IRISA

INSTITUT DE RECHERCHE
EN INFORMATIQUE
ET SYSTÈMES ALÉATOIRES

Campus Universitaire de Beaulieu
35042 RENNES CEDEX FRANCE
Tél.: 02 99 84 71 00 - Télex : UNIRISA 950 473 F
Télécopie : 02 99 84 71 71

V1.5

«Remote Interoperability Test specifications for Transition Mechanisms»

- **6to4**
- **SIIT/NAT-PT**

Authors:

César Viho
Frédéric Roudaut
Octavio Médina
Sébastien Barbin

IRISA
Campus de Beaulieu
35042 Rennes Cedex

Table Of Contents

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE
(UPRESA 6074) UNIVERSITÉ DE RENNES I I.N.S.A. DE RENNES

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE
ET EN AUTOMATIQUE - UNITÉ DE RECHERCHE DE RENNES

1. Introduction

The transition mechanisms are a set of mechanisms for ensuring the integration of IPv6 networks in some existing IPv4 infrastructures, and guaranteeing communications between the two IPv4 and IPv6 worlds.

We can classify the transition mechanisms in four sets:

- *Mechanisms permitting the construction of an IPv6 network over an IPv4 infrastructure*. The set of these mechanisms uses essentially v6 over v4 tunnels. The main mechanisms developed are 6over4 (Transmission of IPv6 over IPv4 Domains without Explicit Tunnels), ISATAP(Intra-Site Automatic Tunnel Addressing Protocol).
- *Mechanisms permitting the accessibility to an already existing IPv6 network*. We can distinguish transition mechanisms like 6to4(Connection of IPv6 Domains via IPv4 Clouds), Tunnel Broker.
- *Cohabitation mechanisms* developed in order to permit communication between IPv4 and IPv6 applications. Main mechanisms are SIIT, NAT-PT where the transition is only done at the edge of the site by a header translation; and DSTM(Dual Stack Transition Mechanism), which use IPv4 over IPv6 tunnel inside the site with the advantage of allowing IPv4 application to communicate over IPv6 infrastructure even if they have not been v6fied (not adapted to be used with IPv6).
- *Mechanisms to generate IPv6 packets from IPv4 applications*. The main mechanisms are Bump in the Stack and Bump in the API.

This document provides Interoperability scenarios for testing Transition Mechanisms such as 6to4, SIIT and NAT-PT in a **"Multi-sites remote session".**

Even though test scenarios were verified, there can still be a few mistakes. All suggestions are welcome and can be send to:

- Frédéric Roudaut (frederic.roudaut@irisa.fr)
- Sébastien Barbin (sbarbin@irisa.fr)
- César Viho (Cesar.Viho@irisa.fr)

2. The 6to4 mechanism

This part provides Interoperability scenarios for testing the 6to4 mechanism in a **"Multi-sites remote session".**

The 6to4 Mechanism is described in RFC 3056, Connection of IPv6 Domains via IPv4 Clouds, B. Carpenter, K. Moore, February 2001.

The motivation for the 6to4 Mechanism is to allow isolated IPv6 domains or hosts, attached to an IPv4 network which has no native IPv6 support, to communicate with other such IPv6 domains or hosts with minimal manual configuration, before they can obtain native IPv6 connectivity.

This test suite does not take into account the possibly use of the DNS with the 6to4 Mechanism. Nevertheless, if needed, tests can be very easily and quickly added. It will involve new issues such as in particular Address Selection:

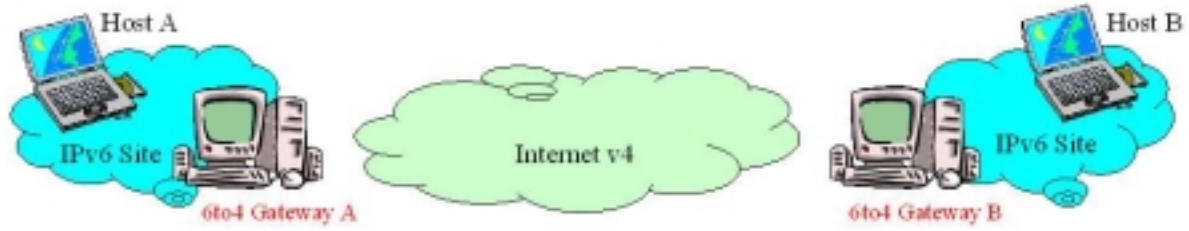
- If one host has only a 6to4 address, and the other one has both a 6to4 and a native IPv6 address, then the 6to4 address should be used for both.
- If both hosts have a 6to4 address and a native IPv6 address, then either the 6to4 address should be used for both, or the native IPv6 address should be used for both. The choice should be configurable. The default configuration should be native IPv6 for both.

