

UNIVERSITE LIBRE DE BRUXELLES
FACULTE DES SCIENCES
SERVICE TELEMATIQUE ET COMMUNICATION

Freenet6 Tunnel Setup Protocol Test Report

January 2003

Fuhua YIN

Faculté des Sciences
Service Télématique et Communication
CP 230 - Blvd du Triomphe
B-1050 Bruxelles

ULB - Rapport STC
03-02
mars 2003

Freenet6 Tunnel Setup Protocol Test Report

Reviewers

Name	Email	Remarks
Paul Van Binst	paul.vanbinst@helios.ihe.ac.be	
Rosette Vandebroucke	vandebroucke@helios.ihe.ac.be	

Revision History

History	Published date	Modification
Version 1.0	2002-05-28	
Version 2.0	2002-05-31	1) Adding reference 2) No need to add global routable address space in host-type tunnel 3) Editing
Version 3.0	2003-01-07	

Authors and Contributors

Name	Email	Remarks
Fuhua Yin	Fuhua.yin@helios.ihe.ac.be	Author
Marcin Michalak		Contributor

1	TUNNEL SETUP PROTOCOL	3
1.1	INTRODUCTION	3
1.2	HOST-TYPE TUNNEL SETUP PROCESS	4
1.3	ANALYSIS OF THE SHELL SCRIPT (LINUX.SH)	7
1.4	TEST PROCEDURES AND RESULTS	9
2	FREENET6 STATIC 48 PREFIX DELEGATION	10
2.1	INTRODUCTION	10
2.2	TEST PROCEDURES	12
2.2.1	<i>Step 1: Preparing Sumalinux</i>	12
2.2.2	<i>Step 2: Installing the TSP client software from freenet6</i>	12
2.2.3	<i>Step 3: Modifying tspc.conf</i>	13
2.2.4	<i>Step4: Running client and building a tunnel.</i>	13
2.2.5	<i>Step 5: Adding global routable IPv6 address space: 3ffe::/16, 2000::/3</i>	15
2.2.6	<i>Step 6: Ping and traceroute testing from STC-rose</i>	15
3	REFERENCE	16

1 Tunnel Setup Protocol

1.1 Introduction

Tunneling techniques often enable new networking functions while still preserving the underlying network as it is [REF 1]. Before establishing tunnels, the two end-points must exchange some parameters and negotiate some capability features. Those parameters or capability features may change over the time, which will cause re-configuring and establishing tunnels on both ends. The process of negotiating parameters and setting up tunnels is handed over to a control protocol- Tunnel Setup Protocol (TSP). It provides a framework of negotiation of tunnel parameters between a tunnel client and a tunnel server or broker.

There are two tunnel models: Tunnel Broker model and Tunnel Server model. In the tunnel broker model, the Tunnel Broker is taking charge of all communications between tunnel server and Tunnel Clients. The Tunnel clients query a broker for a tunnel and the broker finds a suitable tunnel server, asks a Tunnel server to setup the tunnel and send the tunnel information to the tunnel client. See Figure 1. The tunnel clients and tunnel servers or brokers already have the IPv4 Internet connections so that TSP messages could be carried by TCP.

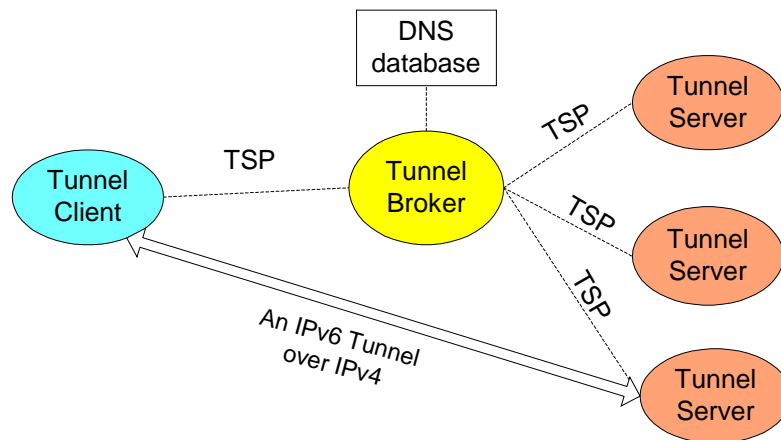


Figure 1 Tunnel Setup Protocol used in Tunnel Broker Model

In the Tunnel Server Model, the tunnel server directly talks to Tunnel Clients to setup the tunnel between the client and itself. See Figure 2.

