6TALK : IPv6 Transition Solution

Joo-Chul Lee, ETRI
April.16~17. 2002, ICAT’02
Contents Outline

- What is 6TALK?
- Why 6TALK?
- 6TALK Scenarios & Solution
- 6TALK Implementations
- 6TALK Test Environments
- Summary & Future Work
What is 6TALK?
What is 6TALK?
Why 6TALK?
Why 6TALK?
6TALK Scenarios & Solution
6TALK Scenarios & Solution
6TALK Implementations
6TALK Implementations
6TALK Test Environments
6TALK Test Environments
Summary & Future Work
Summary & Future Work
What is 6TALK?

6TALK?
- IPv6 TransLator of Krv6
- “Please talk with IPv6 ~”

6TALK history
- Main solution and Strategy of KRv6 project (www.krv6.net)
- IPv4 Web server - www.6talk.net
- IPv6 Web server - www.lab.6talk.net

6TALK goal
- Smart Transition Toolbox Supporting IPv4/IPv6 Inter-working
- Enables an IPv6 island to connect the IPv4 Internet and other IPv6 networks seamlessly in initial IPv6 transition period
What is 6TALK?

Why 6TALK?

6TALK Scenarios & Solution

6TALK Implementations

6TALK Test Environments

Summary & Future Work
Why 6TALK?

- Initial IPv6 Transition Period
- Emerging IPv6 islands
  - Incremental deployment on existing network
    - Corporate networks
  - Large scale deployment of new infrastructure
    - Mobile 3G networks
- Facts of Traffic in IPv6 islands
  - IPv6 is NOT backwards compatible with IPv4
  - Most popular servers are running IPv4 today
  - Don’t want isolation
- Requirement of IPv6 Traffic
  - Seamless Routing and Forwarding
    - IPv4 Translation
    - IPv6 Tunneling
What is 6TALK?
Why 6TALK?
☞ 6TALK Scenarios & Solution
6TALK Implementations
6TALK Test Environments
Summary & Future Work
6TALK Transition – NATPT/SIIT

Basic Concepts

- NAT-PT/SIIT = Network Address Translation-Protocol Translation/Stateless IP, ICMP Translation
- Communication between IPv6 only and IPv4 only hosts.
- NAT (use NAT mechanism for assignment of IPv4 address) + ProtocolTranslator (use SIIT mechanism)
- IPv4/IPv6 address translation and keeping state during the time of the session

How does it work?

- Address Translation
  - Use IPv4 address pool, and maintain mapping table of IPv4/IPv6 address
  - IPv4 ↔ IPv6 Protocol Translation
    - Use SIIT mechanism
    - Provide Header translation rule between IPv4 ↔ IPv6
6TALK Transition – NATPT/SIIT

How does it work? - continued

ALG

- ALG = Application Layer Gateway
- Application specific agent
- Some application carry IP address in payloads. But NAT-PT doesn’t snoop payload of packet. ALG resolve this problem

DNS-ALG

- Query type, address change.

FTP-ALG

- PORT, PASV command.
### 6TALK Transition - NATPT/SIIT

#### SIIT Header translation rule (1/4)
- IPv4 ☢ IPv6

<table>
<thead>
<tr>
<th>Field Name</th>
<th>IPv4</th>
<th>IPv6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>Traffic Class</td>
<td>TOS</td>
<td>0</td>
</tr>
<tr>
<td>Flow Label</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Payload Length</td>
<td></td>
<td>Total Length - Header Length - option Length</td>
</tr>
<tr>
<td>Next Header</td>
<td></td>
<td>IPv4 Protocol Field</td>
</tr>
<tr>
<td>Hop Limit</td>
<td></td>
<td>TTL</td>
</tr>
<tr>
<td>Src Address</td>
<td></td>
<td>Low-order 32bits of IPv6 src address</td>
</tr>
<tr>
<td>Dst Address</td>
<td></td>
<td>Low-order 32bits of IPv6 dst address</td>
</tr>
</tbody>
</table>
### 6TALK Transition – NATPT/SIIT

