to send all packets via the 6to4 relay. The relay router also advertises the 6to4 prefix to the IPv6 network. IPv6 hosts without a 6to4 interface use the relay router to reach 6to4 hosts. Both the 6to4 prefix and the IPv4 address are anycasted globally, so that 6to4 connected hosts can get reasonably efficient paths.

## Teredo

Teredo is defined in RFC4380<sup>2</sup>. It defines a method to allow an IPv4 node behind NAT to get a single IPv6 address, with information about how to encapsulate packets for transmission over IPv4 encoded in it. All Teredo addresses are in the prefix 2001:0000::/32

<sup>2</sup> <http://www.ietf.org/rfc/rfc4380.txt>

Other than end user *clients*, Teredo defines two main node types; Servers, and Relays. Teredo *servers* are used for initial interface setup and initiating communication when a node is behind a restricted NAT. Teredo relays are the heavy lifters, where all traffic in and out of the Teredo protocol traverses.

Teredo has a relay discovery system, which promotes the longer part of the path between a Teredo client and a non-Teredo host over IPv4, as opposed to over IPv6. The Teredo prefix (2001:0000::/32) is advertised to networks that the Teredo relay intends to service. This prefix is advertised to the IPv6 Internet if this is to be a public relay. Teredo clients detect the appropriate relay by sending an ICMP echo request and - in the case of cone NAT - watching for the source IPv4 address in the encapsulated response. Additional procedures exist for restricted NAT, documentation can be found in the RFC, and at Microsoft's Teredo technical description page.<sup>3</sup>

Teredo is implemented by Windows from XP Service Pack 2 upwards, including Vista. In Vista, Teredo is enabled by default under most configurations; Most notably, Windows Firewall must be enabled.

A GPL implementation of Teredo for Linux and BSD exists called "Miredo"<sup>4</sup>. Tui makes use of this.

Currently Teredo has no mechanism to discover an appropriate server based on a client's Internet connection. The default setting for Windows is 'teredo.ipv6.microsoft.com', which can only be changed by CLI commands or in the registry.

## Tui

The Tui project is run by Braintrust, and has received funding from InternetNZ to purchase initial hardware. Citylink have provided hardware for route-servers, and operational support. APNIC have supported the project by providing AS numbers and IPv4 numbers. The code, compiled images, and documentation are freely available, and the first 6 units are given away freely to NZ ISPs.

---

<sup>3</sup> <http://www.microsoft.com/technet/network/ipv6/teredo.mspix>

<sup>4</sup> <http://www.remlab.net/miredo/intro.shtml.en>

# Technical Overview

---

## Route servers

Two Tui route servers exist - one at each of APE and WIX. These devices exchange routes between Tui appliances, and to their respective IPv6 exchanges.

There is routing policy in place to allow Tui appliances in peering networks to tunnel via the route server to reach the local native IPv6 exchange. Non-peering networks may only send traffic to other Tui networks - this ensures that IPv4 peering and transit relationships still apply for all traffic between Tui enabled networks.

Community tags are applied to all routes, so that network operators can select whether to use specific types of Tui routes, based on their needs.

## Tui appliance

A Tui appliance (Tui) is intended to be installed in networks with large numbers of IPv4 or IPv6 connected end users, or with servers with IPv6 enabled content.