2.1 Test architecture

For testing 6to4 in a remote test session we need at least two IPv6 sites without Native IPv6 connectivity in the first step. For the second step at least one of both sites MUST have a native IPv6 connectivity.
Both sites MUST be reachable via the IPv4 Internet.

On each site a global v4 address MUST be assigned. At the edge of both sites a 6to4 router will be present. The different 6to4 routers will be configured for their respective addresses.

6to4 Gateway A and B advertise their 2002:v4addr/48 prefix within their domains with Router Advertisement. Host A and B will configure their 6to4 address with their 2002:v4addr::/48 prefix, and their 6to4 router as their default router.



2.2 Interoperability Test suite for 6to4

2.2.1 Base Tests

a. Base Dialogue

- Send a Ping6 From Host A to Host B

  - 6To4 Gateway A MUST encapsulate echo requests from A in IPv4 Packets for B. It MUST decapsulate IPv4-in-IPv6 Echo Replies from Host B to Host A
  - Moreover 6To4 Gateway B MUST decapsulate echo requests from A in IPv4 Packets for Host B. It MUST encapsulate IPv4-in-IPv6 Echo Replies from Host B to Host A

b. ICMP Destination Unreachable

- Send a Ping6 From Host A to one new Host A'. A' owns a 2002:V4addr/48 prefix, but its V4addr address MUST not be reachable from 6to4 Gateway A.

  - 6To4 Gateway A SHOULD reply to Host A with an ICMPv6 Destination Unreachable message

c. ICMP Time Exceeded

- Send a Ping6 From Host A to Host B with Hop Limit set to 0

  - 6To4 Gateway A MUST discard the Packet and send an ICMPv6 Time exceeded Message with code 0 to Host A

- Send a Ping6 From Host A to Host B with Hop Limit set to 1

  - 6To4 Gateway A MUST decrement the current Hop Limit. It MUST discard the Packet and send an ICMPv6 Time exceeded Message with code 0 to Host A.

- Send a Ping6 From Host A to Host B with Hop Limit set to 2

  - 6To4 Gateway A MUST encapsulate echo requests from A in IPv4 Packets for B. 6To4 Gateway A MUST decrement the current Hop Limit to 1.
  - Moreover 6To4 Gateway B MUST decapsulate echo requests from A in IPv4 Packets for Host B. It MUST decrement the current Hop Limit to 0 and send an encapsulated ICMPv6 Time exceeded Message with code 0 to Host A.

- 6To4 Gateway A MUST decapsulate IPv4-in-IPv6 ICMPv6 Time exceeded Message with code 0 to Host A

2.2.2 Testing Addressing

- Send a Ping From Host A to 0.0.0.0 (2002::1)
  - Packets should be silently discarded

- Send a Ping From Host A to 127.0.0.1 (2002:7f00:0001::1)
  - Packets should be silently discarded

- Send a Ping From Host A to 224.0.0.1 (2002:e000:0001::1)
  - Packets should be silently discarded

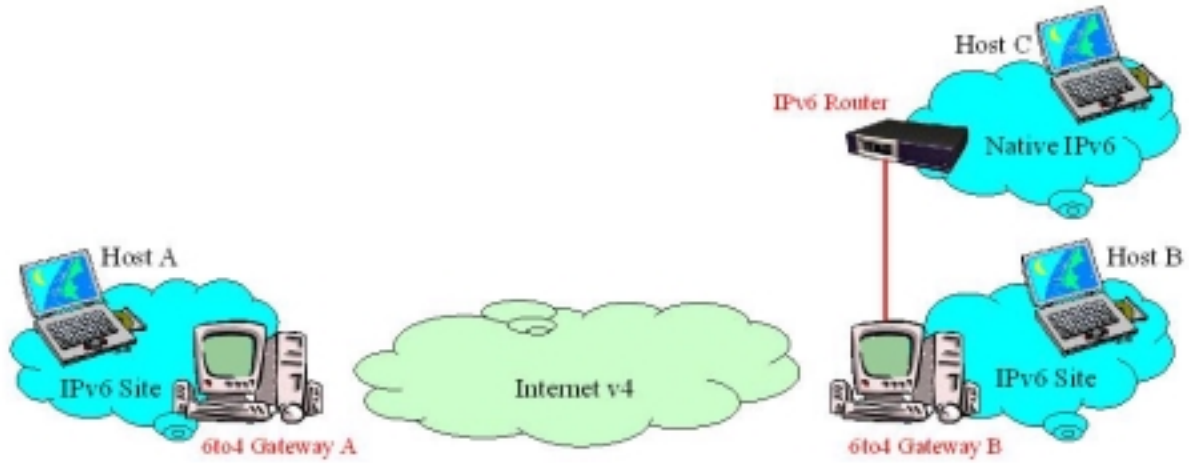
- Send a Ping From Host A to 10.255.255.255 (2002:0aff:ffff::1)
  - Packets should be silently discarded

