Network Working Group
Requests for Comments: 2767
Category: Informational

K. Tsuchiya
H. Higuchi
Y. Atarashi
Hitachi

February 2000

Dual Stack Hosts using the "Bump-In-the-Stack" Technique (BIS)

Status of this Memo

Copyright Notice

Abstract

In the especially initial stage of the transition from IPv4 to IPv6, it is hard to provide a complete set of IPv6 applications. This memo proposes a mechanism of dual stack hosts using the technique called "Bump-in-the-Stack" in the IP security area. The mechanism allows the hosts to communicate with other IPv6 hosts using existing IPv4 applications.

1. Introduction

RFC1933 [TRANS-MECH] specifies transition mechanisms, including dual stack and tunneling, for the initial stage. Hosts and routers with the transition mechanisms are also developed. But there are few applications for IPv6 [IPV6] as compared with IPv4 [IPV4] in which a great number of applications are available. In order to advance the transition smoothly, it is highly desirable to make the availability of IPv6 applications increase to the same level as IPv4. Unfortunately, however, this is expected to take a very long time.

This memo proposes a mechanism of dual stack hosts using the technique called "Bump-in-the-Stack" [BUMP] in the IP security area. The technique inserts modules, which snoop data flowing between a TCP/IPv4 module and network card driver modules and translate IPv4 into IPv6 and vice versa, into the hosts, and makes them self-translators. When they communicate with the other IPv6 hosts, pooled IPv4 addresses are assigned to the IPv6 hosts internally, but the IPv4 addresses never flow out from them. Moreover, since the assignment is automatically carried out using DNS protocol, users do

not need to know whether target hosts are IPv6 ones. That is, this allows them to communicate with other IPv6 hosts using existing IPv4 applications; thus it seems as if they were dual stack hosts with applications for both IPv4 and IPv6. So they can expand the territory of dual stack hosts. Furthermore they can co-exist with other translators because their roles are different.

This memo uses the words defined in [IPV4], [IPV6], and [TRANS-MECH].

2. Components

Dual stack hosts defined in RFC1933 [TRANS-MECH] need applications, TCP/IP modules and addresses for both IPv4 and IPv6. The proposed hosts in this memo have 3 modules instead of IPv6 applications, and communicate with other IPv6 hosts using IPv4 applications. They are a translator, an extension name resolver and an address mapper.

Figure 1 illustrates the structure of the host in which they are installed.

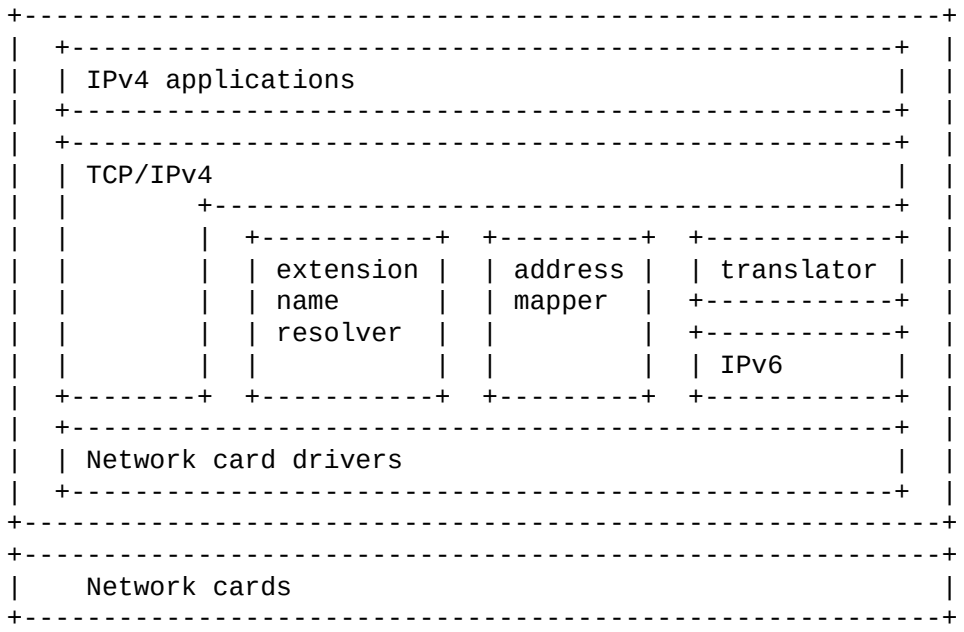

Figure. 1 Structure of the proposed dual stack host

2.1 Translator

It translates IPv4 into IPv6 and vice versa using the IP conversion mechanism defined in [SIIT].

