A SOCKS-based IPv6/IPv4 Translator Architecture

<draft-kitamura-socks-ipv6-trans-arch-00.txt>

Hiroshi Kitamura NEC Corporation

SOCKS64: An IPv4-IPv6 intercommunication gateway using SOCKS5 protocol

<draft-jinzaki-socks64-00.txt>

Shinji Kobayashi Fujitsu Laboratories LTD.

SOCKSv5-based IPv4-IPv6 Transition Mechanisms
History and Status

- RFC1928 "SOCKS Protocol V5," April 1996
- Fujitsu Labs. first introduced the idea at NGTRANS, Dec.1997
  http://www.ietf.org/proceedings/97dec/97dec-final-73.htm
- NEC proposed <draft-kitamura-socks-ipv6-00.txt>
  "SOCKSv5 Protocol Extensions for IPv6/IPv4 Communication Environment," at STP BOF, August 1998

Two Independent Implementations

Fujitsu Labs. distributes an implementation since July 1998
ftp://ftp.kame.net/pub/kame/misc/socks64-v10r3-980623.tgz

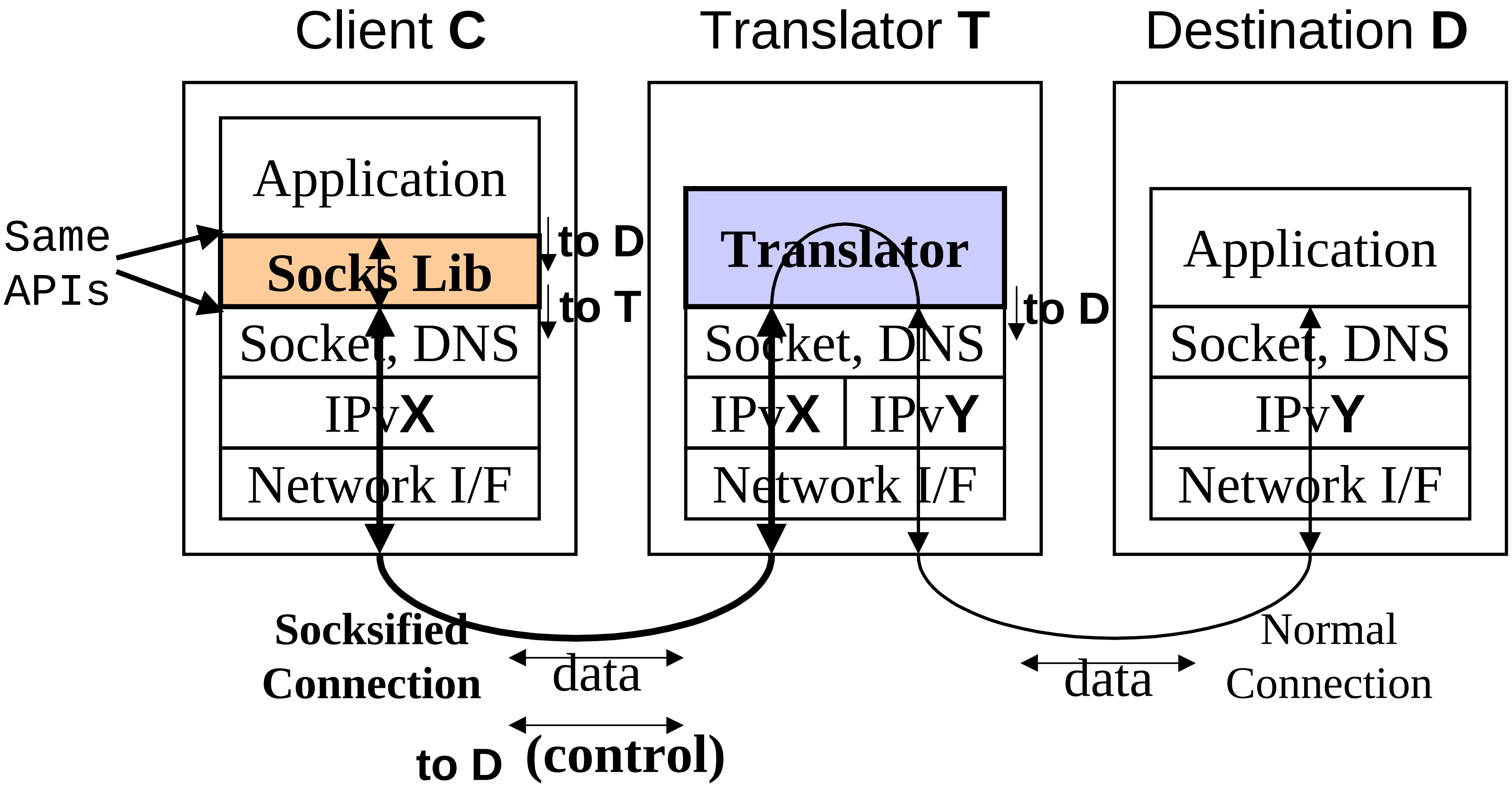
NEC has finished an implementation and is planning to open its source in January 1999 http://www.socks.nec.com/