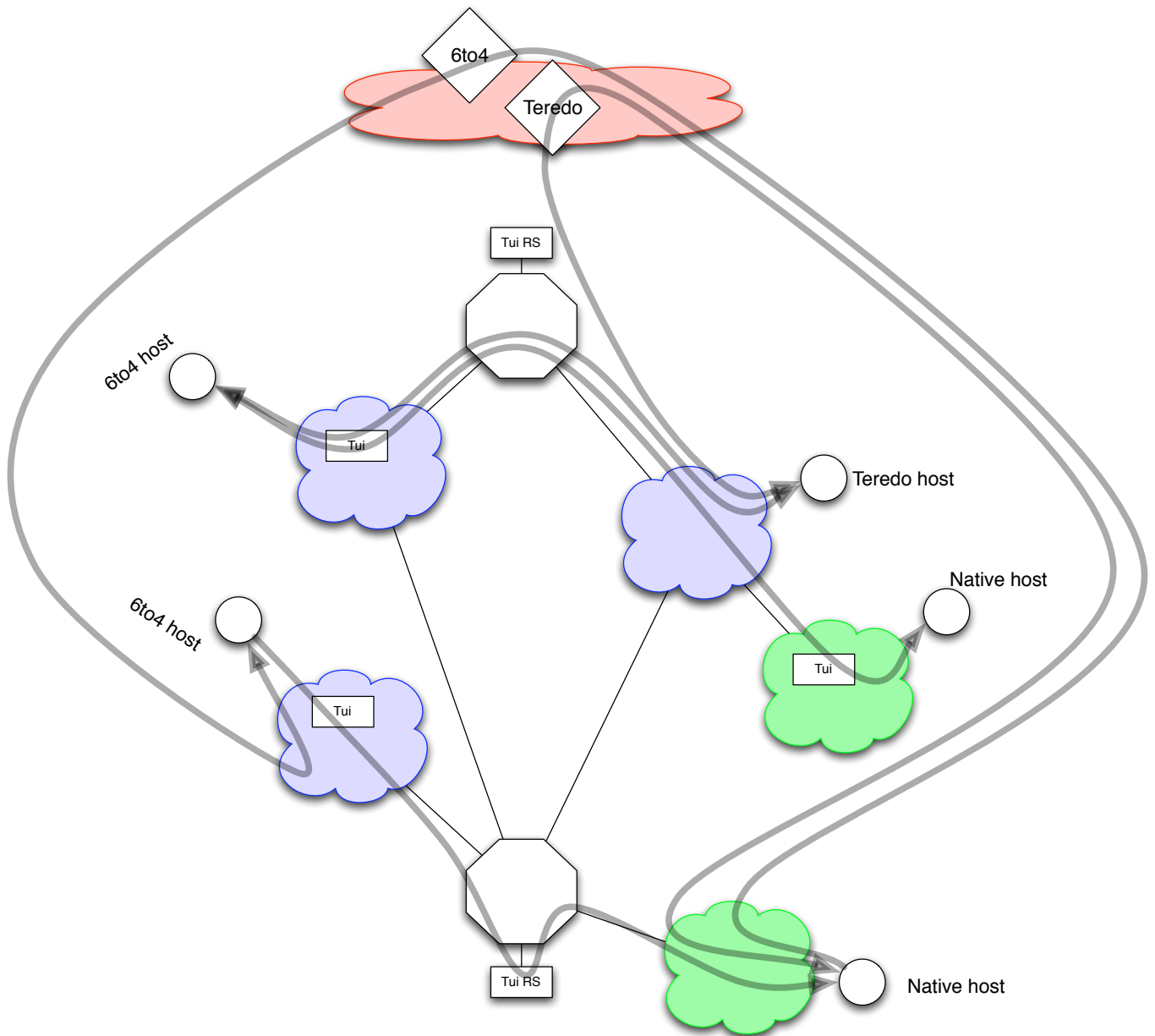
It provides 6to4 and Teredo relays, which are available to hosts on the network - ie ISP customers or IPv6 enabled servers. Also, zero configuration full mesh tunneled IPv6 connectivity is provided between Tui enabled networks over IPv4 best paths, so as not to disturb existing peering and transit arrangements between providers, and to provide efficient connectivity. Peering networks are able to access the IPv6 exchanges by tunneling via the route-servers, as part of that zero configuration mesh.

Tuis are Soekris NET4801 units, with a FreeBSD based software image. Included additional to FreeBSD are Quagga (for BGP4+) and Miredo (for Teredo).

Tuis require IPv6 connectivity to the IPv6 Internet, by way of either a tunnel that terminates directly on the unit, a tunnel that terminates elsewhere, or native IPv6 transit.

Typically, Tui uses the local network's ASN, and participates in iBGP.

Tui is useful for any network - even networks with existing native IPv6 deployed. See the following diagram, showing some potential flows. Teredo and 6to4 relays are important even for native IPv6 networks, if they are wishing to provide efficient paths to hosts that receive their IPv6 connectivity by way of Teredo or 6to4. Native IPv6 networks will also benefit from connectivity to other native networks, over IPv4 best paths, instead relaying through a series of statically configured tunnels.



