Network Working Group
Internet-Draft
Expires: September 6, 2002

A. Azcorra
A. Garcia-Martinez
M. Bagnulo
UC3M

March 8, 2002

Internet Protocol, Version 64 (IPv64) Specification
draft-azcorra-ipv64-04

Status of this Memo

Copyright Notice

Abstract

This document specifies an IPv6 protocol extension that allows IPv6 packets to be backward compatible with IPv4. An IPv6 packet encapsulated in IPv4 in this way (called IPv64) would be processed as native IPv6 by IPv64 routers, and at the same time, in case there is an IPv4-only router in the path, it will be processed as IPv4. Consequently, it is possible to have native end-to-end IPv6 communication, with IPv6 processing at IPv64 routers, through a path that contains some IPv4-only routers.

Protocol conversion from/to IPv6 to/from IPv64 can be made by local

routers at both ends, and therefore the advantages of IPv64 are achieved with standard native IPv6 hosts.

1. Changes from previous version of the draft

Clarification on the usage of the standard IPv6 and IPv4 headers.

Specification of protocol conversion from/to IPv6 to/from IPv64 may be made by local routers at both ends.

Extension header to keep the IPv4 header when doing transit through IPv6-only networks.

Prototype implementation may be downloaded from:

matrix.it.uc3m.es/~ipv64

2. Introduction

The intention of this document is to provide a complementary transition mechanism to facilitate the migration from IPv4 to IPv6. This additional transition mechanism would allow IPv6 to be backward compatible with IPv4. Being backward compatible means that IPv6 packets encapsulated in IPv4 will be processed as IPv6 by IPv6 routers (and not according to the encapsulating IPv4 header), while IPv4 routers will process them as IPv4 (according to the encapsulating IPv4 header).

To distinguish in the remaining text of this document between plain IPv4-tunneled IPv6 packets and IPv4-encapsulated IPv6 packets that will be processed as IPv6 by IPv6 routers, the former will just be called tunneled IPv6 packets, while the latter will be called "IPv64" packets. IPv4 packets that do not carry an IPv6 packet will just be called IPv4 packets.

The approach described in this document has the advantage that it allows the communication between two IPv6 hosts with packets being processed as IPv6 packets at IPv64 routers, and being processed as IPv4 at IPv4-only routers. Therefore, it is possible to use routing based on an IPv6 destination address (including all new types), use IPv6 source routing, and use hop-by-hop extension headers at in-transit IPv64 routers, while in-transit IPv4-only routers will still route the packet correctly.

Current transition approaches work well to interconnect IPv6 islands through IPv4 clouds. The IPv64 approach offers advantages for the coming situation in which there will be an infrastructure composed of a substantial amount of both IPv4 and IPv6 user and transit networks.

To achieve the aforementioned functionalities it is required that

IPv64 routers recognize IPv64 packets, distinguishing them from other IPv4 packets, in order to process them as IPv6 packets.

2.1 Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119.

3. IPv64 Packet Format

The format of an IPv64 packet is the same as an IPv6 packet encapsulated in IPv4, with the specific utilization of the IPv4 fields described in the following subsections.

3.1 IPv4 Total Length

This IPv4 field contains the total length of the IPv4 packet. This field will be unchanged by IPv64 routers unless there is a modification of the size of the enclosed IPv6 packet (i.e. because of modifications of extension headers).

3.2 Bit 16 of the second word of IPv4 header

Bit number 16 of the second word of the IPv4 header (i.e. bit number 48, beginning with bit 0, of the header) will be called "IPv64 packet". This bit MUST be set to 1 to identify that this packet is an IPv64 packet, and not a regular tunneled IPv6 packet. Regular tunneled IPv6 packets will be processed as IPv4 at IPv64 routers. This bit is currently unused under RFC 791 [4].

3.3 IPv4 Fragmentation Control Fields

These rules apply to IPv4 fields Identification, Do not Fragment (DF), More Fragments (MF), and Fragment Offset.

IPv4 in-transit or source fragmentation of IPv64 packets is undesirable because the second and subsequent fragments would not contain the IPv6 headers. As IPv6 headers are not present, IPv64 routers in the path will only be able to process IPv64 fragments as IPv4 packets, thus loosing the whole IPv6 network functionality. Therefore, IPv4 fragmentation is not allowed.

The IPv4 DF bit MUST be set to one. The source node MUST provide an appropriate mechanism to use an IPv4 packet size that MUST be below the minimum IPv4 MTU in the path to each destination, in order to avoid that the IPv64 packet be discarded.