Contents

- Basic SOCKS-based Translator Mechanism
- DNS Name Resolving Procedure
  - ***DNS Name Resolving Delegation*** and ***Address Mapping***
- Advanced SOCKS-based Translator Mechanism
  (**Multiple Chained Relay Mechanism**)
- Characteristics and Constraints

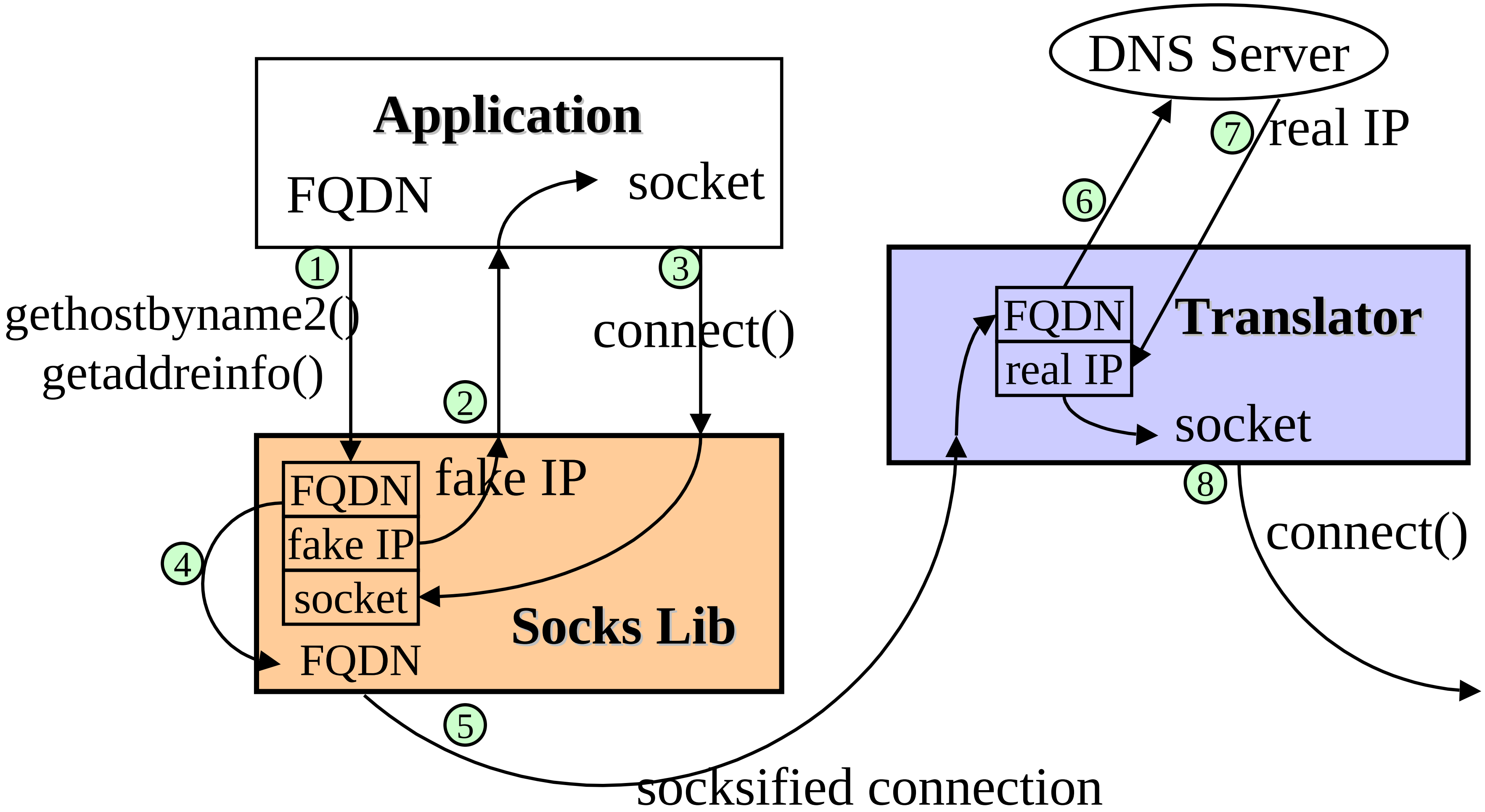
- Development Status
  - Fujitsu Labs. Implementation
- Field Trial Results
  - WIDE Project Trials,   Fujitsu to 6bone connection
- Comparison and Summary