**SIIT Header translation rule (2/4)**

- ICMPv4 $\rightarrow$ ICMPv6

<table>
<thead>
<tr>
<th>Message</th>
<th>IPv4 Type</th>
<th>IPv4 Code</th>
<th>IPv6 Type</th>
<th>IPv6 Code</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Echo/Echo Reply</td>
<td>8/0</td>
<td></td>
<td>128/129</td>
<td></td>
<td>Recalc Checksum</td>
</tr>
<tr>
<td>Information Request/Reply</td>
<td>15/16</td>
<td></td>
<td></td>
<td></td>
<td>Silently Drop</td>
</tr>
<tr>
<td>Timestamp Request/Reply</td>
<td>13/14</td>
<td></td>
<td></td>
<td></td>
<td>Silently Drop</td>
</tr>
<tr>
<td>Address Mask Request/Reply</td>
<td>17/18</td>
<td></td>
<td></td>
<td></td>
<td>Silently Drop</td>
</tr>
<tr>
<td>ICMP router advertisement</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td>Silently Drop</td>
</tr>
<tr>
<td>ICMP router solicitation</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td>Silently Drop</td>
</tr>
<tr>
<td>Unknown ICMP4 types</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Silently Drop</td>
</tr>
<tr>
<td>Membership Query</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Silently Drop</td>
</tr>
<tr>
<td>Membership Report</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Silently Drop</td>
</tr>
</tbody>
</table>
### 6TALK Transition – NATPT/SIIT

#### SIIT Header translation rule (2/4)

- **ICMPv4 <-> ICMPv6 - continued**

<table>
<thead>
<tr>
<th>Message</th>
<th>ICMPv4</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Type</td>
<td>Code</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Destination Unreachable</strong></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0/1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>5/6,7</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>9/10</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>11/12</td>
<td>1</td>
</tr>
<tr>
<td><strong>Redirect</strong></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td><strong>Source Quench</strong></td>
<td>4</td>
<td></td>
</tr>
<tr>
<td><strong>Time Exceeded</strong></td>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td><strong>Parameter Problem</strong></td>
<td>12</td>
<td>4</td>
</tr>
</tbody>
</table>

**Remarks:**
- **IPv4:**
  - Type: 0/1, 2, 3, 4, 5/6, 7, 8, 9/10, 11/12
  - Code: 0, 1, 2, 3, 4

**IPv6:**
- Type: 0, 1, 2, 3, 4
  - Code: 0, 1, 2, 3, 4

**Remarks:**
- No route to destination
- Packet Too Big
- Communication with Destination administratively prohibited
- Silently Drop
- Code field is same
- Need pointer field update
### 6TALK Transition – NATPT/SIIT

#### SIIT Header translation rule (3/4)
- IPv6 ↔ IPv4

<table>
<thead>
<tr>
<th>Field Name</th>
<th>IPv6</th>
<th>IPv4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>Header Length</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>TOS</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Total Length</td>
<td></td>
<td>IPv6 Payload Length + IPv4 Header Length</td>
</tr>
<tr>
<td>Identification</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Flag</td>
<td></td>
<td>MF:0, DF:1</td>
</tr>
<tr>
<td>Fragment Offset</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>TTL</td>
<td></td>
<td>Hop Limit Field</td>
</tr>
<tr>
<td>Protocol</td>
<td></td>
<td>Next Header Field</td>
</tr>
<tr>
<td>Header Checksum</td>
<td></td>
<td>Recalc</td>
</tr>
<tr>
<td>Src Address</td>
<td></td>
<td>Low-order 32bits of IPv6 src Address</td>
</tr>
<tr>
<td>Dst Address</td>
<td></td>
<td>Low-order 32bits of IPv6 dst Address</td>
</tr>
</tbody>
</table>
# 6TALK Transition – NATPT/SIIT