*Potential flows and impact for networks with and without Tui*

# Route Server Deployment

---

The route servers have the following details:

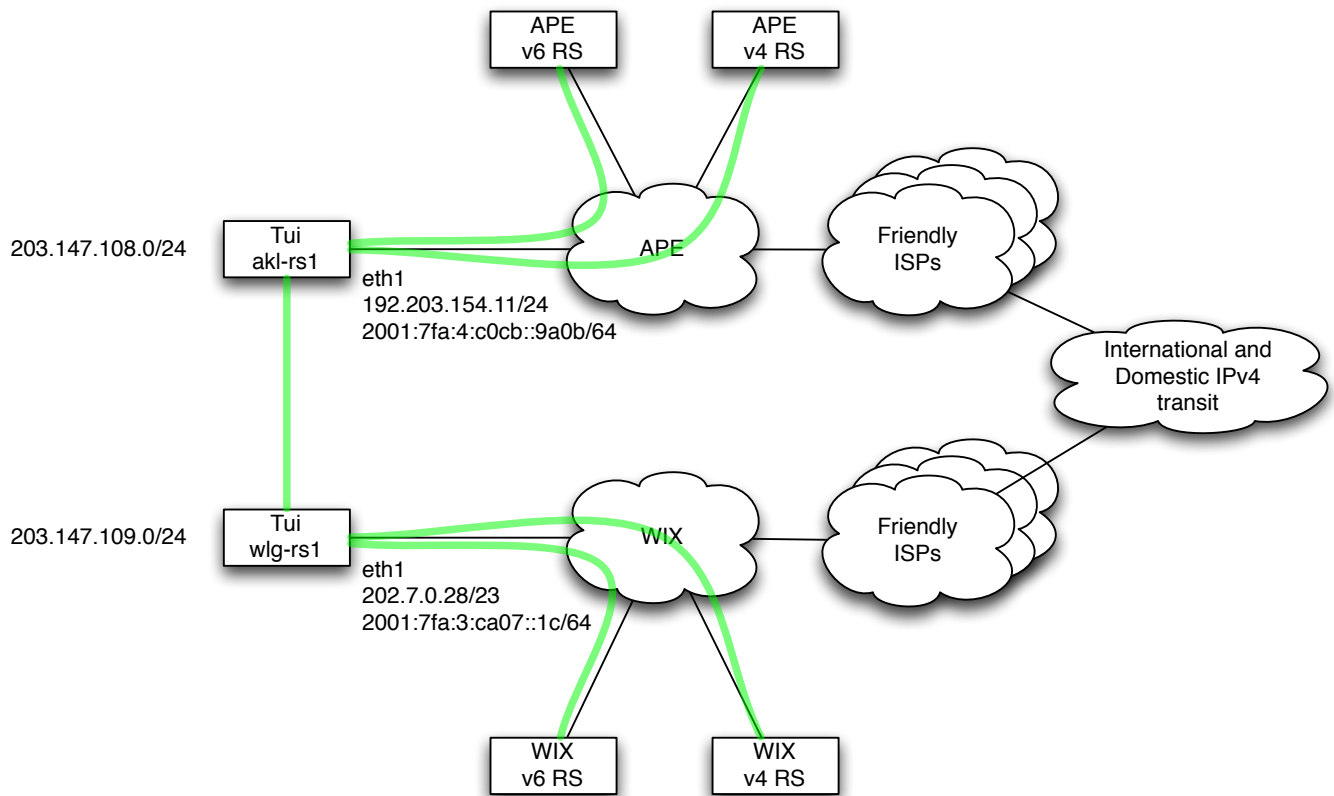
Purpose	RS1.AKL1	RS1.WLG1
Exchange IPv6 address	2001:7fa:4:cocb::9a0b	2001:7fa:3:ca07::1c
Exchange IPv4 address	192.203.154.11	202.7.0.28
Globally reachable IPv4 address	203.147.108.1 (/24)	203.147.109.1 (/24)
6to4 address	2002:cb93:6c01::	2002:cb93:6d01::
AS Number	24021	45163

eBGP sessions are established with the APE and WIX IPv4 route servers. All prefixes are accepted, and the globally reachable IPv4 prefix is advertised. Exchange IPv4 peers are encouraged to accept and transit this /24. The traffic on this prefix is very low, and by having many networks transit the /24 we gain high reliability.