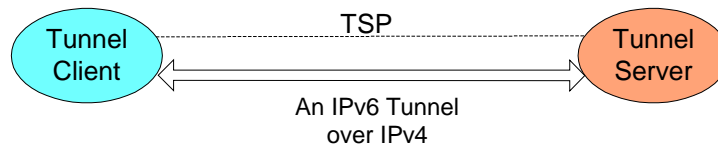


Figure 2 Tunnel Setup Protocol Used on Tunnel Service Model

Freenet6 is one of those who implement TSP and provides the free IPv6 over IPv4 tunnel service. Freenet6 uses the tunnel server model but the tunnel server has the functionality of a broker.

A host or server that is already connected to the Internet and has an IPv6 stack can initiate a tunnel setup process. A TSP client (**tspc**) reads a configuration file (**tspc.conf**) and then sends a request (using TCP) to the TSP server specified in the configuration file. The TSP server processes the request and (according to its local policy), assigns a single IPv6 address or a full IPv6 prefix to the requester. Next, the TSP server establishes a new configured tunnel (IPv6 over IPv4) according to the information sent in the request.

When the client receives tunnel information, it locally configures its tunnel interface and default IPv6 route. The client has now full IPv6 connectivity.

The information received is stored in environment variables and a shell script on Linux or a batch file for Windows is executed. The shell script will execute the commands needed to set the tunnel up. The shell script is called template. The client will execute the template specified in the configuration file. Users can customize the templates according to their local preferences.

Freenet6 provides two types of the tunnel services: host-type and router-type tunnel. The host-type tunnel allocates one site global IPv6 address, while router-type tunnel allocate one /48 prefix which allows the site to deploy huge amount of subnets. The host-type tunnel TSP process analysis and testing are described in Section 1.2, 1.3 and 1.4. The router-type one is taken up by the whole Section 2.

1.2 Host-Type Tunnel Setup Process

The TSP client is run in the client side and TSP server is run on the server side. The messages between TSP client and server are in XML format. Host-type and router-type tunnel setup processes are extremely similar. The only difference is in the parameters exchanged between the client and server.

The first parameter to be negotiated is the version number. The TSP client sends the version number to the TSP server. If this server supports this version, it will reply to the client that it is able to deal with this request. If this server does not support this version, it will send a list of servers that might support this version. The client will go to send a request to the other servers. The TSP process is illustrated in Figure 3. Note that the communication messages are in XML format, and the messages are shown for the host-type tunnel.

Then the client sends authentication information to the TSP server. The TSP server will send authentication result to the client. If it is successful, the client will send a request to build up a tunnel to the server. If the TSP server finds the tunnel resource and successfully allocates an IPv6 address to both sides of the tunnel, it will send all the tunnel information to the requester.

The TSP client then extracts the related information and puts it into a set of environment variables and the values of these environment variables will be passed to the shell script (in our case *linux.sh*). In the following section, we will analyze the main function of *linux.sh*.