## SIIT Header translation rule (4/4)
- **ICMPv6 ↔ ICMPv4**

<table>
<thead>
<tr>
<th>Message</th>
<th>IPv6</th>
<th>Remarks</th>
<th>IPv4</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ICMPv6 Informational Messages</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Echo/Echo Reply</td>
<td>128/129</td>
<td>0/8</td>
<td>Recalc Checksum</td>
<td></td>
</tr>
<tr>
<td>MLD Multicast Listener Query/Report/Done</td>
<td>130/131/132</td>
<td></td>
<td>0/8</td>
<td>Silently Drop</td>
</tr>
<tr>
<td>Neighbor Discover messages</td>
<td>133–137</td>
<td></td>
<td>Silently Drop</td>
<td></td>
</tr>
<tr>
<td>Unknown informational messages</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ICMPv6 Error messages</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destination Unreachable</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Host unreachable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>10</td>
<td>Communication with Destination host administratively prohibited</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>1</td>
<td>Host unreachable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>1</td>
<td>Host unreachable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>3</td>
<td>Port unreachable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>Port unreachable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>Destination Unreachable</td>
</tr>
<tr>
<td>Packet Too Big</td>
<td>2</td>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>
### 6TALK Transition – NATPT/SIIT

#### SIIT Header translation rule (4/4)
- ICMPv6 ↔ ICMPv4 - continued

<table>
<thead>
<tr>
<th>Message</th>
<th>Type</th>
<th>Code</th>
<th>IPv6 Remarks</th>
<th>IPv4 Type</th>
<th>Code</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time Exceeded</td>
<td>3</td>
<td></td>
<td></td>
<td>11</td>
<td></td>
<td>no change</td>
</tr>
<tr>
<td>Parameter Problem</td>
<td>4</td>
<td>1</td>
<td>Unrecognized Next Header type</td>
<td>3</td>
<td>2</td>
<td>ICMPv4 protocol unreachable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0/2</td>
<td>encountered</td>
<td>12</td>
<td>0</td>
<td>Parameter problem</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Erroneous header field encountered / Unrecognized IPv6 option encountered</td>
<td></td>
<td></td>
<td>Silently Drop</td>
</tr>
<tr>
<td>Unknown Error Messages</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
6TALK Transition – NATPT/SIIT

**Basic NAT-PT Operation (IPv6 ↔ IPv4)**
- src IPv6 host knows IPv4 address of dest.

1. Assign new IPv4 address 120.130.26.10 for src IPv6 address from address pool.
2. IPv6 ↔ IPv4 Header Translation

---

**Outbound traffic**

- **IPv6 Host**: linux.ipv6.or.kr (2001:230::1)
- **IPv4 Host**: www.etri.re.kr (129.254.19.28)

**Address pool**: 120.130.26/24
- **Domain prefix**: 2001:230:d:fffd::/96

**IPv6 Address Mapping table**

<table>
<thead>
<tr>
<th>IPv6 Address</th>
<th>IPv4 Address</th>
<th>Etc parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>2001:230::1</td>
<td>120.130.26.10</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

---

**NAT-PT’s Address Mapping table**

- **SA=2001:230::1**
- **DA=2001:230:d:fffd::129.254.19.28**
- **SA=120.130.26.10**
- **DA=129.254.19.28**
6TALK Transition – NATPT/SIIT

NAT-PT with DNS-ALG Operation (IPv6 ⇆ IPv4)
- IPv6 DNS already knows IPv4 DNS’s address, and IPv4 DNS also knows IPv4 address for IPv6 DNS.
- NAT-PT already has one-to-one mapping of IPv6 DNS’s address and an address of IPv4 pool.

1. DNS query ? www.etri.re.kr
3. Addr of www.etri.re.kr ? (A)
4. query response : 129.254.19.28 (A)

IPv6 DNS already knows IPv4 DNS’s address, and IPv4 DNS also knows IPv4 address for IPv6 DNS.
NAT-PT already has one-to-one mapping of IPv6 DNS’s address and an address of IPv4 pool.