When receiving IPv4 packets from IPv4 applications, it converts IPv4 packet headers into IPv6 packet headers, then fragments the IPv6 packets (because header length of IPv6 is typically 20 bytes larger than that of IPv4), and sends them to IPv6 networks. When receiving IPv6 packets from the IPv6 networks, it works symmetrically to the previous case, except that there is no need to fragment the packets.

2.2 Extension Name Resolver

It returns a "proper" answer in response to the IPv4 application's request.

The application typically sends a query to a name server to resolve 'A' records for the target host name. It snoops the query, then creates another query to resolve both 'A' and 'AAAA' records for the host name, and sends the query to the server. If the 'A' record is resolved, it returns the 'A' record to the application as is. In the case, there is no need for the IP conversion by the translator. If only the 'AAAA' record is available, it requests the mapper to assign an IPv4 address corresponding to the IPv6 address, then creates the 'A' record for the assigned IPv4 address, and returns the 'A' record to the application.

NOTE: This action is similar to that of the DNS ALG (Application Layer Gateway) used in [NAT-PT]. See also [NAT-PT].

2.3 Address mapper

It maintains an IPv4 address spool. The spool, for example, consists of private addresses [PRIVATE]. Also, it maintains a table which consists of pairs of an IPv4 address and an IPv6 address.

When the resolver or the translator requests it to assign an IPv4 address corresponding to an IPv6 address, it selects and returns an IPv4 address out of the spool, and registers a new entry into the table dynamically. The registration occurs in the following 2 cases:

(1) When the resolver gets only an 'AAAA' record for the target host name and there is not a mapping entry for the IPv6 address.
(2) When the translator receives an IPv6 packet and there is not a mapping entry for the IPv6 source address.

NOTE: There is only one exception. When initializing the table, it registers a pair of its own IPv4 address and IPv6 address into the table statically.

3. Action Examples

This section describes action of the proposed dual stack host called "dual stack," which communicates with an IPv6 host called "host6" using an IPv4 application.

3.1 Originator behavior

This subsection describes the originator behavior of "dual stack." The communication is triggered by "dual stack."

The application sends a query to its name server to resolve 'A' records for "host6."

The resolver snoops the query, then creates another query to resolve both 'A' and 'AAAA' records for the host name, and sends it to the server. In this case, only the 'AAAA' record is resolved, so the resolver requests the mapper to assign an IPv4 address corresponding to the IPv6 address.

NOTE: In the case of communication with an IPv4 host, the 'A' record is resolved and then the resolver returns it to the application as is. There is no need for the IP conversion as shown later.

The mapper selects an IPv4 address out of the spool and returns it to the resolver.

The resolver creates the 'A' record for the assigned IPv4 address and returns it to the application.

NOTE: See subsection 4.3 about the influence on other hosts caused by an IPv4 address assigned here.

The application sends an IPv4 packet to "host6."

The IPv4 packet reaches the translator. The translator tries to translate the IPv4 packet into an IPv6 packet but does not know how to translate the IPv4 destination address and the IPv4 source address. So the translator requests the mapper to provide mapping entries for them.

The mapper checks its mapping table and finds entries for each of them, and then returns the IPv6 destination address and the IPv6 source address to the translator.

NOTE: The mapper will register its own IPv4 address and IPv6 address into the table beforehand. See subsection 2.3.

The translator translates the IPv4 packet into an IPv6 packet then fragments the IPv6 packet if necessary and sends it to an IPv6 network.

The IPv6 packet reaches "host6." Then "host6" sends a new IPv6 packet to "dual stack."

The IPv6 packet reaches the translator in "dual stack."