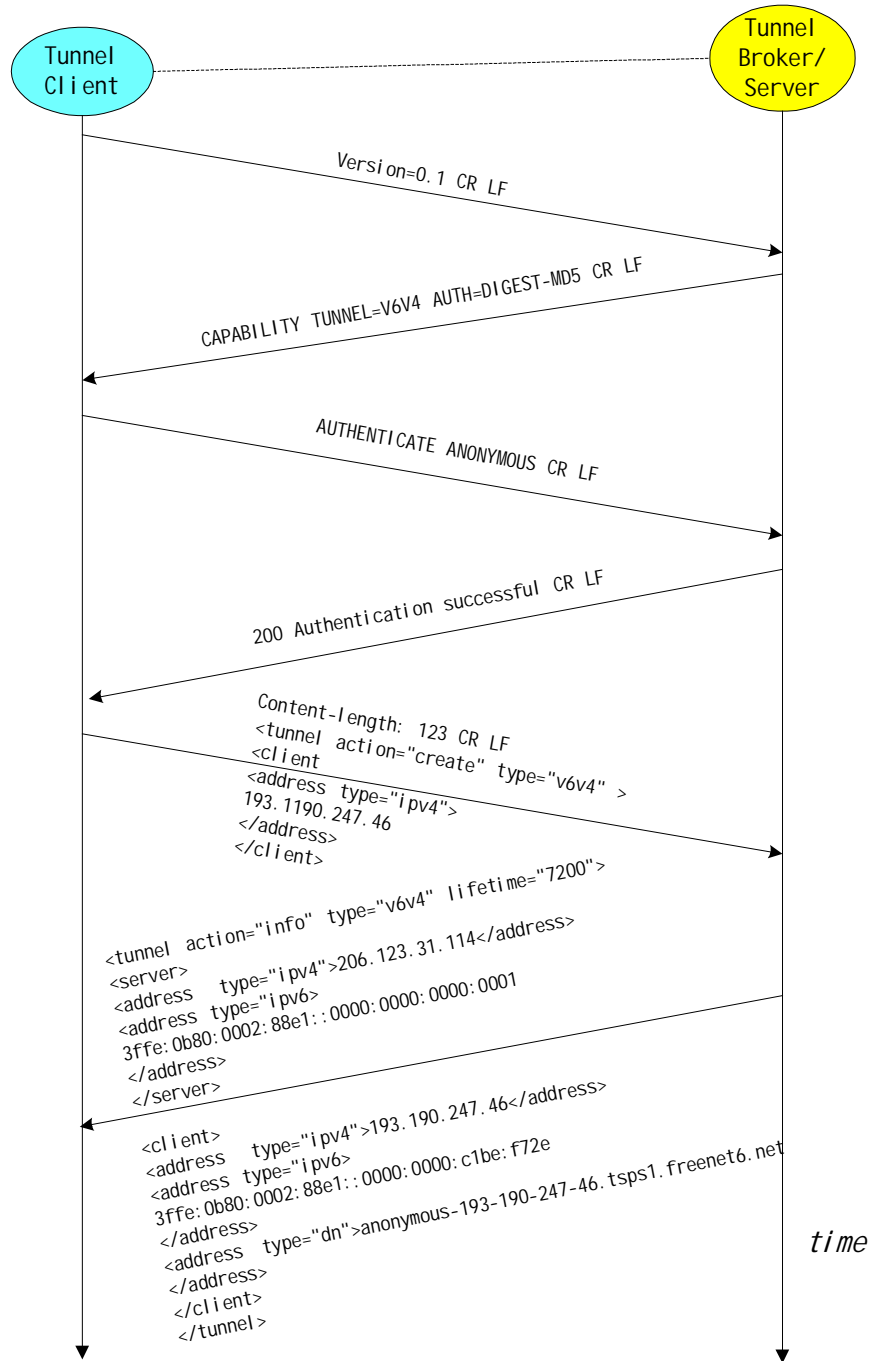


Figure 3 Host-Type TSP Process in a real example

1.3 Analysis of the Shell Script (Linux.sh)

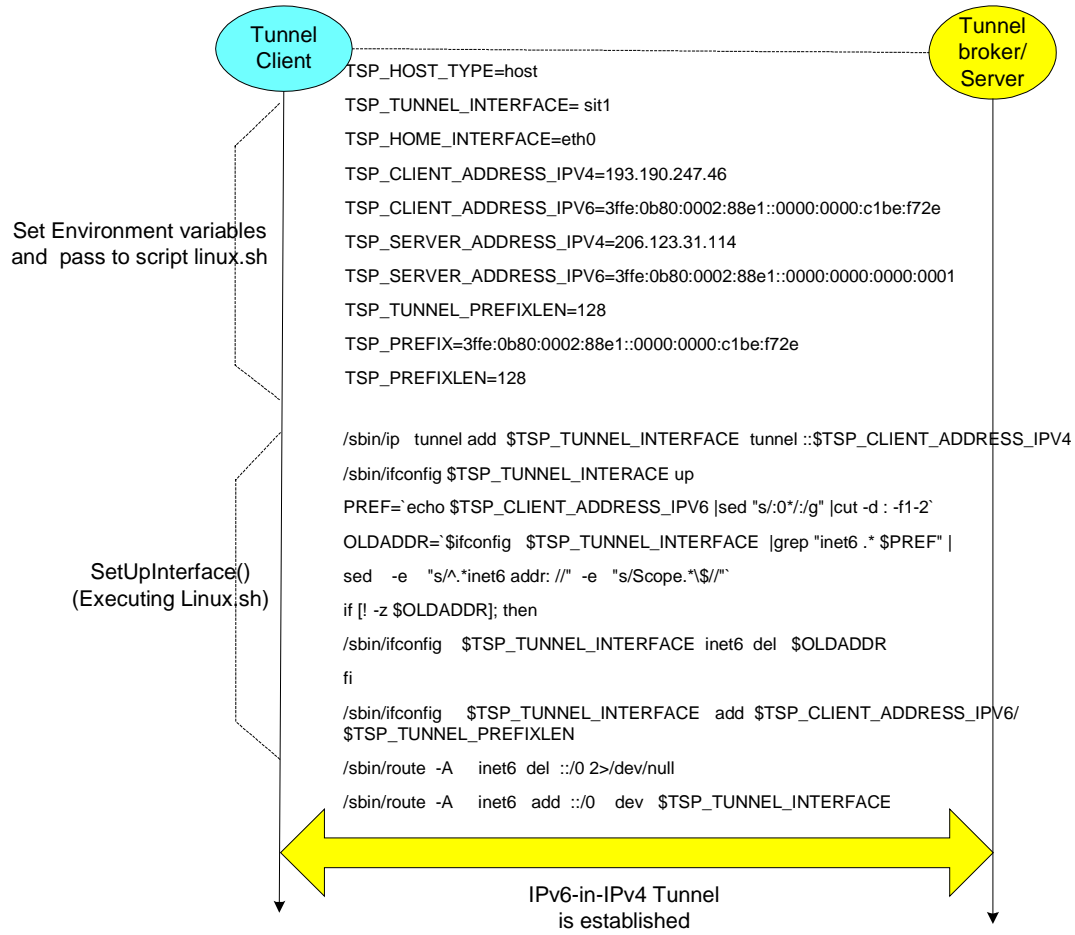


Figure 4 The Main Shell Script Contents

```

/sbin/ifconfig tunnel add $TSP_TUNNEL_INTERFACE tunnel
:::$TSP_CLIENT_ADDRESS_IPV4
    
```

Setting up a link to TSP server by using IPv6 and IPv4 compatible addresses.

```
/sbin/ifconfig $TSP_TUNNEL_INTERACE up
```

Enable the tunnel interface of sit1.

```
PREF=`echo $TSP_CLIENT_ADDRESS_IPV6 |sed "s/:0*/:/g" |cut -d : -f1-2`
```

Sed commands takes out prefix zeros in each column delimited by: in TSP_CLIENT_ADDRESS_IPV6. Cut command takes this resulting IPv6 address as input and picks up only the first two columns.

In our test case, TSP_CLIENT_ADDRESS_IPV6=3ffe:0b80:0002:88e1::0000:0000:c1be: f72e

After sed command is executed, the output is the following:

3ffe:b80: 2:88e1::c1be: f72e which is the input of cut command.

After cut command is executed, PREF=3ffe:b80.

If the tunnel interface sit1 has an IPv6 address beginning with 3ffe:b80, this address means an old tunnel IPv6 address and it must first be deleted.

The following command line sorts the old IPv6 tunnel address with the value of PREF and cuts out anything till “inet6 addr” in the beginning and anything ending with “Scope.*\”.

```
OLDADDR=`$ifconfig $TSP_TUNNEL_INTERFACE |grep "inet6 .* $PREF" | sed -e "s/^\.*inet6 addr: //" -e "s/Scope.*\$/\"`
```

The following command line means if the old IPv6 tunnel address exists, it will be deleted.

```
if [! -z $OLDADDR]; then
```

```
/sbin/ifconfig $TSP_TUNNEL_INTERFACE inet6 del $OLDADDR
```

```
fi
```

Then, configure the tunnel IPv6 interface address.

```
/sbin/ifconfig $TSP_TUNNEL_INTERFACE add
$TSP_CLIENT_ADDRESS_IPV6/$TSP_TUNNEL_PREFIXLEN
```

Add the default route.

```
/sbin/route -A inet6 del ::0 2>/dev/null
```

```
/sbin/route -A inet6 add ::/0 dev $TSP_TUNNEL_INTERFACE
```