1. Assign new IPv4 address 120.130.26.9 for src IPv6 address from address pool.
2. IPv6-IPv4 Header Translation

IPv6/IPv4 DNS mapping entry
6TALK Transition – NATPT/SIIT

NAT-PT with DNS-ALG Operation (IPv4 ↔ IPv6)
IPv4 DNS already knows IPv4 address which assigned for IPv6 DNS from address pool.
NAT-PT already has one-to-one mapping of IPv6 DNS’s address and an address of IPv4 pool

---

IPv6 DNS

<table>
<thead>
<tr>
<th>IPv6 Host</th>
<th>NAT-PT</th>
<th>IPv4 DNS</th>
</tr>
</thead>
</table>

---

IPv6 DNS

<table>
<thead>
<tr>
<th>IPv6 Host</th>
<th>NAT-PT</th>
<th>IPv4 DNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addr of linux.ipv6.or.kr ? (AAAA/A6)</td>
<td></td>
<td>Addr of linux.ipv6.or.kr ? (A)</td>
</tr>
</tbody>
</table>

---

IPv6 DNS

<table>
<thead>
<tr>
<th>IPv6 Host</th>
<th>NAT-PT</th>
<th>IPv4 DNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>query response : 2001:230::1 (AAAA/A6)</td>
<td></td>
<td>query response : 120.130.26.9 (A)</td>
</tr>
</tbody>
</table>

---

IPv6 DNS

<table>
<thead>
<tr>
<th>IPv6 Host</th>
<th>NAT-PT</th>
<th>IPv4 DNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS query ? linux.ipv6.or.kr</td>
<td></td>
<td>DNS query ? linux.ipv6.or.kr</td>
</tr>
</tbody>
</table>

---

IPv6 DNS

<table>
<thead>
<tr>
<th>IPv6 Host</th>
<th>NAT-PT</th>
<th>IPv4 DNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assign new IPv4 address 120.130.26.9 for src IPv6 address from address pool.</td>
<td></td>
<td>Assign new IPv4 address 120.130.26.9 for src IPv6 address from address pool.</td>
</tr>
</tbody>
</table>

---

IPv6 DNS

<table>
<thead>
<tr>
<th>IPv6 Host</th>
<th>NAT-PT</th>
<th>IPv4 DNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPv6/IPv4 Header Translation</td>
<td></td>
<td>IPv6/IPv4 Header Translation</td>
</tr>
</tbody>
</table>

---

IPv6 DNS

<table>
<thead>
<tr>
<th>IPv6 Host</th>
<th>NAT-PT</th>
<th>IPv4 DNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPv6 Address</td>
<td>IPv4 Address</td>
<td>Etc parameter</td>
</tr>
<tr>
<td>2001:230::1</td>
<td>120.130.26.2</td>
<td>...</td>
</tr>
<tr>
<td>2001:230::1</td>
<td>120.130.26.9</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Basic Concepts
- DSTM = **Dual Stack Transition Mechanism**
- When dual stack nodes in IPv6 network need to communicate with existing IPv4 nodes.
- DSTM node which want to communicate IPv4 node gets temporary IPv4 address. And use IPv4 in IPv6 tunneling.
- DSTM = Assignment mechanism of global IPv4 address to IPv6/IPv4 nodes + Tunnel mechanism using Dynamic Tunneling Interface (DTI)

How does it work?
- DSTM host has dual stack, but IPv4 is configured when it is needed.
How does it work? - continued

Major Components of DSTM

- **DHCPv6 server**
  - Allocate global IPv4 address, and TEP address.

- **DSTM daemon**
  - Get IPv4 address and TEP address from DHCPv6 server

- **DTI**
  - **DTI = Dynamic Tunneling Interface**
  - An interface encapsulating IPv4 packets into IPv6 packets.