2.2.3 Testing Fragmentation

- Send a Ping6 packet from Host A to Host B with the Packet size greater than the IPv4 MTU (1500 bytes minus 20 bytes by default for IPv4 over Ethernet).
  - The Packet should be discarded and the 6To4 Gateway A MUST send an ICMP “packet too big” to Host A

2.2.4 Connectivity to Native IPv6 Link

In this test we need an IPv6 Native connectivity. The New test architecture is presented hereafter:



The 6to4 Gateway B that is a relay router is set as default route for 6to4 Gateway A. We will not announce our IPv6 Native Prefix using a routing protocol such as BGP4+.
A native IPv6 prefix is assigned to the IPv6 router.
The IPv6 Router advertises the v6 domain prefix within the v6 domain.
Host C configures its native v6 address and set the IPv6 Router as its default router.
6to4 Gateway A and B advertise their 2002:v4addr/48 prefix within their domains.
Host A and B configure their 6to4 address with their 2002:v4addr::/48 prefix and their 6to4 router as their default route.

For our test we have to add a path in the 6to4 Gateway B for the Native Prefix and a path in the IPv6 router for the 6to4 2002::/16 prefix. It can be done dynamically using a routing protocol between the 6to4 Gateway B and the IPv6 router or statically with the following:

- Add a path in the routing table of Gateway B for the Native IPv6 Prefix or even a default route.
- Add one entry in the routing table of the IPv6 router for the 6to4 prefix 2002::/16

Do the following:

- Send a Ping6 from Host A to Host C

  - 6To4 Gateway A MUST encapsulate echo requests from A in IPv4 Packets for C. It MUST decapsulate IPv4-in-IPv6 Echo Replies from Host C to Host A
  - Moreover 6To4 Gateway B MUST decapsulate echo requests from A in IPv4 Packets for Host C. This IPv6 Packet MUST be forwarded to the IPv6 router. Furthermore, Echo replies send by the IPv6 Router from Host C MUST be encapsulated in the 6to4 Gateway B in IPv4 packet for 6to4 Gateway A.

- Send a Ping6 from Host B to Host C

This packet MUST be forwarded by the 6to4 Gateway to the IPv6 router. Host C MUST reply to Host B with an IPv6 Echo Reply through the IPv6 router and through the 6to4 Gateway B.

- Send a Ping6 from Host A to Host B

  - 6To4 Gateway A MUST encapsulate echo requests from A in IPv4 Packets for B. It MUST decapsulate IPv4-in-IPv6 Echo Replies from Host B to Host A
  - Moreover 6To4 Gateway B MUST decapsulate echo requests from A in IPv4 Packets. It MUST encapsulate IPv4-in-IPv6 Echo Replies from Host B to Host A

3. The SIIT and NAT-PT mechanisms

This part provides Interoperability scenarios for testing the SIIT and NAT-PT mechanisms in a **"Multi-sites remote session".**

SIIT is a protocol translation mechanism at the edge of the network that allows communication between IPv6-only and IPv4-only nodes via protocol independent translation of IPv4 and IPv6 datagrams, requiring no state information for the session. The SIIT proposal assumes that V6 nodes are assigned a V4 address for communicating with V4 nodes.

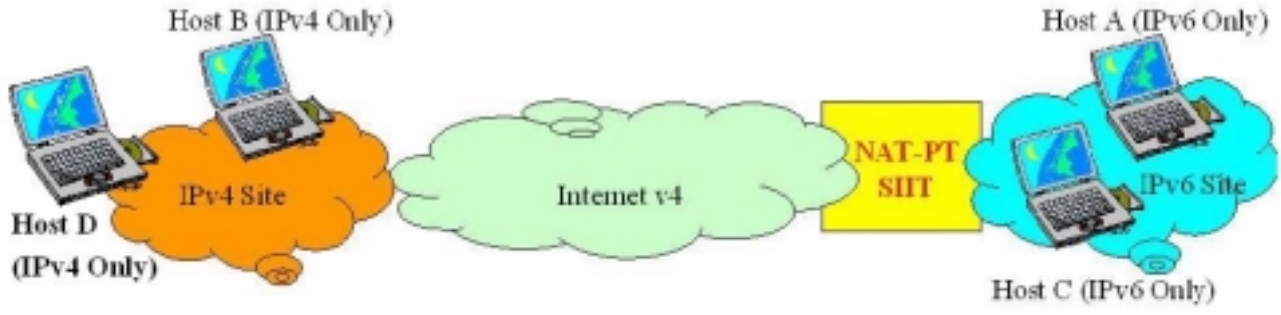
NAT-PT is quite similar to SIIT. It provides transparent routing to end-nodes in V6 realm trying to communicate with end-nodes in V4 realm and vice versa. This is achieved using a combination of Network Address Translation and Protocol Translation. Nevertheless, NAT-PT uses a pool of V4 addresses for assignment to V6 nodes on a dynamic basis as sessions are initiated across V4-V6 boundaries.