Basic Translator mechanism



DNS Name Resolving Procedure

(DNS ***Name Resolving Delegation*** and ***Address Mapping***)



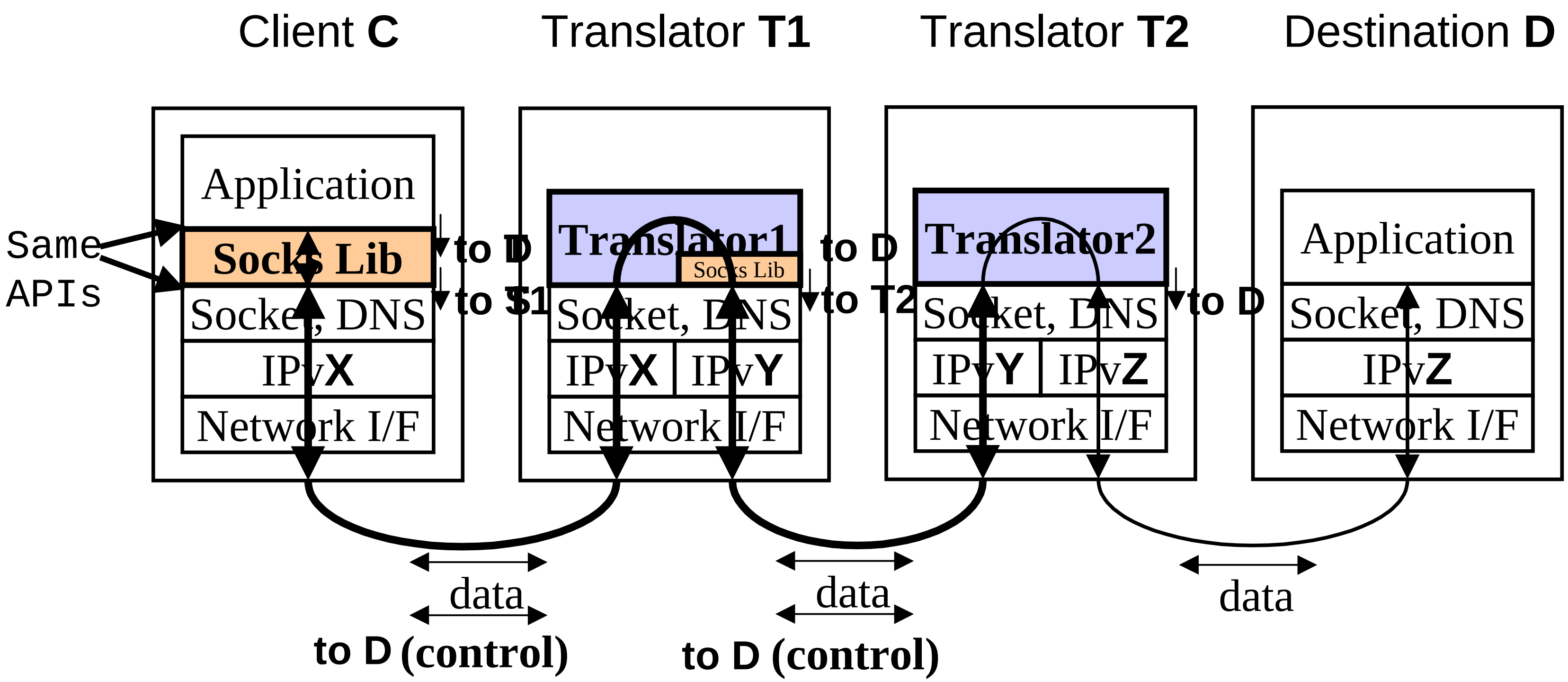
DNS Name Resolving Procedure

(DNS ***Name Resolving Delegation*** and ***Address Mapping***)

1. Try DNS name resolving with “FQDN”
2. Receive “fake IP”
3. Create connection with “socket” (includes “fake IP”)
4. Pick up registered “FQDN” from the mapping table
5. Send “FQDN” to the Translator on the socksified connection
6. DNS name resolve by the normal DNS server
7. Receive “real IP”
8. Create connection to the Destination with “socket” (includes “real IP”)

Current SOCKSv5 protocol <u>*dose not have a dedicated handshake*</u>
for the **DNS name resolving delegation**.
<draft-kitamura-socks-ipv6-00.txt> proposed this extension.

Advanced Translator mechanism
(Multiple Chained Relay)



Socksified Connection

Socksified Connection

Normal Connection

Characteristics of Multiple Chained Relay (compared with IP Tunneling technique)

- **No Fragmentation vulnerability**
  - IP tunneling technique (en/decapsulation) change the packet size.
    It has Fragmentation vulnerability
  - SOCKS mechanism dose **NOT** change the packet size.
- **No Hop limit (metric number) problem**
  - The tunneling technique creates one virtual connections over the dynamically routed and configured networks.
    Real hop limit (metric number) information is hidden
  - SOCKS mechanism is composed of real connections.
    Real hop limit (metric number) information is **NOT** hidden
- Well-authenticated relay by the native SOCKS methods