The translator gets mapping entries for the IPv6 destination address and the IPv6 source address from the mapper in the same way as before.

Then the translator translates the IPv6 packet into an IPv4 packet and tosses it up to the application.

The following diagram illustrates the action described above:

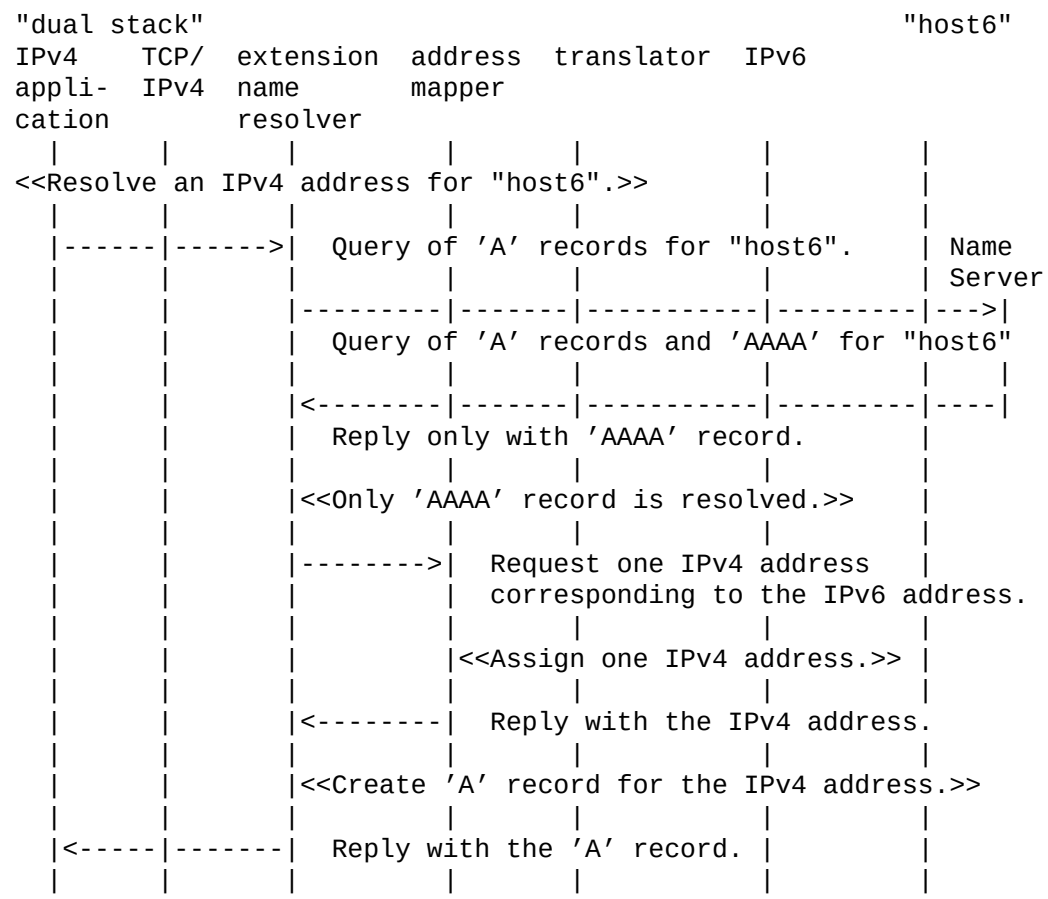

Figure 2 Action of the originator (1/2)