The SIIT Mechanism is described in RFC 2765, Stateless IP/ICMP Translation Algorithm (SIIT), February 2000. The NAT-PT Mechanism is described in RFC2766, Network Address Translation - Protocol Translation (NAT-PT), February 2000

This test suite does not take into account the possibly use of the DNS with the SIIT/NAT-PT Mechanism.

3.1 Test architecture

The test architecture consists in an IPv6 site and an IPv4 site interconnected by the Internet v4. The IPv6 site MUST contains an IPv6-only host and the NAT-PT/SIIT box whereas the IPv4 site needs only an IPv4-only host.
Moreover, we will need two other hosts on the IPv6 and IPv4 sites to induce ICMP error messages.



3.2 Interoperability Test suite for SIIT and NAT-PT

3.2.1 IPv6 To IPv4

a. Simple Scenario

Application Requirements:

- Need IPv6 application capable of sending individual UDP packets
- Application must be able to modify Hop Limit (ex-TTL)
- Application must be able to include extensions in IPv6 header.
- Application must be able to send fragmented packets.

NAT-PT Setup:

- Setup IPv4 Address pool in NAT box.
- Setup announcement of IPv6 route containing NAT prefix in the IPv6 domain.
- Setup reverse routing in IPv4 domain to force packets to be routed to NAT box.

Initialization:

- Verify that IPv6 routing is correct
- Verify that IPv4 routing is correct
- Verify that no state is present in NAT box.

Test a1: General Case

Test Packet Structure

- UDP/IPv6 packet
- Packet is not a fragment (NO Fragment Header Extension)
- No extensions
- Hop Limit > 1
- src/dst addresses: variable

Expected Format of Translated IPv6 Header

- Version = 4
- Hdr Len = 5 (no options)
- ToS = IPv6 TCLASS
- Total Length = IPv6 Payload Length+ IPv4 header length (20)
- ID = 0
- Flags MF = 0, DF = 1
- Frag Offset = 0
- TTL = IPv6 Hop Limit - 1
- Proto = IPv6 Next Header
- Hdr Chksum = Needs to be computed after IPv4 header creation

Expected format of translated addresses

- *Classic SIIT behavior*

  - Src Addr = if IPv6 Src Addr = translated addr
    ➔ IPv4 addr = lower 32 bits of IPv6 addr
    else
    ➔ IPv4 addr = 0.0.0.0
  - Dst Addr = if IPv6 Dst Addr = mapped addr
    ➔ IPv4 addr = lower 32 bits of IPv6 addr
    else
    ➔ not translated? (Test 3.1.b)

- *NAT-PT behavior*

  - Src Addr = standard IPv6 Addr of A
    ➔ IPv4 addr = address from pool
    ➔ state is created in NAT box to keep mapping
  - Dst Addr = if IPv6 Dst Addr = [NAT prefix]::[IPv4 addr of B]
    ➔ IPv4 addr = lower 32 bits of IPv6 addr
    else
    ➔ not translated?

SIIT Testing

1. Send test pkt (src=translated addr, dst=mapped addr)
   ➔ packet is translated

2. Send test pkt (src=NO translated addr, dst=mapped addr)
   ➔ packet is translated, IPv4 src addr = 0.0.0.0

3. Send test pkt (src=translated addr, dst=NO mapped addr)
   ➔ packet should be dropped

NAT-PT Testing:

4. Send test pkt (src=standard IPv6 addr, dst=[NAT prefix]::[IPv4 addr of B])
   ➔ packet is translated, state is created

5. Send second test pkt (src, dst addresses same as 1)
   ➔ packet is translated, IPv4 src addr must be the same as 1.

6. Send test pkt (src=standard IPv6 addr, dst=NO [NAT prefix]::[IPv4 addr of B])
   ➔ packet should be dropped

Test a2: IPv6 extensions present

Test Packet Structure

- UDP/IPv6 packet
- Packet is not a fragment (NO Fragment Header Extension)
- IPv6 src address = standard IPv6 address of node (requires state previously created)
- IPv6 dst address = [NAT prefix]::[IPv4 addr of B]
- Hop Limit > 1
- extensions present

Testing: extensions, other than routing header, are present

1. Send Test packet from A to B, include Hop-by-Hop extensions
   - ➔ packet is translated
   - ➔ extensions are ignored
   - ➔ IPv4 Total Length adjusted to skip Extension headers.

2. Send Test packet from A to B, include destination extensions
   - ➔ packet is translated
   - ➔ extensions are ignored
   - ➔ IPv4 Total Length adjusted to skip Extension headers.