Topology Combinations

C === T === D

IPv4 - IPv4

<u>IPv4 - **IPv6**</u>

<u>**IPv6** - IPv4</u>

IPv6 - IPv6

C === T1 === T2 ===D

IPv4 - IPv4 - IPv4

IPv4 - IPv4 - **IPv6**

<u>IPv4 - **IPv6** - IPv4</u>

IPv4 - **IPv6** - **IPv6**

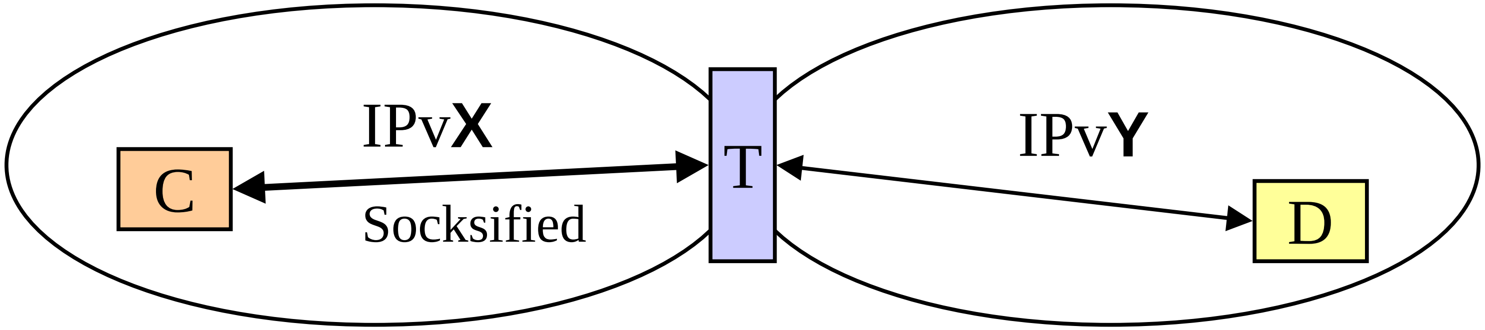
IPv6 - IPv4 - IPv4

<u>**IPv6** - IPv4 - **IPv6**</u>

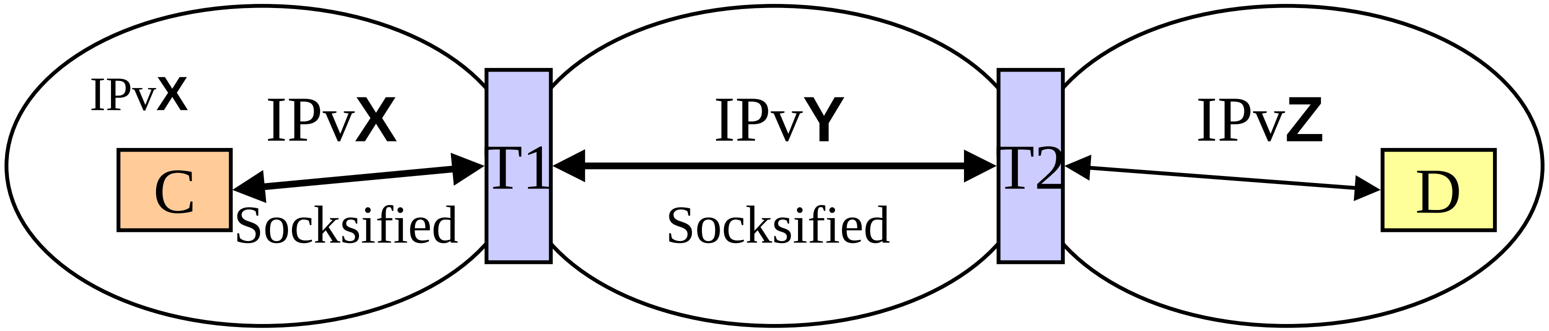
IPv6 - **IPv6** - IPv4

IPv6 - IPv6 - IPv6

Support Topologies



Basic (Neighboring Connection)



Advanced (Multiple Chained Relay Connection)

Implementation History in Fujitsu Labs.

- implemented the prototype on Solaris IPv6 in Aug., 1996
- first introduced the idea at NGTRANS in Dec., 1997