- **TEP**
  - **TEP = Tunnel End Point**
  - Destination of IPv6 packet that contain IPv4 packet.
  - Generally this will be a dual stack border router.
6TALK Transition – DSTM

How does it work? - continued

DSTM client/server example

- X6 (dualstack node) ⚫ Y6/Y4 (TEP) ⚫ Z4 (IPv4 node)
- X resolved Z as IPv4 address “Z4”
- Application on X sends first IPv4 packet and it arrives DTI
- X get temporary IPv4 address and TEP from a DHCPv6 server
- the DTI sends IPv6 packet encapsulating IPv4 packet to the TEP
- Y (TEP) caches association between IPv4 and IPv6 addresses of X, and sends IPv4 packet to Z
- Reply procedure is reverse process of this.
6TALK Transition – DSTM

How does it work? - continued

1. DNS Query for ‘A’ RR

2. Get IPv4 Address, TEP address

3. DTI sends IPv6 packet which contains IPv4 packet to TEP

Border Router
(cache the association between IPv4 and IPv6 address of DSTM host)
6TALK Transition – 6to4

**Basic Concepts**

- To Interconnect isolated IPv6 domains in an IPv4 world
- **Automatic tunneling** which don’t use IPv4-compatible address.
- IPv4 endpoints of the tunnel are identified in the prefix of the IPv6 domain **6to4 Prefix**

**How does it work?**

**6to4 Prefix**

- 6to4 address
  - 2002
  - IPv4
  - 48 bits
  - ISPv4 assigned
  - pre-defined

- 6to4 IPv6 address
  - 2002
  - IPv4
  - 48 bits
  - ISPv4 assigned
  - SLA
  - auto-configured

- Interface ID
  - 64 bits
  - 23/47

- This is IPv4 Tunneling End Point address.
- This address is assigned on interface of 6to4 router which connects 6to4 router to IPv4 network

All 6to4 hosts have 6to4 IPv6 address including 6to4 prefix
How does it work? - continued

- **6to4 router**
  - A router between wide-area IPv4 network and IPv6 site.
  - Encapsulation and decapsulation are occur in 6to4 router.

- **6to4 host**
  - A host which has 6to4 address

- **Relay router**
  - A 6to4 router which support transit routing between 6to4 address and native IPv6 address
  - Relay router has at least 1 logical 6to4 pseudo interface and at least 1 IPv6 interface.
  - It advertises 6to4 prefix and native IPv6 prefix.
6TALK Transition – 6to4

How does it work? – continued

- Sending packet scenario
  - An 6to4 host A gets 6to4 address of dest host B (by DNS or something..)
  - A sends packets to default router (6to4 router A`)
  - 6to4 router A` encapsulates packet by using dest host B’s 6to4 address and forward it to IPv4 network
  - 6to4 border router B` receives and decapsualtes packet from A`
  - B` forward packet to B
6TALK Transition – 6to4

Sending & Encapsulation Rule

IPv4 Header

IPv6 Header

payload

payload

6to4 host

6to4 router

IPv4 network

6to4 host

6to4 router

6to4 site

6to4 host

6to4 host

6to4 site

6to4 host

6to4 site

……

src: 2002:aabb:ccdd:...
dst: 2001:AABB:CCDD:...

……

src: aa.bb.cc.dd
dst: AA.BB.CC.DD

……

src: 2002:aabb:ccdd:...
dst: 2002:AABB:CCDD:...
6TALK Scenarios(1/3)

IPv6 island
(no ipv4 routing)

IPv4 Internet

6TALK box
- NATPT - enable
- (DSTM - disable)

Decision - 6TALK box

All of IPv6 packets which have NAT-PT dummy prefix are translated into IPv4 packets, and vice versa

Pros & Cons
- Easy to deploy (no change of IPv6 hosts)
- Does not support end-to-end connectivity and have scalability problem (# of flow)
6TALK Scenarios (2/3)

IPv6 island (no ipv4 routing)

6TALK box
- (NATPT - disable)
- DSTM - enable
- Decision - IPv6 host
  - If DNS query result is an IPv4 address (A) then select DSTM mechanism

Pros & Cons
- Support End-to-end connectivity
- Need extra components (DHCPv6, DSTM client daemon,...)
6TALK Scenarios (3/3)

IPv6 island (no ipv6 routing)

IPv4 Internet

IPv6 networks

- 6TALK box
  - 6to4 or configured tunnel configuration
- Decision - IPv6 host
  - If DNS query result is a **6to4 address (AAAA)**, then select **6to4 mechanism**
  - If DNS query result is **other IPv6 address**, then select **normal IPv6 routing**
- Pros & Cons
  - VPNv6 support
6TALK Solutions

6TALK
- Smart Toolbox for smooth migration towards IPv6
- NAT-PT/SIIT, DSTM, 6to4, Configured Tunnel, etc.