Testing: routing header extension present

3. Send Test packet to NAT box. Include routing header extension with Segments Left = 0.
   - ➔ packet is translated, extension ignored, IPv4 Length adjusted.

4. Send Test packet to NAT box. Include routing header extension with Segments Left > 0.
   - ➔ packet is dropped
   - ➔ return ICMPv6 "parameter problem" (Type 4/Code 0)

Test a3: Hop Limit = 0

Test Packet Structure

- UDP/IPv6 packet
- Packet is not a fragment (NO Fragment Header Extension)
- IPv6 src address = standard IPv6 address of node (requires state previously created)
- IPv6 dst address = [NAT prefix]::[IPv4 addr of B]
- no extensions
- Hop Limit = 0

Testing

1. Send Test packet from A to B, set Hop Limit = 0
   ➔ packet is dropped
   ➔ return ICMPv6 "TTL exceded"

b. Fragmentation Handling

Application Requirements: c.f. section 3.2.1.a
NAT-PT Setup: c.f. section 3.2.1.a
Initialization: c.f. section 3.2.1.a

Test b1: IPv6 Fragment Header Extension present (Generic)

Test Packet Structure

- UDP/IPv6 packet
- Packet IS fragmented (Fragment Header Extension present)
- Pkt length < 1280
- No other extensions
- Hop Limit > 1
- IPv6 src address = standard IPv6 address of node (requires state previously created)
- IPv6 dst address = [NAT prefix]::[IPv4 addr of B]

Expected Format of Translated IP Header

- Version = 4
- Hdr Len = 5 (no options)
- ToS = IPv6 TCLASS
- Total Len = IPv6 Payload Length - Frag Ext Hdr length (8) + IPv4 hdr length (20)
- ID = low order 16 bits in Frag Hdr ID
- Flags MF = IPv6 Frag Hdr MF, DF = 0
- Frag Offset = IPv6 Frag Hdr Offset
- TTL = IPv6 HopLimit - 1
- Proto = Frag Hdr Next Hdr
- Hdr Chksum = Needs to be computed after IPv4 hdr creation
- Src Addr = IPv4 src addr = address from pool
  ➔ state is created in NAT box to keep mapping
- Dst Addr = IPv4 dst addr = lower 32 bits of IPv6 dst addr

Testing:

1. Send Test (fragmented) packet from A to B

2. Verify header translation
   ➔ Verify ID, flags, fragment offset is correctly mapped
   ➔ Verify Checksum is correctly computed (at destination).

3. Verify payload is not modified.
   ➔ no reassembly problems at B.

Test b2: Fragmentation Required

Test Packet Structure

- UDP/IPv6 packet
- Packet NOT fragmented (Fragment Header Extension NOT present)
- Pkt length > 1280
- No extensions
- Hop Limit > 1
- IPv6 src address = standard IPv6 address of node (requires state previously created)
- IPv6 dst address = [NAT prefix]::[IPv4 addr of B]

Testing:

1. Send test packet from A to B. (Packet Length > 1280 bytes)

2. Verify IPv4 packets on output
   - ➔ packet is translated (cf. test 3.2.1.a1)
   - ➔ several IPv4 fragments are generated
   - ➔ Verify ID (value?), flags and Fragment Offset in each one of the fragments

3. If packet is not translated
   - ➔ ICMP "packet too big" must be sent to A by NAT box.
   - ➔ If ICMP sent, verify next packets have Pkt Length < 1280.

c. ICMPv6 -> ICMPv4

Application Requirements:

- Need IPv6 OS implementing ICMP
- ICMP implementation must be able to send informational as well as error ICMP messages.
- Need 3rd station (C) in IPv6 network to induce ICMP error messages
- IPv6 address of C = [NAT prefix]::[IPv4 addr of B]
- Because of routing, replies from A go to NAT box, not to C.
- IPv6 Gateway capacity may be needed (have A react like an IPv6 router)

NAT-PT Setup: c.f. section 3.2.1.a
Initialization: c.f. section 3.2.1.a

Expected format of translated packet (IP + ICMP)

- IP Header Translation: c.f. section 3.2.1.a
- ICMP Header Translation

  - type = must be translated (see below)
  - code = must be translated (see below)
  - checksum = must be recomputed to exclude pseudo-header and type/code change.

Test c1: ICMP Informational Messages

Test Packet Structure

- ICMPv6/IPv6 packet
- Hop Limit > 1
- IPv6 src address = std IPv6 addr of node (requires state previously created)
- IPv6 dst address = [NAT prefix]::[IPv4 addr of B]
- test of several type/code for ICMP.