eBGP sessions are established with the APE and WIX IPv6 route servers, and IPv6 prefixes for Tui hosts in Peering networks are advertised, with a next hop address of the Tui appliance's exchange IPv6 address. All prefixes are accepted.

eBGP sessions are established with each Tui appliance, using 6to4 addresses for end points of the eBGP sessions. All prefixes advertised by Tui appliances are passed on, with their next-hops unmodified. Peering networks receive advertisements for all of the local exchange's IPv6 prefixes, with the next hop set to the route-servers 6to4 address. Multi-hop is not required as the 6to4 cloud is a single hop.

## Tui - Deployment Guide



*Connectivity of the Tui route-servers in Auckland and Wellington*

# Tui Node Deployment

---

This section describes deployment of a Tui node.

## Configuration basics

Tui requires extensive use of SSH. Recommended clients are OpenSSH, included with most Linuxes, BSDs, and with OS X; or Putty, freely available for Windows hosts.

Tui is currently only usable as the root user, who's default password is NadsEnhylf. The password can be changed with the passwd command.

Tui's interface configuration and initialisation commands are stored in /flash/rc. This is called from /etc/rc at the end of the boot process. This file can be edited with vi, or uploaded using scp. When configuration is changed in /flash/rc, Tui must either be restarted, or the incremental changes performed manually.

Tui's routing process (Quagga, used for BGP and OSPF) configuration is performed by running vtysh. Configuration is very similar to Cisco.

Tui does not use Quagga for configuring interfaces, due to some incompatibilities in the way 6to4 interfaces are defined.

NTP configuration is kept in /config/ntp.conf. There are two NTP servers configured by default.

SNMP configuration is kept in /config/snmpd.config. The default read community is alwiktuf.

## Building an image

Tui compact flash cards have the following file structure:

```
cf:/
cf:/grub
cf:/grub/stage2
cf:/grub/stage1
cf:/grub/fat_stage1_5
cf:/grub/menu.lst
cf:/grub/default
cf:/config.img
cf:/rc
cf:/packages
cf:/pf.conf
cf:/net4501-kernel-<version>
```

```
cf:/net4801-kernel-<version>
```

1. Download the latest cfc card.tgz, kernel and packages from the Tui website

2. Format the CF card as FAT:

```
newfs_msdos /dev/da1s1
```

3. Mount the CF card

```
mkdir /mnt/flash  
mount_msdosfs /dev/da1s1 /mnt/flash
```

4. Extract the cfc card.tgz contents to the CF card

```
tar -xzf cfc card-version.tgz -C /mnt/flash
```

5. Copy the kernel to the CF card's root

6. Copy the packages to the CF card's packages directory

7. Edit grub/menu.lst, and update the kernel file names

8. Edit rc and configure the interfaces

9. Create a grub map file, called grub.map on your local machine with the following contents:

```
(hd0) /dev/da1
```

10. Install grub on the CF card:

```
# sysctl kern.geom.debugflags=16  
# grub --device-map=grub.map  
grub> root (hd0,0)  
Filesystem type is fat, partition type 0xb  
  
grub> setup (hd0)  
Checking if "/boot/grub/stage1" exists... no  
Checking if "/grub/stage1" exists... yes  
Checking if "/grub/stage2" exists... yes  
Checking if "/grub/fat_stage1_5" exists... yes  
Running "embed /grub/fat_stage1_5 (hd0)"... 17 sectors are  
embedded.  
succeeded  
Running "install /grub/stage1 (hd0) (hd0)1+17 p (hd0,0)/grub/  
stage2 /grub/menu.lst"... succeeded  
Done.  
  
grub> quit
```