A straightforward mechanism, although somehow inefficient in transmission overhead, is to always use 576 octets as MTU. Other more efficient mechanisms remain for further study.

In case the upper layer at the source desires to send a packet size above the path MTU (e.g. to send a large UDP datagram), IPv6

fragmentation will be used.

As a consequence of the above discussion, IPv4 fields MF and Fragment

Offset MUST always be set to zero.

3.4 IPv4 Time To Live

This IPv4 field will only be processed by IPv4-only routers. IPv64 capable routers MUST NOT modify this field. IPv64 capable routers MUST modify the Hop Limit field in the enclosed IPv6 header.

3.5 IPv4 Header Checksum

In case an IPv64 capable router modifies a field in the IPv4 header, then the checksum will have to be recalculated. Examples of cases in which an IPv64 capable router has to modify a field in the IPv4 header are the modification of a hop-by-hop extension header (that implies a modification in the IPv4 Total Length field), or the remarking of the IPv4 TOS field at a Differentiated Services IPv64 edge router.

3.6 IPv4 Source and Destination addresses

These fields MUST contain a sufficiently close IPv4 source and destination addresses for the IPv64 packet. A sufficiently close address means that it is not required that the IPv4 packet be addressed to the final destination, but rather to a place beyond which it will be processed only by IPv64 capable routers (possibly combined with IPv6-only routers).

The source IPv4 address has to be sufficiently close to the source IPv6 address, while the destination IPv4 address has to be sufficiently close to the destination IPv6 address.

For example, it is possible to have a domain with a border router that only has one public IPv4 address in its interface, while there are many IPv6 hosts and IPv64 routers internally. In this case the public IPv4 address used by any host sending packets to any host within that domain will be the one of the border router, so the packets are correctly routed from anywhere in the Internet to the border router. Beyond the border router the IPv4 address will be ignored, as all routers are IPv64, and packets will be routed based only on the IPv6 destination address field.

The destination address field in the IPv4 header and the IPv6 header need not correspond to the same system and interface, but they must be consistent, as described above. The same applies for the IPv4 and IPv6 source address fields.

3.7 IPv6 Extension Headers in IPv64 packets

IPv6 extension headers are allowed in IPv64 packets. The extension headers are located, as in regular IPv6 packets, following the IPv6 header, and with the same structure and semantics.

4. Identification of IPv64 packets at IPv64 nodes

When an IPv64 router receives an IPv4 packet with value 4 in the Internet Version field, it will need to know whether it is a native IPv4 packet (including in this category pure tunneled IPv6 packets) or an IPv64 packet, in order to decode and process it correctly.

The proposal for this function is that the currently unused bit in the IPv4 header (that has been called "IPv64 Packet" field in this document) be set to 1 in IPv64 packets. The IP specification in RFC 791 [4] indicates that even thought this bit is unused, it must be set to 0. Therefore, IPv4 nodes sending IPv4 packets would set this bit to 0, while IPv64 packets would have it set to 1. IPv64 routers

or destination nodes would use the value of this bit to distinguish between incoming IPv4 packets and IPv64 packets.

This proposal is built on the assumption that all IPv4 implementations comply with RFC 791 [4], setting bit 16 of the second word of the header to zero at the source, and ignoring its value when processing the packet at routers and the destination node. As this might not be the case, and the number of non-compliant implementations could be significant, several alternative procedures have been considered, but they are not described in this version of the draft.

5. Processing IPv64 packets at IPv64 nodes

An IPv64 source node needs to know, in addition to the source and destination IPv6 of the packet to be built, the corresponding sufficiently close IPv4 addresses. The procedure to obtain the corresponding IPv4 addresses is described in the next section.

An IPv64 router receiving an IPv64 packet will first have to identify it as such (see section 4 in this document). Once it has identified the packet as IPv64, then it will process the packet as a native IPv6 packet, ignoring the fields of the IPv4 header, with the exception of the TOS field, whose value is used instead of the one from Traffic Class field (IPv4 remarking is acceptable). Once the packet has been processed as an IPv6 packet, and the outgoing IPv6 packet has been constructed, the outgoing IPv4 header will be constructed. The outgoing IPv4 basic header (i.e. without the options field) will be the same as the incoming one, with the following exceptions:

- Total Length: it will be recalculated if the length of the IPv6 enclosed packet has been modified (e.g. routing header).
- Type Of Service: it will be modified in case this is an edge router and remarking of the DSCP field is needed. In this case, the value of the outgoing DSCP field will be set according to the Differentiated Services specification.