Testing:

1. Send ICMP echo request from A to B

2. Verify ICMPv4 translated packet
   - ➔ Type & Code translated
     - echo request: IPv6 Type= 128, IPv4 Type = 8
     - echo reply: IPv6 Type= 129, IPv4 Type = 0
   - ➔ ICMP checksum recomputed

3. Verify reception on B
   ➔ B must send ICMPv4 echo reply

4. Send Multicast ICMPv6 messages (types 130,131,132)
   ➔ verify packet is silently dropped

5. Send Neighbor Discovery ICMPv6 messages (types 133,134,135,136,137)
   ➔ verify packet is silently dropped

Test c2: ICMP Error Messages

Test Packet Structure

- ICMPv6/IPv6 packet
- Hop Limit > 1
- IPv6 src address = standard IPv6 address of node (requires state previously created)
- IPv6 dst address = [NAT prefix]::[IPv4 addr of B]
- test of several type/code for ICMP errors.

Expected Behavior:

- Type and code must be translated according to table below.
- checksum needs to be recomputed
- Embedded header must be translated (cf. section 3.2.1.a1)

(ICMP packets contain the IP header of packet that originated the error)

ICMPv6 error name	ICMPv6 type,code	translated ICMPv4 type,code	additional actions
no route to dest	1,0	1,1	
comm with dest adm proh	1,1	1,10	
beyond of scope of src addr	1,2	1,1	
address unreach	1,3	1,1	
port unreach	1,4	1,3	
packet too big	2,x	3,4	modifiy MTU field
time exceeded	3,x	3,11	
parameter problem	4,1	3,2	
parameter problem	4,(!=1)	12,0	update pointer to param
unknown			silently drop

Testing:

1. Send IPv6 packet from C to A
   - A activated as router
   - pkt dst address unreachable from A
   - ICMP "destination unreachable" is generated (1,0)

2. Verify pkt translation
   - ICMPv4 type/code are translated according to table A
   - What IPv4 src address is given to A?
   - What IPv4 address given to unreachable node?

3. Send IPv6 packet from C to A
   - pkt length is too big
   - ICMP "packet too big" is generated (2,0)

4. Verify pkt translation (same as 2)

5. Send IPv6 packet from C to A
   - Hop Limit = 0
   - ICMP "Time Exceeded" is generated (3,0)

6. Verify pkt translation (same as 2)

3.2.2 IPv4 To IPv6

a. Simple Scenario

Application Requirements:

- Need IPv4 application capable of sending individual UDP packets
- Application must be able to modify TTL
- Application must be able to include options in IPv4 header.
- Application must be able to send fragmented packets.

NAT-PT Setup:

- Setup IPv4 Address pool in NAT box.
- State must have been previously created to map A IPv4 -> IPv6 address.
- Setup announcement of IPv6 route containing NAT prefix in the IPv6 domain.
- Setup reverse routing in IPv4 domain to force packets to be routed to NAT box.

Initialization:

- Verify IPv6 routing is correct
- Verify IPv4 routing is correct
- Verify state is present in NAT box (mapping A6 <-> A4).

Test a1: General Case

Test Packet Structure

- UDP/IPv4 packet
- Packet is not a fragment (DF=1, MF=0, Fragment offset=0)
- No options
- TTL > 1
- src addr = IPv4 address of B
- dst addr = allocated IPv4 address of A

Expected Format of Translated IPv4 Header

- Version = 6
- Tclass = IPv4 ToS (might be 0)
- Flow Label = 0
- Payload Len = IPv4 length - IPv4 header length (20)
- Next header = IPv4 Protocol field

- Hop Limit = IPv4 TTL - 1

Expected format of translated addresses

Classic SIIT behavior

- Src Addr = ::ffff:[IPv4 addr of B] (mapped address)
- Dst Addr = ::ffff:0:[IPv4 addr of A] (translated address)

NAT-PT behavior

- Src Addr = [NAT prefix]::[IPv4 addr of B]
- Dst Addr = standard IPv6 address of A

SIIT Testing:

1. Send test packet from B to A
   ➔ Verify construction of translated packet (translation always occur)

NAT-PT Testing:

1. Send test packet from B to A, state exists
   ➔ Verify construction of translated packet

2. Send test packet from B to A, state DOES NOT exist
   ➔ Verify behavior (packet is not translated)
   ➔ What happens? (ICMP message to B?)