1.4 Test Procedures and Results

- untar the package in a temporary directory
 - #cd /tmp
 - # tar zxf freenet6package
- Build the package according to the operation system
 - # cd freenet6-
 - # make
 - #make all target=linux
- Install the package
 - # make install target=linux installdir=/usr/local/tsp
- Run tspc
 - # cd /usr/local/tsp/bin
 - # ./tspc -vvf ./tspc.conf
- Check interface configuration and ping test
 - #ifconfig

```
eth0      Link encap:Ethernet  HWaddr 00:10:5A:01:26:4F
          inet addr:193.190.247.46  Bcast:193.190.247.255
          Mask:255.255.255.0
          inet6 addr: fe80::10:5a01:264f/10 Scope:Link
          inet6 addr: fe80::210:5aff:fe01:264f/10 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1467119 errors:0 dropped:0 overruns:0 frame:0
          TX packets:38830 errors:0 dropped:0 overruns:0 carrier:0
          collisions:169 txqueuelen:100
          RX bytes:325365442 (310.2 Mb)  TX bytes:3856466 (3.6 Mb)
          Interrupt:15 Base address:0x1000
```

```
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:360 errors:0 dropped:0 overruns:0 frame:0
          TX packets:360 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:23441 (22.8 Kb)  TX bytes:23441 (22.8 Kb)
```

```
sit1      Link encap:IPv6-in-IPv4
          inet6 addr: 3ffe:b80:2:88e1::c1be:f72e/128 Scope:Global
```

```

inet6 addr: fe80::c1be:f72e/10 Scope:Link
UP POINTOPOINT RUNNING NOARP MTU:1480 Metric:1
RX packets:52 errors:0 dropped:0 overruns:0 frame:0
TX packets:42 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:6288 (6.1 Kb) TX bytes:6464 (6.3 Kb)

```

➤ #ip 6 tunnel show

```

sit0: ipv6/ip remote any local any ttl 64 nopmtudisc
sit1: ipv6/ip remote 206.123.31.114 local any ttl 64

```

➤ #ping 3ffe:0b80:0002:88e1:0000:0000:0000:0001

```

PING 3ffe:0b80:0002:88e1:0000:0000:0000:0001 (3ffe:b80:2:88e1::1): 56
data bytes
64 bytes from 3ffe:b80:2:88e1::1: icmp_seq=0 ttl=64 time=126.66 ms
64 bytes from 3ffe:b80:2:88e1::1: icmp_seq=1 ttl=64 time=108.476 ms
64 bytes from 3ffe:b80:2:88e1::1: icmp_seq=2 ttl=64 time=109.675 ms
64 bytes from 3ffe:b80:2:88e1::1: icmp_seq=3 ttl=64 time=112.031 ms
64 bytes from 3ffe:b80:2:88e1::1: icmp_seq=4 ttl=64 time=113.971 ms
64 bytes from 3ffe:b80:2:88e1::1: icmp_seq=5 ttl=64 time=112.216 ms
64 bytes from 3ffe:b80:2:88e1::1: icmp_seq=6 ttl=64 time=182.568 ms
64 bytes from 3ffe:b80:2:88e1::1: icmp_seq=7 ttl=64 time=110.903 ms

```

```

--- 3ffe:0b80:0002:88e1:0000:0000:0000:0001 ping statistics ---
8 packets transmitted, 8 packets received, 0% packet loss
round-trip min/avg/max = 108.476/122.062/182.568 ms

```

Note: 3ffe:0b80:0002:88e1:0000:0000:0000:0001 is the freenet6 tunnel interface address for this specific tunnel.

➤ #traceroute6 3ffe:0b80:0002:88e1:0000:0000:0000:0001

```

1 3ffe:b80:2:88e1::1 (3ffe:b80:2:88e1::1) 110.838 ms 127.174 ms
122.909 ms

```

2 Freenet6 Static 48 prefix delegation

2.1 Introduction

Freenet6 static 48 prefix delegation is for the router-type tunnel. One /48 IPv6 prefix allows a site to deploy up to 65 535 subnets (/64 IPv6 prefix). Each subnet could handle 2^{64} nodes (18 446 744 073 709 551 616 IPv6 addresses), therefore, the number of unicast IPv6 addresses you could use with only one /48 prefix is very large: 65 535 subnets * 18 446 744 073 709 551 616.

Freenet6 TSP can also allocate a /48 IPv6 prefix to a site so that this site is able to deploy maximum 2^{64} subnets with each subnet of 2^{64} nodes. To obtain a /48 IPv6 prefix from

freenet6, you must create your own account from the freenet web page. In addition, you need to modify the tspc.conf by doing the following:

- Add userid and password
 - Add your userid to tspc.conf (e.g **userid=abcxxx123**)
 - Add your password to tspc.conf (e.g **passwd=F83%?fs21**)
- Add special paramaters to request /48 prefix
 - Add **host_type=router** in tspc.conf
 - Add **prefixlen=48** in tspc.conf
 - Add **if_prefix=YOUR_NETWORK_INTERFACE** in tspc.conf

The field YOUR_NETWORK_INTERFACE will be used to configure properly your computer to act as IPv6 router. IPv6 forwarding will be enabled between this interface and the configured tunnel (IPv6 over IPv4). IPv6 router advertisements will be activated on this interface to allow hosts to auto-configure by themselves their IPv6 addresses. However, the network interface name is not the same for every operating system. Please look on your system to find the right name to use as field.