11. Done

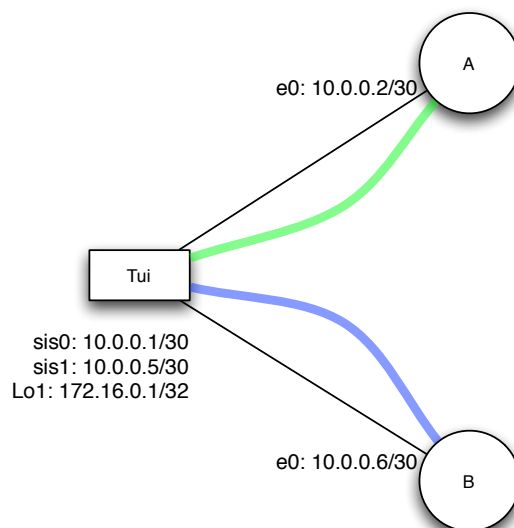
## IPv4 connectivity

Tui requires IPv4 Internet connectivity, on at least one ethernet interface. Over this, Tui essentially requires a single globally reachable IPv4 /32, and routes to the public Internet. These examples describe several ways to do this in an iBGP network, and due to the infinite number of possible configurations, this is not an exhaustive list of scenarios.

For these examples we use 10.0.0.0/8 as being available for use as link nets in your AS, and 172.16.0.1/32 as the Tui loopback address assigned from your address space.

### Two ISP routers each with a separate circuit to Tui

This describes connectivity with two ISP routers, each with its own circuit to the Tui node. Each circuit could be a VLAN across a switching infrastructure. Tui does not support VLAN tagging, so a separate switch port and cable would be needed for each router Tui interconnects with.



*Two ISP routers each with a separate circuit to Tui*

Tui's interface configuration is stored in /flash/rc. The IP networks used for sis0 and sis1 are used only for iBGP. If your AS has a policy of using private addresses for link nets, this policy is acceptable for Tui. Interfaces are configured with the following lines:

```
ifconfig sis0 10.0.0.1/30
ifconfig sis1 10.0.0.5/30
```

Tui requires a loopback interface in your IPv4 space.

```
ifconfig lo1 create
ifconfig lo1 172.16.0.1/32
```

Tui is designed to join your iBGP network. The following is the configuration for this on the Tui side.

```
router bgp <asn>
  no bgp default ipv4-unicast
  !
  address-family ipv4
  neighbor isp-connection-v4 peer-group
  neighbor isp-connection-v4 activate
  neighbor isp-connection-v4 prefix-list v4-loopback out
  neighbor isp-connection-v4 remote-as <asn>
  neighbor 10.0.0.2 description ISP Router A
  neighbor 10.0.0.2 peer-group isp-connection-v4
  neighbor 10.0.0.5 description ISP Router B
  neighbor 10.0.0.5 peer-group isp-connection-v4
  exit-address-family
  !
  ip prefix-list v4-loopback description Globally reachable IPv4
  loopback
  ip prefix-list v4-loopback seq 10 permit 172.16.0.1/32
```

Notice that we do not configure OSPF, or set up iBGP between the loopback addresses. This is to simplify deployment. OSPF is available in Quagga if desired, and then iBGP can be configured in a full mesh, or with your route-servers, or whatever is appropriate for your architecture.

Assuming OSPF and loopbacks are not used, we set up iBGP between the link local addresses. Additionally, in order to avoid having to join your existing full iBGP mesh, Tui is configured as a route-reflector client on the routers with which it peers.

### ISP Router A:

```
ISP Router A#
router bgp <asn>
  address-family ipv4
  neighbor 10.0.0.1 remote-as <asn>
  neighbor 10.0.0.1 route-reflector-client
  neighbor 10.0.0.1 prefix-list tui-in in
  exit-address-family
  !
  ip prefix-list tui-v4-loopback description IPv4 prefixes to accept
  from Tui
  ip prefix-list tui-v4-loopback seq 10 permit 172.16.0.1/32
  !
```