Test a2: IPv4 options present

Test Packet Structure

- UDP/IPv4 packet
- Packet is not a fragment (DF=1, MF=0, Fragment offset=0)
- options
- TTL > 1
- src addr = IPv4 address of B
- dst addr = allocated IPv4 address of A

Testing: options, other than source route, are present

1. Send Test packet from B to A. Include options (other than unexpired Source Route)

2. Verify IPv6 packet construction on output
   - ➔ packet is translated (c.f. test 3.2.2.a1)
   - ➔ Options are discarded

Testing: source route option present

3. Send IPv4 packet from B to A. Include unexpired Source Route option
4. IPv6 packet must not be created
   - ➔ packet is discarded
   - ➔ must return ICMPv4 "destination unreachable/source route failed" (Type 3/Code 5)

Test a3: TTL = 0

Test Packet Structure

- UDP/IPv4 packet
- Packet is not a fragment (DF=1, MF=0, Fragment offset=0)
- no options
- TTL =0
- src addr = IPv4 address of B
- dst addr = allocated IPv4 address of A

Testing:

1. Send IPv4 packet from B to A. Set TTL = 0

2. IPv6 packet must not be created
   - ➔ packet is discarded
   - ➔ must return ICMPv4 "TTL exceeded"

b. Fragmentation Handling

Application Requirements: (c.f. section 3.2.1.a)
NAT-PT Setup: (c.f. section 3.2.1.a)
Initialization: (c.f. section 3.2.1.a)

Test b1: IPv4 header flag DF=0 (Generic)

Test Packet Structure

- UDP/IPv4 packet
- Packet IS fragmented (DF=0)
- Pkt length < 1280
- No options
- TTL > 1
- src addr = IPv4 address of B
- dst addr = allocated IPv4 address of A

Expected Format of Translated IP Header

IPv6 Header:

- Version	= 6	
- Tclass	= IPv4 ToS (might be 0)	
- Flow Label	= 0	
- Payload Len	= IPv4 length - IPv4 header length (20) - IPv4 options (if present) + 8 (fragment header length)	
- Next hdr	= Fragment Header (44)	
- Hop Limit	= IPv4 TTL - 1	
- Src Addr	= ::ffff:[IPv4 src addr]	(mapped address)
- Dst Addr	= ::ffff:0:[IPv4 dst addr]	(translated address)

Fragment Extension Header:

- Next Header	= IPv4 protocol field
- Fragment Offset	= IPv4 Fragment Offset field
- M flag	= IPv4 More Fragments flag
- ID	= low-order 16 bits from IPv4 ID field

Testing:

1. Send Test (fragmented) packet from B to A

2. Verify header translation
   ➔ Verify ID, flags, fragment offset are correctly mapped

➔ Verify IPv4 Checksum is correctly recomputed (at destination).

3. Verify payload is not modified.
   ➔ no reassembly problems at A.

Test b2: Packet Length exceeds IPv6 MTU (1280)

Test Packet Structure

- UDP/IPv4 packet
- Packet IS fragmented (DF=0)
- Pkt length > 1280
- No options
- TTL > 1
- src addr = IPv4 address of B
- dst addr = allocated IPv4 address of A

Testing:

1. Send test packet from B to A. Set DF = 0, Packet Length > IPv6 MTU

2. Verify IPv6 packets on output
   - ➔ IPv4 packet is fragmented prior to translation
   - ➔ 2 or more IPv6 packets are produced (check ID, flags, offset in fragment header).

3. Verify IPv6 packets length
   - ➔ must not exceed 1280 bytes (each fragment)

4. Verify integrity of IPv6 packets
   - ➔ can be done by checking Fragment Header Ext. on each fragment (Fragment offset, M flag, ID)
   - ➔ can be done by comparing payload on final destination (Requires IPv6 reassembly already tested)

c. ICMPv4 -> ICMPv6

Application Requirements:

- Need IPv4 ICMP
- ICMP implementation must be able to send informational as well as error ICMP messages.
- Need 3rd station (D) in IPv4 network to induce ICMP error messages in B
  - IPv4 address of D = allocated IPv4 address of A
  - Because of routing, replies from B go to NAT box, not to D.
- IPv4 Gateway capacity may be needed (have B react like an IPv4 router)

NAT-PT Setup: (c.f. section 3.2.1.a)
Initialization: (c.f. section 3.2.1.a)

Expected format of translated packet (IP + ICMP)

- IP Header Translation: (c.f. section 3.2.1.a1)
- ICMP Header Translation

  - type = must be translated (see below)
  - code = must be translated (see below)
  - checksum = must be computed based on pseudo-header and type/code change.

Test c1: ICMP Informational Messages

Test Packet Structure

- ICMPv4/IPv4 packet
- TTL > 1
- src address = IPv4 address of B
- dst address = allocated IPv4 address of A
- test of several type/code for ICMP.