Platform	Possible NETWORK_INTERFACE_NAME
Unix	eth0, eth1, fxp0, ep0, ed0, le0, lnc0,...
Windows 2000/NT	3,4,5,... Read Microsoft MSRIIPv6 document
Cisco	Tunnelxx, ethernet0/1, fast-ethernet0/1

If you install the TSP client software with demanding one /48 prefix in your host PC, you need to install a radvd.rpm package to make your pc behave like a router.

We have successfully built a /48 prefix delegation tunnel to freenet6 tunnel server. Our test environment consists of two PCs connected to the same LAN (STC) and all have the Internet connections. The TSP client software has been installed in the Linux platform computer called Sumalinux, which is supposed to act as a router. Another PC, called STC-rose, has Win2000 with IPv6 Stack in it.

STC-rose should be able to auto-configure its IPv6 address with knowledge from the advertisement of the router (Sumalinux). The test environment is described in Figure 5. In the following sections, we will describe our test procedures and results.

2.2 Test Procedures

2.2.1 Step 1: Preparing Sumalinux

```
# ip tunnel show
# ip tunnel del dev sit1 ( delete the existing tunnel)
# rpm -Uvh radvd.rpm ( install a router-like software)
```

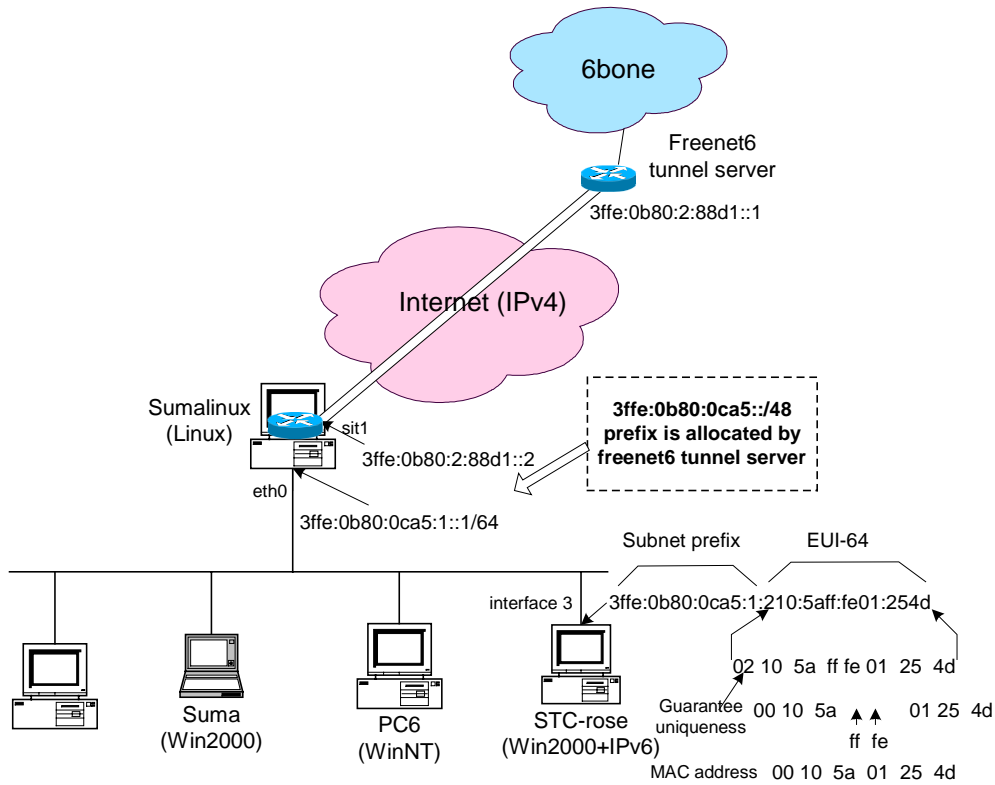


Figure 5 Test Environment for the Point-to-Point IPv6-in-IPv4 Tunnel a with /48 Prefix