6TALK - IPv4/IPv6 Inter-working Solutions
- Scenario 1 & Scenario 3, or
- Scenario 2 & Scenario 3

How can network administrators choose proper mechanisms?
- # of IPv6 users (scalability), network transition period, IPv6 services type, etc.
- DNS query information is important!
  - Enhanced DNS ALGs
What is 6TALK?
Why 6TALK?
6TALK Scenarios & Solution
☞ 6TALK Implementations
6TALK Test Environments
Summary & Future Work
6TALK Specifications

Transition mechanisms

- NAT-PT
  - Implement NAT-PT (RFC2766) and SIIT (RFC2765)
  - It also supports NAPT-PT
  - It is based on Linux Netfilter framework

- DSTM & DSTM extensions
  - Will implement DSTM (draft-ietf-ngtrans-dstm-05) and DSTM extension (draft-ietf-ngtrans-dstm-ext-00)
  - Plan to implement in 2002

- IPv6 over IPv4 Tunneling (including 6to4)
  - Already implemented in Linux

Operation environment

- Linux Kernel 2.4.8 and Embedded Linux using MPC8260 H/W
6TALK Function Modules

- User Interfaces
- DNS-ALG
- socket
- TCP/UDP
- IPv6/IPv4 Transition Manager
- IPv6
- NA(P)T - PT
- IPv6/IPv4 mapping table
- IPv4 Address Pool
- IPv4
- SIIT
- NIC
- NIC
- IPv6/IPv4 mapping table

- DSTM
- DSTM-EXT

Years:
- 2001
- 2002
- 2003

Additional NGtrans
## 6TALK Hardware

<table>
<thead>
<tr>
<th>Item</th>
<th>Spec</th>
<th>Etc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>MPC8260Z U200A</td>
<td></td>
</tr>
<tr>
<td>RAM</td>
<td>SDRAM 64M</td>
<td></td>
</tr>
<tr>
<td>Flash Memory</td>
<td>1M</td>
<td>For PPC Boot</td>
</tr>
<tr>
<td>Flash Memory</td>
<td>16M</td>
<td>For Embedded Linux</td>
</tr>
<tr>
<td>Serial Port</td>
<td>2ports</td>
<td></td>
</tr>
<tr>
<td>10M Ethernet</td>
<td>4ports</td>
<td></td>
</tr>
<tr>
<td>Fast Ethernet</td>
<td>2ports</td>
<td></td>
</tr>
</tbody>
</table>
6TALK User Interfaces

Console mode (text)

Web CGI mode
6TALK Implementation (1)

NAT-PT Implementation

- Linux 2.4.x Netfilter framework Base
- Kernel netfilter framework
  - ip(6)table structure
    - List of packet matching & manipulation rule for each netfilter application (NAT, NAT-PT, packet mangle, packet filter, etc.)
  - connection tracking structure
    - Maintain status of each connection.
- User interface
  - ip(6)tables commands
6TALK Implementation (2)

Structure of NAT-PT in Netfilter

- IPv6 connection tracking
- packet translation module using SIIT
- NATPT ip6table
- IPv6/IPv4 mapping table
- match & manip rule

ip6tables user interface

IPv6 stack

DNS, FTP... ALGs

IPv4 connection tracking

IPv4 stack

Network Interface

Kernel space
6TALK Implementation (3)

Structure of NAT-PT user interface

- Bash Shell
- 6TALK Application for Interface
- Linux Kernel
- NAT-PT module
- Buffer
  