Testing:

1. Send ICMP echo request from B to A

2. Verify ICMPv4 translated packet
   - ➔ Type & Code translated
     - echo request: IPv4 Type = 8, IPv6 Type= 128
     - echo reply: IPv4 Type = 0, IPv6 Type= 129
   - ➔ ICMP checksum recomputed

3. Verify reception on A
   ➔ A must send ICMPv6 echo reply to [NAT prefix]::[IPv4 addr of B]

4. Send ICMP Router advertisement/solicitation from B to A (types 9,10)
   ➔ verify packet is silently dropped

5. Send ICMP Information request/reply from B to A (types 17,18)
   ➔ verify packet is silently dropped

Test c2: ICMP Error Messages

Test Packet Structure

- ICMPv6/IPv6 packet
- TTL > 1
- src address = IPv4 address of B
- dst address = allocated IPv4 address of A
- test of several type/code for ICMP errors.

Expected Behavior:

- Type and code must be translated according to table below.
- checksum needs to be recomputed
- embedded header must be translated (cf. section 3.2.2.a1)
  (ICMP packets contain the IP header of packet that originated the error)

ICMPv4 error type,code	ICMPv6 translated type,code
3,0	1,0
3,1	1,0
3,2	4,1
3,3	1,4
3,4	2,0
3,5	1,0
3,6	1,0
3,7	1,0
3,8	1,0
3,9	1,1
3,10	1,1
3,11	3,11
3,12	1,0

Testing:

1. Send IPv4 packet from D to B
   - B activated as router
   - pkt dst address unreachable from B
   - ICMP "network unreachable" is generated (3,0)

2. Verify pkt translation
   - ICMPv4 type/code are translated according to table above

3. Send IPv6 packet from D to B
   - pkt length is too big, DF=1
   - ICMP "fragmentation needed" is generated (3,4)

4. Verify pkt translation (same as 2)

5. Send IPv6 packet from C to A
   - TTL = 0
   - ICMP "Time Exceeded" is generated (3,11)

6. Verify pkt translation (same as 2)

d. Translation of IPv4 UDP packets

Application Requirements: (c.f. section 3.2.1.a)
NAT-PT Setup: (c.f. section 3.2.1.a)
Initialization: (c.f. section 3.2.1.a)

Expected Behavior:

- This test concerns UDP packets whose checksum = 0.
- Packet can be translated only if it is not fragmented.

Test d1: Translation of IPv4 UDP packets (not fragmented)

Test Packet Structure

- UDP/IPv4 packet
- UDP checksum = 0
- Packet IS NOT fragmented (DF=1)
- src addr = IPv4 address of B
- dst addr = allocated IPv4 address of A

Testing:

1. Send test packet from B to A, UDP checksum = 0
   ➔ packet must be translated

2. Verify UDP header on translated IPv6 packet
   ➔ A Valid checksum must have been added by NAT box
   ➔ can be verified at final destination

Test d2: Translation of IPv4 UDP fragmented packets

Test Packet Structure

- UDP/IPv4 packet
- UDP checksum = 0
- Packet IS fragmented (DF=0)
- src addr = IPv4 address of B
- dst addr = allocated IPv4 address of A

Testing:

1. Send test (fragmented) packet from B to A. Set checksum = 0

2. Verify that:
   - ➔ first fragment is discarded, state concerning addresses and ports is generated
   - ➔ following fragments are silently discarded

3. Verify what happens when first fragment is missing (no state!)

4. References

- RFC2529, Transmission of IPv6 over IPv4 Domains without Explicit Tunnels, B. Carpenter, C Jung, March 1999, PROPOSED STANDARD.
- RFC2765, Stateless IP/ICMP Translation Algorithm (SIIT), E. Nordmark, February 2000, PROPOSED STANDARD
- RFC2766, Network Address Translation - Protocol Translation (NAT-PT) G. Tsirtsis, P. Srisuresh, February 2000, PROPOSED STANDARD
- RFC2767, Dual Stack Hosts using the Bump-In-the-Stack Technique (BIS), K. Tsuchiya, H. Higuchi, Y. Atarashi, February 2000, INFORMATIONAL
- RFC2893, Dual Stack, Configured tunneling of IPv6 over IPv4, IPv4-compatible IPv6 addresses, Automatic tunneling of IPv6 over IPv4, R. Gilligan, E. Nordmark, August 2000, PROPOSED STANDARD
- RFC3053, IPv6 Tunnel Broker, A. Durand, P. Fasano, I. Guardini, D. Lento, January 2001, INFORMATIONAL
- RFC3056, Connection of IPv6 Domains via IPv4 Clouds, B. Carpenter, K. Moore, February 2001, PROPOSED STANDARD
- RFC3089, A SOCKS-based IPv6/IPv4 Gateway Mechanism, H. Kitamura, April 2001, INFORMATIONAL
- RFC3142, An IPv6-to-IPv4 Transport Relay Translator , J. Hagino, K. Yamamoto, June 2001, INFORMATIONAL
- draft-ietf-ngtrans-dstm-08.txt, Dual Stack Transition Mechanism (DSTM), Jim Bound, Laurent Toutain, Octavio Medina, Francis Dupont, Hossam Afifi, Alain Durand