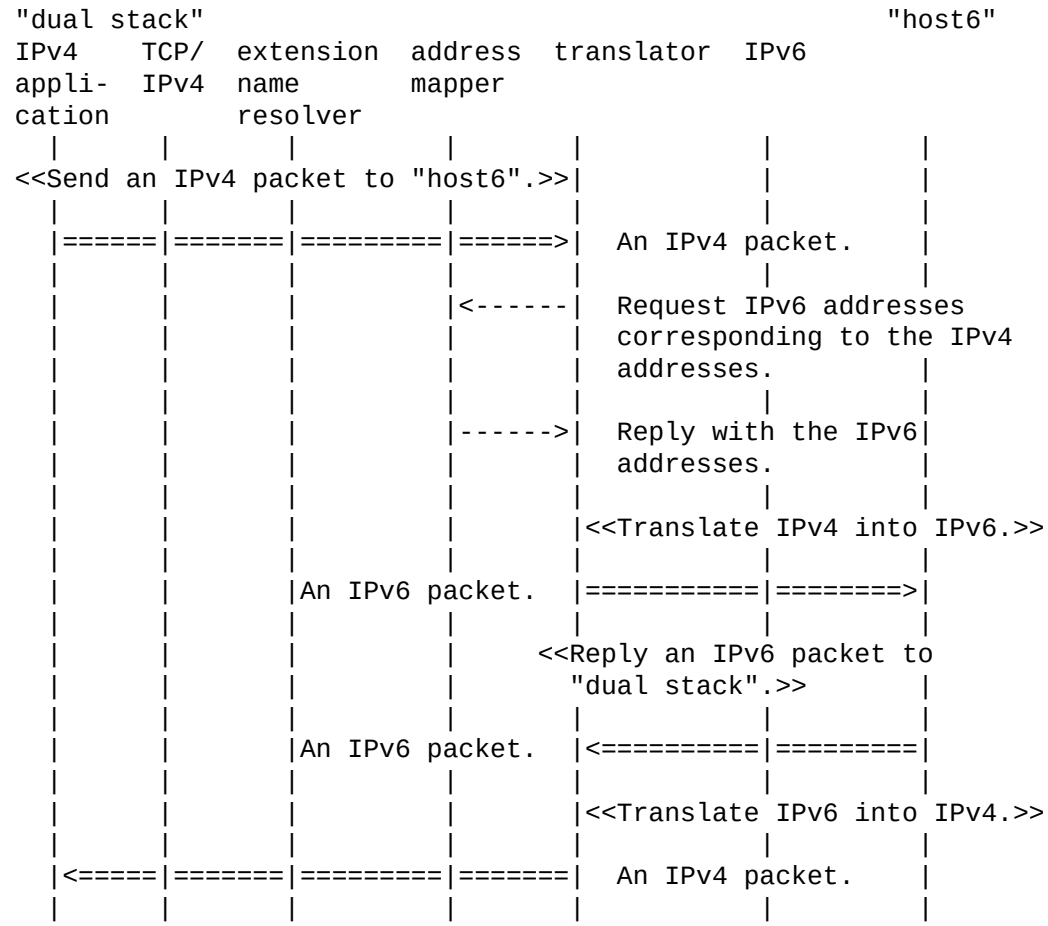

Figure 2 Action of the originator (2/2)

3.2 Recipient behavior

This subsection describes the recipient behavior of "dual stack." The communication is triggered by "host6."

"host6" resolves the ’AAAA’ record for "dual stack" through its name server, and then sends an IPv6 packet to the IPv6 address.

The IPv6 packet reaches the translator in "dual stack."

The translator tries to translate the IPv6 packet into an IPv4 packet but does not know how to translate the IPv6 destination address and the IPv6 source address. So the translator requests the mapper to provide mapping entries for them.

The mapper checks its mapping table with each of them and finds a mapping entry for the IPv6 destination address.

NOTE: The mapper will register its own IPv4 address and IPv6 address into the table beforehand. See subsection 2.3.

But there is not a mapping entry for the IPv6 source address, so the mapper selects an IPv4 address out of the spool for it, and then returns the IPv4 destination address and the IPv4 source address to the translator.

NOTE: See subsection 4.3 about the influence on other hosts caused by an IPv4 address assigned here.

The translator translates the IPv6 packet into an IPv4 packet and tosses it up to the application.

The application sends a new IPv4 packet to "host6."

The following behavior is the same as that described in subsection 3.1.