  http://www.ietf.org/proceedings/97dec/97dec-final-73.htm
- distributes an implementation since Jul., 1998

  ftp://ftp.kame.net/pub/kame/misc/socks64-v10r3-980623.tgz
- Internet-Draft in Nov., 1998

  ftp://ftp.ietf.org/internet-drafts/draft-jinzaki-socks64-00.txt

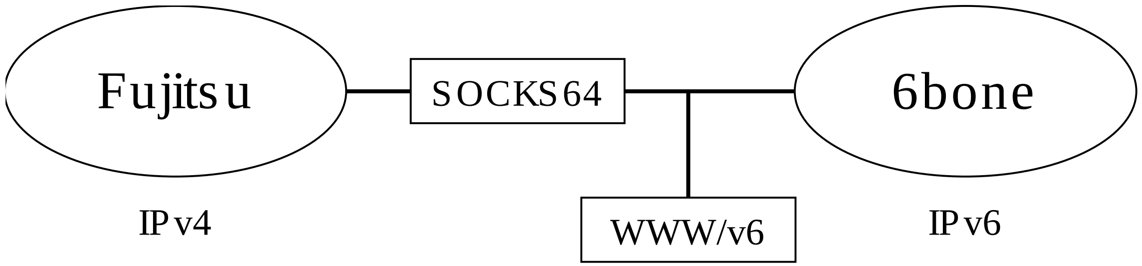
Implementation Status

- SOCKS64: Fujitsu Lab.'s Implementation
- Runs on BSD/OS 3.1 with KAME
- SOCKS5 Client Library (e.g. SocksCap32) can be used without modification
- Freely available

  ftp://ftp.kame.net/pub/kame/misc/socks64-v10r3-980623.tgz
- Support application dependent handling for ftp
  - Convert PORT and LPRT commands
  - Convert PASV and LPSV commands

