Network Working Group
Request for Comments: 3142
Category: Informational

J. Hagino
K. Yamamoto
IIJ Research Laboratory

June 2001

An IPv6-to-IPv4 Transport Relay Translator

Status of this Memo

Copyright Notice

Abstract

The document describes an IPv6-to-IPv4 transport relay translator (TRT). It enables IPv6-only hosts to exchange {TCP,UDP} traffic with IPv4-only hosts. A TRT system, which locates in the middle, translates {TCP,UDP}/IPv6 to {TCP,UDP}/IPv4, or vice versa.

The memo talks about how to implement a TRT system using existing technologies. It does not define any new protocols.

1. Problem domain

When you deploy an IPv6-only network, you still want to gain access to IPv4-only network resources outside, such as IPv4-only web servers. To solve this problem, many IPv6-to-IPv4 translation technologies are proposed, mainly in the IETF ngtrans working group. The memo describes a translator based on the transport relay technique to solve the same problem.

In this memo, we call this kind of translator "TRT" (transport relay translator). A TRT system locates between IPv6-only hosts and IPv4 hosts and translates {TCP,UDP}/IPv6 to {TCP,UDP}/IPv4, vice versa.

Advantages of TRT are as follows:

- TRT is designed to require no extra modification on IPv6-only initiating hosts, nor that on IPv4-only destination hosts. Some other translation mechanisms need extra modifications on IPv6-only initiating hosts, limiting possibility of deployment.

- The IPv6-to-IPv4 header converters have to take care of path MTU and fragmentation issues. However, TRT is free from this problem.

Disadvantages of TRT are as follows:

- TRT supports bidirectional traffic only. The IPv6-to-IPv4 header converters may be able to support other cases, such as unidirectional multicast datagrams.

- TRT needs a stateful TRT system between the communicating peers, just like NAT systems. While it is possible to place multiple TRT systems in a site (see Appendix A), a transport layer connection goes through particular, a single TRT system. The TRT system thus can be considered a single point of failure, again like NAT systems. Some other mechanisms, such as SIIT [Nordmark, 2000], use stateless translator systems which can avoid a single point of failure.

- Special code is necessary to relay NAT-unfriendly protocols. Some of NAT-unfriendly protocols, including IPsec, cannot be used across TRT system.

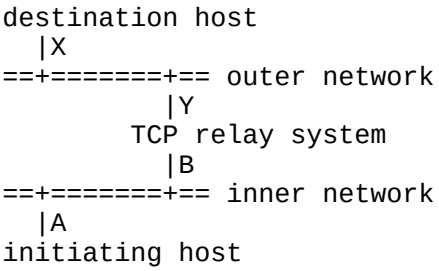
This memo assumes that traffic is initiated by an IPv6-only host destined to an IPv4-only host. The memo can be extended to handle opposite direction, if an appropriate address mapping mechanism is introduced.

2. IPv4-to-IPv4 transport relay

To help understanding of the proposal in the next section, here we describe the transport relay in general. The transport relay technique itself is not new, as it has been used in many of firewall-related products.

2.1. TCP relay

TCP relay systems have been used in firewall-related products. These products are designed to achieve the following goals: (1) disallow forwarding of IP packets across a system, and (2) allow {TCP,UDP} traffic to go through the system indirectly. For example, consider a network constructed like the following diagram. "TCP relay system" in the diagram does not forward IP packet across the inner network to the outer network, vice versa. It only relays TCP traffic on a specific port, from the inner network to the outer network, vice versa. (Note: The diagram has only two subnets, one for inner and one for outer. Actually both sides can be more complex, and there can be as many subnets and routers as you wish.)



When the initiating host (whose IP address is A) tries to make a TCP connection to the destination host (X), TCP packets are routed toward the TCP relay system based on routing decision. The TCP relay system receives and accepts the packets, even though the TCP relay system does not own the destination IP address (X). The TCP relay system pretends to having IP address X, and establishes TCP connection with the initiating host as X. The TCP relay system then makes a another TCP connection from Y to X, and relays traffic from A to X, and the other way around.

Thus, two TCP connections are established in the picture: from A to B (as X), and from Y to X, like below:

TCP/IPv4: the initiating host (A) --> the TCP relay system (as X)
address on IPv4 header: A -> X
TCP/IPv4: the TCP relay system (Y) --> the destination host (X)
address on IPv4 header: Y -> X

The TCP relay system needs to capture some of TCP packets that is not destined to its address. The way to do it is implementation dependent and outside the scope of this memo.

2.2. UDP relay

If you can recognize UDP inbound and outbound traffic pair in some way, UDP relay can be implemented in similar manner as TCP relay. An implementation can recognize UDP traffic pair like NAT systems does, by recording address/port pairs onto an table and managing table entries with timeouts.

3. IPv6-to-IPv4 transport relay translator

We propose a transport relay translator for IPv6-to-IPv4 protocol translation, TRT. In the following description, TRT for TCP is described. TRT for UDP can be implemented in similar manner.