The following diagram illustrates the action described above:

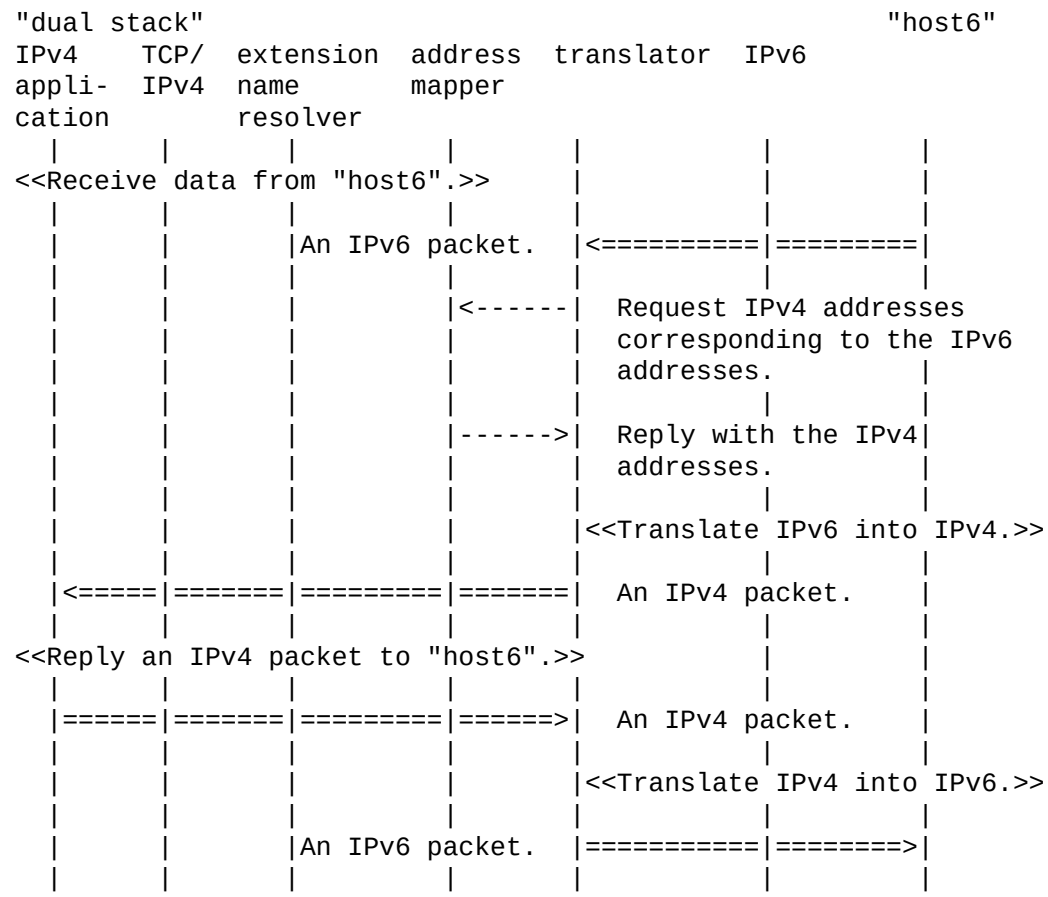

Figure 3 Action of the recipient

4. Considerations

This section considers some issues of the proposed dual stack hosts.

4.1 IP conversion

In common with NAT [NAT], IP conversion needs to translate IP addresses embedded in application layer protocols, which are typically found in FTP [FTP]. So it is hard to translate all such applications completely.

4.2 IPv4 address spool and mapping table

The spool, for example, consists of private addresses [PRIVATE]. So a large address space can be used for the spool. Nonetheless, IPv4

addresses in the spool will be exhausted and cannot be assigned to IPv6 target hosts, if the host communicates with a great number of other IPv6 hosts and the mapper never frees entries registered into the mapping table once. To solve the problem, for example, it is desirable for the mapper to free the oldest entry in the mapping table and re-use the IPv4 address for creating a new entry.

4.3 Internally assigned IPv4 addresses

IPv4 addresses, which are internally assigned to IPv6 target hosts out of the spool, never flow out from the host, and so do not negatively affect other hosts.

5. Applicability and Limitations

This section considers applicability and limitations of the proposed dual stack hosts.

5.1 Applicability

The mechanism can be useful for users in the especially initial stage where some applications not modified into IPv6 remain. And it can also help users who cannot upgrade their certain applications for some reason after all applications have been modified. The reason is that it allows hosts to communicate with IPv6 hosts using existing IPv4 applications, and that they can get connectivity for both IPv4 and IPv6 even if they do not have IPv6 applications as a result.