2.2.2 Step 2: Installing the TSP client software from freenet6

- Untar the package in a temporary directory
 - #cd /tmp
 - # tar zxf freenet6package
- Build the package according to the operation system
 - # cd freenet6-0.9.6

- #make all target=linux
- Install the package
 - # make install target=linux installdir=/usr/local/tsp

2.2.3 Step 3: Modifying tspc.conf

Tspc.conf is under the directory of /usr/local/tsp/bin

```
#cd /usr/local/tsp/bin
# vi tspc.conf
```

```
delete userid=anonymous
add
```

```
userid=bamboostar
passwd=F83%?fs21
host_type=router
prefixlen=48
if_prefix=eth0
```

2.2.4 Step4: Running client and building a tunnel.

```
#cd /usr/local/tsp/bin
# ./tspc -vvvf ./tspc.conf
```

We obtain the following data after running tspc:

<i>TSP host type:</i>	<i>router</i>
<i>TSP tunnel interface:</i>	<i>sit1</i>
<i>TSP client IPv4 address:</i>	<i>193.190.247.46</i>
<i>TSP client IPv6 address:</i>	<i>3ffe:0b80:0002:88d1:0000:0000:0000:0002</i>
<i>TSP server IPv4 address:</i>	<i>206.123.31.114</i>
<i>TSP server IPv6 address:</i>	<i>3ffe:0b80:0002:88d1:0000:0000:0000:0001</i>
<i>TSP tunnel prefix length:</i>	<i>128</i>
<i>TSP prefix:</i>	<i>3ffe:0b80:0ca5</i>
<i>TSP Prefix length:</i>	<i>48</i>
<i>eth0 IPv6 address:</i>	<i>3ffe:0b80:0ca5:1::1/64</i>

These newly allocated IPv6 addresses on interfaces are indicated in Figure 5. The subnet of “router” sumalinux has IPv6 address space of 3ffe:0b80:0ca5:1::/64. STC-rose (Win2000+IPv6 stack) obtains its global IPv6 address by adding EUI-64 to the subnet prefix of 3ffe:0b80:0ca5:1::/64. EUI-64 is formed by inserting ffe in the middle of the MAC address of that interface and setting the second bit of the highest byte as 1 (Refer to Figure 5).

In STC-rose DOS command prompt

```
C:\> ipv6 if
```

We can find the interface 4 has a global IPv6 address related to the subnet prefix.

```
#ip tunnel show
```

```
sit0: ipv6/ip remote any local any ttl 64 nopmtudisc  
sit1: ipv6/ip remote 206.123.31.114 local any ttl 64
```

sit1 is the tunnel interface and the other end of the tunnel is the freenet6 server (206.123.31.114). sit0 does not represent a tunnel.

The interface configuration is shown by typing the command of ifconfig.

```
#ifconfig
```

```
eth0      Link encap:Ethernet  HWaddr 00:10:5A:01:26:4F  
          inet addr:193.190.247.46  Bcast:193.190.247.255  
          Mask:255.255.255.0  
          inet6 addr: fe80::10:5a01:264f/10 Scope:Link  
          inet6 addr: 3ffe:b80:ca5:1::1/64 Scope:Global  
          inet6 addr: fe80::210:5aff:fe01:264f/10 Scope:Link  
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1  
          RX packets:144875 errors:3 dropped:0 overruns:0 frame:5  
          TX packets:14379 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:27 txqueuelen:100  
          RX bytes:44871390 (42.7 Mb)  TX bytes:7883651 (7.5 Mb)  
          Interrupt:15 Base address:0x1000  
  
lo        Link encap:Local Loopback  
          inet addr:127.0.0.1  Mask:255.0.0.0  
          inet6 addr: ::1/128 Scope:Host  
          UP LOOPBACK RUNNING  MTU:16436  Metric:1  
          RX packets:47 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:47 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:0  
          RX bytes:3508 (3.4 Kb)  TX bytes:3508 (3.4 Kb)  
  
sit1     Link encap:IPv6-in-IPv4  
          inet6 addr: 3ffe:b80:2:88d1::2/128 Scope:Global  
          inet6 addr: fe80::c1be:f72e/10 Scope:Link  
          UP POINTOPOINT RUNNING NOARP  MTU:1480  Metric:1  
          RX packets:1392 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:11767 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:0
```