For address mapping, we reserve an IPv6 prefix referred to by C6::/64. C6::/64 should be a part of IPv6 unicast address space

assigned to the site. Routing information must be configured so that packets to C6::/64 are routed toward the TRT system. The following diagram shows the network configuration. The subnet marked as "dummy prefix" does not actually exist. Also, now we assume that the initiating host to be IPv6-only, and the destination host to be IPv4-only.

```
destination host
  |X4
==+=======+== outer network
          |Y4
        TRT system --- dummy prefix (C6::/64)
          |B6
==+=======+== inner network
  |A6
initiating host
```

When the initiating host (whose IPv6 address is A6) wishes to make a connection to the destination host (whose IPv4 address is X4), it needs to make an TCP/IPv6 connection toward C6::X4. For example, if C6::/64 equals to fec0:0:0:1::/64, and X4 equals to 10.1.1.1, the destination address to be used is fec0:0:0:1::10.1.1.1. The packet is routed toward the TRT system, and is captured by it. The TRT system accepts the TCP/IPv6 connection between A6 and C6::X4, and communicate with the initiating host, using TCP/IPv6. Then, the TRT system investigates the lowermost 32bit of the destination address (IPv6 address C6::X4) to get the real IPv4 destination (IPv4 address X4). It makes an TCP/IPv4 connection from Y4 to X4, and forward traffic across the two TCP connections.

There are two TCP connections. One is TCP/IPv6 and another is TCP/IPv4, in the picture: from A6 to B6 (as C6::X4), and Y4 to X4, like below:

```
TCP/IPv6: the initiating host (A6) --> the TRT system (as C6::X4)
    address on IPv6 header: A6 -> C6::X4
TCP/IPv4: the TRT system (Y4) --> the destination host (X4)
    address on IPv4 header: Y4 -> X4
```

4. Address mapping

As seen in the previous section, an initiating host must use a special form of IPv6 address to connect to an IPv4 destination host. The special form can be resolved from a hostname by static address mapping table on the initiating host (like /etc/hosts in UNIX), special DNS server implementation, or modified DNS resolver implementation on initiating host.

5. Notes to implementers

TRT for UDP must take care of path MTU issues on the UDP/IPv6 side. The good thing is that, as we do not relay IP layer packets between IPv4 and IPv6, we can decide IPv6 path MTU independently from IPv4 traffic. A simple solution would be to always fragment packets from the TRT system to UDP/IPv6 side to IPv6 minimum MTU (1280 octets), to eliminate the need for IPv6 path MTU discovery.

Though the TRT system only relays {TCP,UDP} traffic, it needs to check ICMPv6 packets destined to C6::X4 as well, so that it can recognize path MTU discovery messages and other notifications between A6 and C6::X4.

When forwarding TCP traffic, a TRT system needs to handle urgent data [Postel, 1981] carefully.

To relay NAT-unfriendly protocols [Hain, 2000] a TRT system may need to modify data content, just like any translators which modifies the IP addresses.

Scalability issues must carefully be considered when you deploy TRT systems to a large IPv6 site. Scalability parameters would be (1) number of connections the operating system kernel can accept, (2) number of connections a userland process can forward (equals to number of filehandles per process), and (3) number of transport relaying processes on a TRT system. Design decision must be made to use proper number of userland processes to support proper number of connections.

To make TRT for TCP more scalable in a large site, it is possible to have multiple TRT systems in a site. This can be done by taking the following steps: (1) configure multiple TRT systems, (2) configure different dummy prefix to them, (3) and let the initiating host pick a dummy prefix randomly for load-balancing. (3) can be implemented as follows; If you install special DNS server to the site, you may (3a) configure DNS servers differently to return different dummy prefixes and tell initiating hosts of different DNS servers. Or you can (3b) let DNS server pick a dummy prefix randomly for load-balancing. The load-balancing is possible because you will not be changing destination address (hence the TRT system), once a TCP connection is established.

For address mapping, the authors recommend use of a special DNS server for large-scale installation, and static mapping for small-scale installation. It is not always possible to have special resolver on the initiating host, and assuming it would cause deployment problems.

6. Applicability statement

Combined with a special DNS server implementation (which translates IPv4 addresses into IPv6), TRT systems support IPv6-to-IPv4 translation very well. It requires no change to existing IPv6 clients, nor IPv4 servers, so the TRT system can be installed very easily to existing IPv6-capable networks.

IPv4-to-IPv6 translation is much harder to support with any of the translator techniques [Yamamoto, 1998]. While it is possible to use TRT system for IPv4-to-IPv6 translation, it requires nontrivial mapping between DNS names to temporary IPv4 addresses, as presented in NAT-PT RFC [Tsirtsis, 2000].