- o IPv4 addresses: only modified if NAT is being performed. The combined usage of IPv64 and IPv4 NAT is left for further study.

- o Checksum: it will be recalculated if the incoming IPv4 header has been modified.

Therefore, at an IPv64 router the fields of the IPv4 header in the incoming IPv64 packet are used only to:

1. Identify the packet as an IPv64 packet and not a plain IPv4 packet.

2. The Traffic Class value in the incoming IPv6 header has to be ignored, and its value be taken from the TOS octet of the IPv4 header. Notice that an IPv4 edge router performing remarking would only remark the DSCP in the IPv4 TOS field.

3. Generate the appropriate IPv4 header in the outgoing IPv64 packet.

In the case of direct delivery of the IPv64 packet to its IPv6 destination, the address resolution function MUST be performed first

using the IPv6 destination address.

6. IPv64 Protocol Translation

IPv64 packet may transit correctly through IPv64 networks, IPv6 dual-stack networks and IPv4-only networks, but may not interoperate with IPv6-only nodes. For this reason, protocol conversion between IPv6 and IPv64 has been introduced.

Protocol translation may be used to avoid modifying the end-systems in order to make them IPv64 compliant. In this scenario, protocol translation would be provided at the IPv64 router with a direct delivery capacity to the IPv6-only end-system. Therefore, IPv6-only hosts would communicate with their local router using native IPv6, while all the remaining end to end path would be performed with an IPv64 packet generated at the local router. An IPv64 router with IPv6 hosts connected to one of its interfaces can be configured to perform protocol translation at that interface. This means that the router will translate incoming IPv6 packets from that interface to IPv64 packets, and IPv64 packets directed to a host on that interface will be translated to native IPv6.

Protocol translation may also be used to perform transit of IPv64 packets through an IPv6-only network, without the need to perform tunneling of IPv64 packets within IPv6 packets. Routers at both ends of the IPv6-only network would perform protocol translation to/from IPv64 from/to native IPv6.

6.1 Protocol translation from IPv64 to IPv6

Translating an IPv64 packet into an IPv6 packet is made by suppresing the IPv4 header from the IPv64 packet. However, it is mandatory to keep the IPv4 header information within the packet in order to allow an immediate IPv6 to IPv64 translation of the packet, as it might be needed further in the path to perform the reverse translation (IPv6 to IPv64).

In order to keep the IPv4 header information of the IPv64 packet within the native IPv6-only packet, a specific extension header is proposed. The extension header would allow that the IPv4 header information is available if translation back to IPv64 is needed, while it would allow correct processing at the destination IPv6 host if it receives the native IPv6 packet. The detailed coding of the IPv4 information required in the extension header is left for further study.

6.2 Protocol translation from IPv6 to IPv64

Protocol translation from IPv6 to IPv64 requires that the IPv64 router has an implemented function that obtains the sufficiently

close IPv4 addresses associated to both the IPv6 source and destination. This function is the same as the one required in native IPv64 end-systems to be able to generate the IPv64 packet from the IPv6 source and destination addressing information.

This function is not needed in all IPv64 routers. Typically, it will be installed in those interfaces of those IPv64 routers that have IPv6-only hosts connected, or in those interfaces of those IPv64 routers that connect to an IPv6-only network to perform transit through it.

This function has also to be installed in an IPv64 router that needs to produce its own IPv64 traffic (e.g. to communicate directly with a host or with another router).

In the particular case of translating an IPv6 packet that has been produced from a previous translation of IPv64 to IPv6, the IPv4 addresses are kept in a specific IPv6 extension header (see the previous sub-section). In this case, it is trivial to obtain the required sufficiently close IPv4 source and destination addresses because they are contained in the packet to be translated.

In the remaining situations, the generation of an IPv4 address from an IPv6 address will be made by applying a combination of complementary (not alternative) procedures. Notice that the relation is not biyective but inyective. This is, several IPv6 addresses will produce the same "sufficiently close" public IPv4 address. For this reason, the implementation of the function will be based on associating an IPv4 address to an IPv6 prefix. The association of an IPv4 address to a single IPv6 address is just a particular case, which is not excluded, but that will not be the most frequent case. The complementary procedures to be used are the following:

1. Configured Table: the system has a table in which each entry contains an IPv6 address/prefix, and its associated sufficiently close IPv4 address. The system will perform table lookup of the desired IPv6 address to find an applicable table entry that renders the corresponding public IPv4 address. This procedure is suitable for the source address (that will be locally known), and for some cases of destination addresses, but it will not serve in the general case for any IPv6 destination address.