Field Trial Results (1)

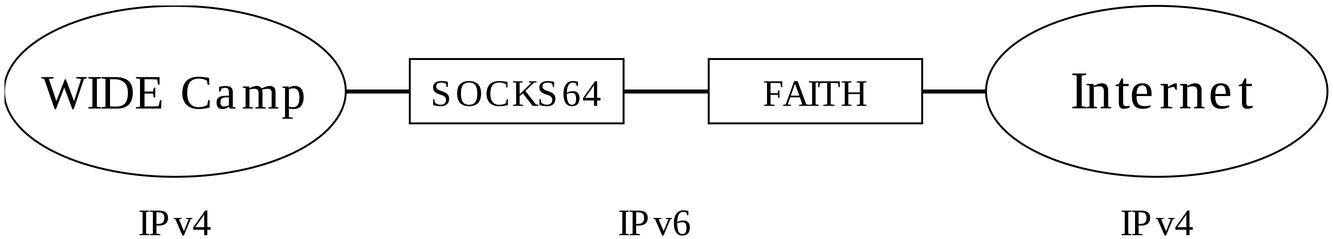
- 6bone - Fujitsu
  - Since Jul. 1998
  - Fujitsu Network is already “socksified”
  - 20 clients
  - telnet, ftp, http



http://www.v6.pds-flab.rwcp.or.jp/ (IPv6 only)

Field Trial Results (2)

- WIDE Camp
  - Sep. 1997, Mar. 1998, Sep. 1998
  - 35 clients
  - telnet, ftp, http, ssh, pop, imap3
  - Cooperation with FAITH IPv6-IPv4 Translator



Comparison

	Implementation Layer	DNS Change	Address Table	Client Library
NAT based	Network	Needed	Needed	NOT Needed
FAITH	Application	Needed	Needed	NOT Needed
SOCKS5 based	Application	NOT Needed	NOT Needed	Needed

- “NAT based” can be fast
- “SOCKS5 based” does not require DNS change nor managing address table
- “SOCKS5 based” requires the client library

Characteristics (1/2)

- **DNS modification is NOT necessary**
  - Address map servers are NOT necessary
  - Global and wide reserved address space is NOT necessary
- Application independent
  - Basically support all applications which use the socket and DNS APIs
  - *Exceptions*: applications which exchange IP address information at the application level. (e.g., ftp PORT command)
- **OS and NIC types independent**
  - Support both UNIX and Windows OSs
  - Not depend on types of physical NICs.
- Only easy socksify procedure is necessary
  - Dynamic link library technique helps the socksification.
- IPv6 new features (e.g., IPSec) can be introduced easily
  - Since relayed two connections are terminated at the Translator

Characteristics (2/2)

- Current existing client SOCKSv5 library can be used
  - In case of the IPv4 -> IPv6 direction translation,
    current existing client SOCKSv5 library can be used without modification.
- Both **TCP** and **UDP** relay translations are possible.
  - Since the SOCKSv5 protocol support both TCP and UDP relays, this architecture can translate not only TCP but also UDP relays.
- Both **IPv4->IPv6** and **IPv6->IPv4** translations are possible
- **Multiple chained relay is possible**.
- Can support FTP (IP address information exchange applications)
  - The Translator has the capability to introduce protocol translation routines.
  - If protocols are known (e.g., ftp), the Translator can support them by introducing special protocol translation routines

Constraints

- Essential constraint
  - getpeername() and getsockname() functions can not provide correct IP address information. Because IPv6 and IPv4 are different protocols. **Port information is correct**.
  - Most applications call them to get port information. From the actual viewpoint, ***this constraint is small***.
- Limitation of the SOCKS mechanism
  - Current SOCKSv5 can not socksify all of tricky applications.
- Fake address dealing constraint
  - The fake address must be dealt as a temporary value in the application.
  - Most applications record FQDN and does not record resolved fake addres From the actual viewpoint, ***this constraint is small***.