### ISP Router B:

```
ISP Router A#
router bgp <asn>
  address-family ipv4
  neighbor 10.0.0.5 remote-as <asn>
```

## Tui - Deployment Guide

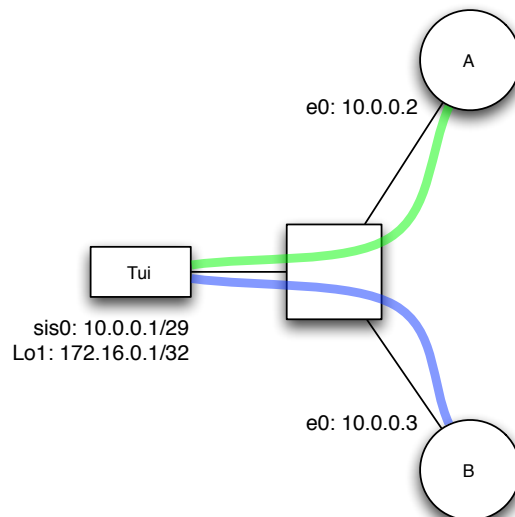
```
neighbor 10.0.0.5 route-reflector-client
neighbor 10.0.0.5 prefix-list tui-in in
exit-address-family
!
ip prefix-list tui-v4-loopback description IPv4 prefixes to accept
from Tui
ip prefix-list tui-v4-loopback seq 10 permit 172.16.0.1/32
!
```

At this point, BGP should come up, and Tui should be able to reach the Tui route-servers, and the global Internet from it's loopback address. We test this with ping, using -S to specify the source address to Tui's loopback address. ctrl-c can be used to stop ping:

```
# ping -S 172.16.0.1 60.234.76.2
PING 60.234.76.2 (60.234.76.2) from 172.16.0.1: 56 data bytes
64 bytes from 172.16.0.1: icmp_seq=0 ttl=59 time=48.623 ms ^C
#
```

## Two ISP routers with a shared link net with Tui

This describes connectivity with two ISP routers where they share a common VLAN to connect to Tui. A network shorter than /30 (ie. /29) is used for link net addressing. For brevity, we list example configurations only.



*Two ISP routers with a shared link net*

/flash/rc:

```
ifconfig sis0 10.0.0.1/30
ifconfig lo1 create
ifconfig lo1 172.16.0.1/32
```

Tui iBGP:

## Tui - Deployment Guide

```
router bgp <asn>
  no bgp default ipv4-unicast
  !
  address-family ipv4
  neighbor isp-connection-v4 peer-group
  neighbor isp-connection-v4 activate
  neighbor isp-connection-v4 prefix-list v4-loopback out
  neighbor isp-connection-v4 remote-as <asn>
  neighbor 10.0.0.2 description ISP Router A
  neighbor 10.0.0.2 peer-group isp-connection-v4
  neighbor 10.0.0.3 description ISP Router B
  neighbor 10.0.0.3 peer-group isp-connection-v4
  exit-address-family
!
ip prefix-list v4-loopback description Globally reachable IPv4
loopback
ip prefix-list v4-loopback seq 10 permit 172.16.0.1/32
```

### ISP Router A:

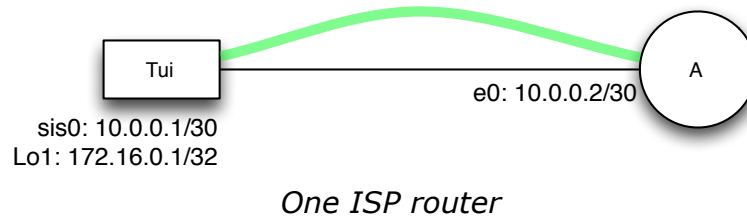
```
ISP Router A#
router bgp <asn>
  address-family ipv4
  neighbor 10.0.0.1 remote-as <asn>
  neighbor 10.0.0.1 route-reflector-client
  neighbor 10.0.0.1 prefix-list tui-in in
  exit-address-family
!
ip prefix-list tui-v4-loopback description IPv4 prefixes to accept
from Tui
ip prefix-list tui-v4-loopback seq 10 permit 172.16.0.1/32
!
```

### ISP Router B:

```
ISP Router A#
router bgp <asn>
  address-family ipv4
  neighbor 10.0.0.1 remote-as <asn>
  neighbor 10.0.0.1 route-reflector-client
  neighbor 10.0.0.1 prefix-list tui-in in
  exit-address-family
!
ip prefix-list tui-v4-loopback description IPv4 prefixes to accept
from Tui
ip prefix-list tui-v4-loopback seq 10 permit 172.16.0.1/32
!
```

## One ISP router

This describes connectivity with a single ISP router. For brevity, we list example configurations only.