As presented in the earlier sections, TRT systems use transport layer (TCP/UDP) relay technique to translate IPv6 traffic to IPv4 traffic. It gives two major benefits: (1) the implementation of the TRT system can be done very simple, (2) with the TRT system path MTU discovery issue is easier to deal with, as we can decide IPv6 path MTU independently from IPv4 path MTU. Even with the simplicity, the TRT system can cover most of the daily applications (HTTP, SMTP, SSH, and many other protocols). For NAT-unfriendly protocols, a TRT system may need to modify data content, just like any translators/NATs. As the TRT system reside in transport layer, it is not possible for the TRT system to translate protocols that are not known to the TRT system.

Normally users do not want to translate DNS query/reply traffic using the TRT system. Instead, it makes more sense to run standard DNS server, or special DNS server that helps TRT system, somewhere in the site IPv6 network. There are two reasons to it:

- Transport issue - It is a lot easier to provide recursive DNS server, accessible via IPv6, than to translate DNS queries/replies across the TRT system. If someone tries to ask TRT to translate DNS packets, the person would put C6::X (where C6 is TRT reserved prefix and X is an IPv4 address of a DNS server) into /etc/resolv.conf. The configuration is rather complicated than we normally want.

- Payload issue - In some installation it makes more sense to transmit queries/replies unmodified, across the TRT system. In some installation it makes more sense to translate IPv4 DNS queries (like queries for AAAA record) into queries for A record, and vice versa, to invite traffic into the TRT system. It depends on the installation/configuration at the user’s site.

7. Security Considerations

Malicious party may try to use TRT systems akin to an SMTP open relay [Lindberg, 1999] for traffic to IPv4 destinations, which is similar to circumventing ingress filtering [Ferguson, 1998] , or to achieve some other improper use. TRT systems should implement some sorts of access control to prevent such improper usage.

A careless TRT implementation may be subject to buffer overflow attack, but this kind of issue is implementation dependent and outside the scope of this memo.

Due to the nature of TCP/UDP relaying service, it is not recommended to use TRT for protocols that use authentication based on source IP address (i.e., rsh/rlogin).

A transport relay system intercepts TCP connection between two nodes. This may not be a legitimate behavior for an IP node. The document does not try to claim it to be legitimate.

IPsec cannot be used across a relay.

Use of DNS proxies that modify the RRs will make it impossible for the resolver to verify DNSsec signatures.

References

[Nordmark, 2000.] Nordmark, E., "Stateless IP/ICMP Translator (SIIT)", RFC 2765, February 2000.

[Postel, 1981.] Postel, J., "Transmission Control Protocol", STD 7, RFC 793 September 1981.

[Hain, 2000.] Hain, T., "Architectural Implications of NAT", RFC 2993, November 2000.

[Yamamoto, 1998] K. Yamamoto, A. Kato, M Sumikawa, and J. Murai, "Deployment and Experiences of WIDE 6bone", in Proceedings of INET98, 1998.

[Tsirtsis, 2000.] Tsirtsis, G. and P. Srisuresh, "Network Address Translation - Protocol Translation (NAT-PT)", RFC 2766, February 2000.

[Lindberg, 1999.] Lindberg, G., "Anti-Spam Recommendations for SMTP MTAs", RFC 2505, February 1999.

[Ferguson, 1998.] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", RFC 2267, January 1998.

Appendix A. Operational experiences

WIDE KAME IPv6 stack implements TRT for TCP, called "FAITH". The implementation came from WIDE Hydrangea IPv6 stack, which is one of ancestors of the KAME IPv6 stack.

The FAITH code has been available and operational for more than 5 years. The implementation has been used at WIDE research group offsite meeting, and IETF ipngwg 1999 Tokyo interim meeting. At the latter occasion, we configured IPv6-only terminal network cluster just like we do in IETF meetings, and used a TRT system to support more than 100 IPv6 hosts on the meeting network to connect to outside IPv4 hosts. From statistics we gathered SSH, FTP, HTTP, and POP3 are the most popular protocol we have relayed. The implementation was also used in the terminal cluster IPv6 network at IETF48, IETF49 and IETF50.

The source code is available as free software, bundled in the KAME IPv6 stack kit.

Special DNS server implementations are available as "newbie" DNS server implementation by Yusuke DOI, and "totd" DNS proxy server from University of Tromso (Norway).

Acknowledgements

The authors would like to thank people who were involved in implementing KAME FAITH translator code, including Kei-ichi SHIMA and Munechika SUMIKAWA.

Authors' Addresses

Jun-ichiro itojun HAGINO
Research Laboratory, Internet Initiative Japan Inc.
Takebashi Yasuda Bldg.,
3-13 Kanda Nishiki-cho,
Chiyoda-ku, Tokyo 101-0054, JAPAN

Phone: +81-3-5259-6350
Fax: +81-3-5259-6351
EMail: itojun@iijlab.net

Kazu YAMAMOTO
Research Laboratory, Internet Initiative Japan Inc.
Takebashi Yasuda Bldg.,
3-13 Kanda Nishiki-cho,
Chiyoda-ku, Tokyo 101-0054, JAPAN

Phone: +81-3-5259-6350
Fax: +81-3-5259-6351
EMail: kazu@iijlab.net

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.