- Storage (Flash memory)
- PC platform
- MPC8260 platform

6TALK Operator

6TALK Text Interface

6TALK Web Interface

6TALK::IP>

6TALK::NATPT>

/etc/cfg_natpt

write

read

write

read

CGI
6TALK Implementation Issues (1/2)

**Fragmentation issues**
- MTU of IPv4 network and IPv6 network might be different. So we can’t translate each fragmented packet directly.
  - defragment first and fragment again

**ALG (Application Level Gateway)**
- If upper layer protocol’s payload includes ip address, we should translate that.
  - 6TALK supports DNS-ALG
6TALK Implementation Issues (2/2)

- **ICMP error message handling**
  - Also translate ip header embedded in icmp error message

- **User interface extension**
  - Add shared library for 6TALK
What is 6TALK?
Why 6TALK?
6TALK Scenarios & Solution
6TALK Implementations
☞ 6TALK Test Environments
Summary & Future Work
**6TALK Test Environments**

- **prefix**

- **IPv4 pool**
  - 129.254.179.130 – 129.254.179.142

- **DNSv6 static mapping**
  - 129.254.179.129

- **IPv4 pool**
  - 129.254.165.122

- **IPv6 Client**
  - mkshin.lab.6talk.net
  - 2001:230:0:5:a00:46ff:fe0d:b0b0/64
  - (Linux2.4.8)

- **WWWv6 server**
  - www.lab.6talk.net
  - 2001:230:0:5.201:2ff:fe96:ccab/64
  - (Linux2.4.8)

- **IPv6 HUB1**
  - 6TALK-C3620
  - Ethernet1/1
  - 2001:230:0:3:A00:3EFF:FE42:1559/64

- **IPv6 HUB2**
  - IPv6 native nodes
  - Ethernet1/2

- **IPv6 Client**
  - bsd.lab.6talk.net
  - 2001:230:0:5.2e0:29ff:fe1f:7fde/64
  - (Freebsd3.2)

- **IPv6 Client**
  - www.lab.6talk.net
  - 2001:230:0:5.201:2ff:fe96:ccab/64
  - (Linux2.4.8)

- **IPv6 HUB1**
  - 6TALK-C3620
  - Ethernet1/1
  - 2001:230:0:3:A00:3EFF:FE42:1559/64

- **ETRI IPv4 Backbone Router**
  - 129.254.165.1

- **6Bone IPv6 Router - Cisco7500**
  - ipv6-gw.ipv6.or.kr

- **6Bone IPv6 Router - Cisco3620**
  - Ethernet1/1
  - Ethernet1/2

- **IPv4 HUB**
  - 6TALK – v4 Internet

- **IPv6 Client**
  - mkshin.lab.6talk.net
  - 2001:230:0:5:a00:46ff:fe0d:b0b0/64
  - (Linux2.4.8)
Test

[root@Runic ~]# ping6 aaaa:bbbb:cccc:dddd:eeeee:ffff:403a:4cb3 -s 10000
10008 bytes from aaaa:bbbb:cccc:dddd:eeeee:ffff:403a:4cb3: icmp_seq=7 hups=233 time=324.423 msec

What is 6TALK?
Why 6TALK?
6TALK Scenarios & Solution
6TALK Implementations
6TALK Test Environments
☞ Summary & Future Work
Summary

6TALK : Smart Transition Toolbox supporting Inter-working for IPv6-IPv4 and/or IPv6-IPv4-IPv6 seamlessly
Summary & Future Work

General Platform for IPv6 Transition Mechanism Implementation
  - Open architecture
    - Linux base
  - Sets of IPv6 transition mechanisms
    - IPv4/IPv6 Translation: NAT-PT / SIIT (Now)
    - Basic Tunneling: 6to4 and configured tunnel (Now)
    - DSTM and DSTM extensions (This year)
    - Combinations of transition tools (2003)
      - DSTM+SIIT, 6to4+ISATAP, etc.
  - Optimization for performance
  - Conformance test (i.e. by using TAHI)