/flash/rc:

```
ifconfig sis0 10.0.0.1/30
ifconfig lo1 create
ifconfig lo1 172.16.0.1/32
```

Tui iBGP:

```
router bgp <asn>
  no bgp default ipv4-unicast
  !
  address-family ipv4
  neighbor isp-connection-v4 peer-group
  neighbor isp-connection-v4 activate
  neighbor isp-connection-v4 prefix-list v4-loopback out
  neighbor isp-connection-v4 remote-as <asn>
  neighbor 10.0.0.2 description ISP Router A
  neighbor 10.0.0.2 peer-group isp-connection-v4
  exit-address-family
  !
  ip prefix-list v4-loopback description Globally reachable IPv4
  loopback
  ip prefix-list v4-loopback seq 10 permit 172.16.0.1/32
```

ISP Router A:

```
ISP Router A#
router bgp <asn>
  address-family ipv4
  neighbor 10.0.0.1 remote-as <asn>
  neighbor 10.0.0.1 route-reflector-client
  neighbor 10.0.0.1 prefix-list tui-in in
  exit-address-family
  !
  ip prefix-list tui-v4-loopback description IPv4 prefixes to accept
  from Tui
```

```
ip prefix-list tui-v4-loopback seq 10 permit 172.16.0.1/32
!
```

## IPv6 connectivity

Tui provides connectivity to other Tui users and peers on the IPv6 peering exchanges. The tui network does not provide connectivity to the international IPv6 network. As Tui acts as a 6to4 relay, it needs connectivity to the global IPv6 network.

## Joining an existing IPv6 network

If you already have an IPv6 network with connectivity to the global IPv6 network, Tui simply needs to be configured to join this network.

This method is equally appropriate if you are receiving a tunnel from a third party for IPv6 transit, or if you have native IPv6 transit.

If you have an existing IPv6 network but are not able to connect to it directly from Tui, please see the section below 'Creating a new IPv6 network' for information about configuring Tui for static IPv6 over IPv4 tunneling, then see the documentation below about configuring iBGP.

IPv6 interface and iBGP configuration is almost identical to IPv4, so this section is intentionally brief.

/flash/rc:

```
ifconfig sis0 inet6 2001:0db8:1234::0:1/112
ifconfig sis1 inet6 2001:0db8:1234::1:1/112
```

Tui should also be configured for iBGP to your existing network:

```
tui#
router bgp <asn>
  no bgp default ipv4-unicast
  !
  address-family ipv6
  neighbor isp-connection-v6 peer-group
  neighbor isp-connection-v6 activate
  neighbor isp-connection-v6 remote-as <asn>
  neighbor 2001:0db8:1234::0:2 peer-group isp-connection-v6
  neighbor 2001:0db8:1234::1:2 peer-group isp-connection-v6
  exit-address-family
!
```

As with IPv4, OSPF (OSPFv3) is optionally available, and iBGP sessions should be configured to treat Tui as a route-reflector-client to avoid building a full mesh with your existing network. This is discussed in the IPv4 connectivity section above.

## Creating a new IPv6 network

If your network infrastructure does not support IPv6, or if you are not ready to roll out IPv6 to your network, Tui can function as your endpoint for an IPv6 tunnel from your provider. GRE and 6in4 are both supported.

Interface configuration differs for GRE and 6in4, both are shown below, and should be included in /flash/rc:

```
ifconfig gre0 create
ifconfig gre0 inet6 2001:0db8:1234::2:1/112
ifconfig gre0 tunnel 192.0.2.255 <remote tunnel endpoint>
ifconfig gre0 up
```

```
ifconfig gif0 create
ifconfig gif0 inet6 2001:0db8:1234::2:1 2001:0db8:1234::2:2 \
  prefixlen 128
ifconfig gif0 tunnel 192.0.2.255 <remote tunnel endpoint>
ifconfig gif0 up
```