RX bytes:745079 (727.6 Kb) TX bytes:6843643 (6.5 Mb)

Till now, we have built a point-to-point tunnel to the freenet6 tunnel server. We can ping to www.kame.net and other IPv6 available addresses. However we could not send packets from STC-rose to outside IPv6 addresses. The problem is the script linux.sh has not built routes to the global routable IPv6 address space. If we check the IPv6 routing on sit1, there is no global IP address

```
# route -A inet6|grep sit1
```

```
fe80::/10          ::          UA      256    0      0  sit1
ff00::/8          ::          UA      256    0      0  sit1
::/0              ::          U       1      0      0  sit1
```

We have solved this problem by doing the next step.

2.2.5 Step 5: Adding global routable IPv6 address space: 3ffe::/16, 2000::/3

```
# route -A inet6 add 3ffe::/16 dev sit1
# route -A inet6 add 2000::/3 dev sit1
# route -A inet6 |grep sit1
```

```
3ffe::/16          ::          U       1      0      0  sit1
2000::/3           ::          U       1      0      0  sit1
fe80::/10          ::          UA      256    0      0  sit1
ff00::/8          ::          UA      256    0      0  sit1
::/0              ::          U       1      0      0  sit1
```

2.2.6 Step 6: Ping and traceroute testing from STC-rose

```
C:\> ping6 3ffe:b80:2:88d1::1
```

```
Pinging 3ffe:b80:2:88d1::1 with 32 bytes of data:
Reply from 3ffe:b80:2:88d1::1: bytes=32 time=125ms
Reply from 3ffe:b80:2:88d1::1: bytes=32 time=325ms
```

Note: 3ffe:b80:2:88d1::1 is the tunnel interface address in the side of Freenet6 server in Canada.

```
C:\> tracert6 3ffe:b80:2:88d1::1
```

```
Tracing route to 3ffe:b80:2:88d1::1
over a maximum of 30 hops:
```

```
 1    <1 ms    <1 ms    <1 ms    3ffe:b80:ca5:1::1
 2   143 ms   120 ms   127 ms   3ffe:b80:2:88d1::1
```

Trace complete.

Note: The packets sending from STC-rose experiences 2 hops. In the first hop, the packets arrive at Sumalinux and in the second hop arrive at Freenet server in Canada. The Sumalinux really acts as a router to forward packets for STC-rose.

C:\> ping6 www.kame.net

Pinging kame220.kame.net [3ffe:501:4819:2000:210:f3ff:fe03:4d0] with 32 bytes of data:

Request timed out.

Reply from 3ffe:501:4819:2000:210:f3ff:fe03:4d0: bytes=32 time=596ms

Reply from 3ffe:501:4819:2000:210:f3ff:fe03:4d0: bytes=32 time=598ms

Reply from 3ffe:501:4819:2000:210:f3ff:fe03:4d0: bytes=32 time=578ms

C:\> tracert6 www.kame.net

Tracing route to kame220.kame.net [2001:200:0:4819:210:f3ff:fe03:4d0] over a maximum of 30 hops:

1	<1 ms	<1 ms	<1 ms	3ffe:b80:ca5:1::1
2	144 ms	125 ms	129 ms	3ffe:b80:2:88d1::1
3	124 ms	126 ms	117 ms	VIAGENIE.r00.snjsca03.us.b6.verio.net [2001:218:0:80:1:840:1:e]
4	426 ms	429 ms	413 ms	3ffe:8000:ffff:b::1
5	742 ms	662 ms	678 ms	2001:200:0:1802:2a0:ccff:fe73:4413
6	714 ms	739 ms	708 ms	pc6.otemachi.wide.ad.jp [2001:200:0:1800::9c4:0]
7	717 ms	820 ms	714 ms	2001:200:0:6001:2a0:ccff:fe73:386c
8	746 ms	737 ms	708 ms	pc3.yagami.wide.ad.jp [2001:200:0:1c04::1000:2000]
9	704 ms	705 ms	729 ms	paradise.kame.net [3ffe:501:4819:2000:2e0:18ff:fe98:f19d]
10	*	705 ms	707 ms	2001:200:0:4819:210:f3ff:fe03:4d0

Trace complete

3 Reference

REF 1 Tunnel Setup Protocol, IETF draft-vg-ngtrans-tsp-00.

REF 2 <http://www.freenet6.net>

REF 3 <http://bieringer.de/linux/IPv6-HOWTO/script/current>