Note that it can also work in conjunction with a complete IPv6 stack. They can communicate with both IPv4 hosts and IPv6 hosts using IPv4 applications via the mechanism, and can also communicate with IPv6 hosts using IPv6 applications via the complete IPv6 stack.

5.2 Limitations

The mechanism is valid only for unicast communication, but invalid for multicast communication. Multicast communication needs another mechanism.

It allows hosts to communicate with IPv6 hosts using existing IPv4 applications, but this can not be applied to IPv4 applications which use any IPv4 option since it is impossible to translate IPv4 options into IPv6. Similarly it is impossible to translate any IPv6 option headers into IPv4, except for fragment headers and routing headers. So IPv6 inbound communication having the option headers may be rejected.

In common with NAT [NAT], IP conversion needs to translate IP addresses embedded in application layer protocols, which are typically found in FTP [FTP]. So it is hard to translate all such applications completely.

It may be impossible that the hosts using the mechanism utilize the security above network layer since the data may carry IP addresses.

Finally it can not combine with secure DNS since the extension name resolver can not handle the protocol.

6. Security Considerations

This section considers security of the proposed dual stack hosts.

The hosts can utilize the security of all layers like ordinary IPv4 communication when they communicate with IPv4 hosts using IPv4 applications via the mechanism. Likewise they can utilize the security of all layers like ordinary IPv6 communication when they communicate with IPv6 hosts using IPv6 applications via the complete IPv6 stack. However, unfortunately, they can not utilize the security above network layer when they communicate with IPv6 hosts using IPv4 applications via the mechanism. The reason is that when the protocol data with which IP addresses are embedded is encrypted, or when the protocol data is encrypted using IP addresses as keys, it is impossible for the mechanism to translate the IPv4 data into IPv6 and vice versa. Therefore it is highly desirable to upgrade to the applications modified into IPv6 for utilizing the security at communication with IPv6 hosts.

7. References

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

[IPV4] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[FTP] Postel, J. and J. Reynolds, "File Transfer Protocol", STD 9, RFC 959, October 1985.

[NAT] Kjeld B. and P. Francis, "The IP Network Address Translator (NAT)", RFC 1631, May 1994.

[IPV6] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[PRIVATE] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G. J. and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.

[TRANS-MECH] Gilligan, R. and E. Nordmark, "Transition Mechanisms for IPv6 Hosts and Routers", RFC 1933, April 1996.

[BUMP] D.A. Wagner and S.M. Bellovin, "A Bump in the Stack Encryptor for MS-DOS Systems", The 1996 Symposium on Network and Distributed Systems Security (SNDSS'96) Proceedings.

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

8. Acknowledgements

The authors gratefully acknowledge the many helpful suggestions of the members of the WIDE Project, Kazuhiko YAMAMOTO, Jun MURAI, Munechika SUMIKAWA, Ken WATANABE, and Takahisa MIYAMOTO, at large.

9. Authors' Addresses

Kazuaki TSUCHIYA
Enterprise Server Division, Hitachi, Ltd.
810 Shimoimaizumi, Ebina-shi, Kanagawa-ken, 243-0435 JAPAN

Phone: +81-462-32-2121
Fax: +81-462-35-8324
EMail: tsuchi@ebina.hitachi.co.jp

Hidemitsu HIGUCHI
Enterprise Server Division, Hitachi, Ltd.
810 Shimoimaizumi, Ebina-shi, Kanagawa-ken, 243-0435 JAPAN

Phone: +81-462-32-2121
Fax: +81-462-35-8324
EMail: h-higuti@ebina.hitachi.co.jp

Yoshifumi ATARASHI
Enterprise Server Division, Hitachi, Ltd.
810 Shimoimaizumi, Ebina-shi, Kanagawa-ken, 243-0435 JAPAN

Phone: +81-462-32-2121
Fax: +81-462-35-8324
EMail: atarashi@ebina.hitachi.co.jp

10. Full Copyright Statement

Acknowledgement

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