Regardless of the tunneling mechanism used, it is likely that eBGP is required over this link. Simple eBGP configuration follows:

```
tui#
router bgp <asn>
  no bgp default ipv4-unicast
  !
  address-family ipv6
    network 2001:0db8::/32
    neighbor 2001:0db8:1234::2:2 remote-as <remote_asn>
    neighbor 2001:0db8:1234::2:2 prefix-list our-v6-prefixes out
  exit-address-family
  !
  ipv6 prefix-list our-v6-prefixes description Our IPv6 prefixes
  ipv6 prefix-list our-v6-prefixes seq 10 permit 2001:0db8::/32
  !
```

This instructs Tui to advertise your IPv6 prefixes - and only your IPv6 prefixes - to your IPv6 tunnel provider. At this point, your IPv6 prefix should be globally visible.

## 6to4 configuration

Tui provides a 6to4 relay, and uses 6to4 for communication between other Tuis, and the Tui route-servers. Configuring 6to4 involves two things; configuring the 6to4 interface (stf0) and advertising the 6to4 prefixes with iBGP. Note that your iBGP peers must be configured to accept these prefixes. If the configuration in this document was used, this simply means amending the IPv4 prefix list tui-in to include 192.88.99.0/24.

The following configures DNAT to allow 6to4 relaying. Add this to /flash/pf.conf:

```
rdr inet proto ipv6 from any to 192.88.99.1 -> 192.0.2.255
```

The following configures the 6to4 interface, and should be added to /flash/rc:

```
ifconfig lo2 create
ifconfig lo2 192.88.99.1/24
ifconfig stf0 create
ifconfig stf0 inet6 2002:c000:02ff:: prefixlen 16
pfctl -ef /flash/pf.conf
```

The following configures iBGP to advertise the 6to4 prefixes:

```
tui#
router bgp <asn>
  address-family ipv4
  network 192.88.99.0/24
  exit-address-family
  !
  address-family ipv6
  network 2002::/16
  exit-address-family
  !
```

## Teredo configuration

Miredo runs in relay mode by default on Tui, so no configuration of the daemon is required.

To enable Tui to route Teredo traffic for your non-Teredo IPv6 hosts, simply advertise 2001::/32 to your network:

```
tui#
router bgp <asn>
  address-family ipv6
  network 2001::/32
  exit-address-family
  !
```

## Connecting Tui to the Tui route servers

Once IPv4 connectivity is established, Tui can be connected to the Tui route-servers in Auckland and Wellington. This is as simple as an eBGP session for each, with a custom route-map to set the next hop on prefixes we advertise.

```
router bgp <asn>
  address-family ipv6
  neighbor tui-route-server peer-group
  neighbor tui-route-server prefix-list our-v6-prefixes out
  neighbor tui-route-server route-map tui-rs-out out
  neighbor 2002:cb93:6c01:: peer-group tui-route-server
  neighbor 2002:cb93:6c01:: remote-as 4021
  neighbor 2002:cb93:6d01:: peer-group tui-route-server
  neighbor 2002:cb93:6d01:: remote-as 45163
  exit-address-family
!
route-map tui-rs-out permit 10
  set ipv6 next-hop 2002:c000:0200::
!
```

## Routing Policy

Determining preference of Tui-learned IPv6 routes and routes learned via other means (ie. transit, static tunnels, or native peering) is likely to depend on an individual network, however, some basic ideas are suggested:

- Tui should have a lower preference than routes learned via native peering
- Tui should have a higher preference than routes learned via international transit
- Tui communities are added to all routes that are re-advertised by the route-servers, which can be used to prioritise certain classes of routes in different ways. These communities are available from the Tui website.

# Resources

---

Further resources:

- <http://www.braintrust.co.nz/tui/> - The official Tui website
- <http://www.quagga.net/> - Quagga software suite, used as the BGP and OSPF routing daemon
- <http://www.remlab.net/miredo/> - Miredo, Tui's Teredo implementation
- <http://www.freebsd.org/> - FreeBSD, the basis for Tui
- <http://www.citylink.co.nz/> - Citylink run the NZ peering exchanges
- <http://www.internetnz.co.nz/> - InternetNZ sponsor of the Tui project, to provide the hardware used for the free deployments
- <http://www.soekris.com/> - Soekris manufacture the 4801 and 4501 units that are Tui's hardware platform