2. Backward learning: the system will learn sufficiently close destination IPv4 addresses by inspecting the IPv4 and IPv6 source addresses of IPv64 packets that are received by it. This method is particularly suitable for information servers, in which the system will normally send packets that are responses to incomming packets. By performing backward learning the system will always

have the correctly resolved IPv4 address to the IPv6 destination that it wants to respond to.

3. IPv6 addresses with embedded IPv4 address: the system obtains the sufficiently close IPv4 address from the IPv6 address itself.

This is applicable, for example, to IPv6 addresses that code an IPv4 address.

4. Cached table: the system will maintain a cached table of previously resolved associations. The table will have the same structure as the configured table above (pairs of an IPv6 address/prefix plus its associated IPv4 address). As in any cached table, entries will be suppressed either by timeout (to allow automatic update of changing situations), or by removing the oldest ones when the cache size-limit is reached.

5. DNS look up: performing DNS lookup of a specific entry defined for this purpose. Notice that it is not needed to have a specific entry for each destination, and is enough to have a sufficiently close IPv4 address for a whole domain (e.g. as done for e-mail gateways). It must be taken into account that to perform DNS look up it is required first to perform reverse DNS lookup, to obtain the name from the IPv6 address. This subject remains for further study.

Protocol translation must be made guaranteeing the requirement already described that IPv4 fragmentation of IPv64 packets MUST NOT take place. This implies that the IPv6 MTU being used MUST be, at most, 20 octets smaller than the actual IPv4 MTU of the path. This subject remains for further study.

7. Other Processing Considerations

7.1 Processing IPv64 packets at IPv4-only nodes

Plain IPv4 nodes will treat IPv64 packets as IPv4 packets, as they can not distinguish IPv64 packets from IPv4 packets. It is required that IPv4 nodes comply with RFC 791, in the sense that the unused bit of the header (bit number 48, beggining with bit 0) MUST be forwarded unmodified.

A plain IPv4 end system that receives an IPv64 packet will pass all data after the IPv4 header to the corresponding upper layer protocol entity identified in the Protocol field of the IPv4 header. If the upper layer protocol entity is an IPv6 protocol entity, then the encapsulated packet would be correctly processed.

7.2 Processing IPv64 packets at IPv6-only nodes

IPv6-only nodes cannot process regular IPv64 packets as they begin with the IPv4 header. For this purpose, the function of protocol translation to/from IPv6 has been detailed in the corresponding section. By performing protocol translation at the edges of IPv6-only networks it is possible to perform transit through them, without loosing any of the advantages of IPv64.

7.3 Processing IPv64 packets at IPv6 dual-stack nodes

When performing transit through an IPv6 dual-stack network, two approaches are possible.

The first approach is to just forward the IPv64 packet into the network. In this case, the IPv6 dual-stack nodes will treat IPv64 packets as IPv4 packets, as they can not distinguish IPv64 packets from IPv4 packets.

The second approach is to perform protocol translation, as performed when doing transit through an IPv6-only network. In this case the packet would be processed within the network as IPv6, and protocol translation back to IPv64 would be needed at the outgoing edge of the transit network.

7.4 Firewalls and other protocol functions

The impact of Firewalls, NAT, ICMP diagnostics, ECN, path MTU discovery and other functions in relation to IPv64 remain for further study.

8. Conclusions

The IPv64 transition mechanism described in this document is compatible with other transitions mechanisms based on NAT and on different tunneling approaches, and might be used in conjunction with them.

As more dual-stack IPv6 routers incorporate these functions (becoming IPv64 routers), then more IPv64 packets will be processed as IPv6 instead of as IPv4, smoothly migrating the network functionality to IPv6.

Current transition approaches work well to interconnect IPv6 islands through IPv4 clouds. The advantages of the IPv64 approach will arise in the coming situation in which there will be an infrastructure composed of highly interconnected IPv4 and IPv6 user and transit networks.

IPv64 requires modifications in the procedures of IPv6 implementations in order to recognize and process IPv64 packets as IPv6, instead of forwarding them as IPv4. However, these changes need not be deployed elsewhere, and might be deployed just at some routers without affecting the functionality of the network.

The implementation complexity to upgrade a dual-stack IPv6 router to become an IPv64 transit (core) router is negligible, compared with the complexity of the dual-stack sytem itself. The complexity of the protocol translation functions is also considered low. The per-packet computational cost of pure IPv6 to IPv64 protocol conversion

makes it suitable only to be deployed in the local routers of native IPv6-only hosts local networks.

9. Acknowledgements

This work has used valuable comments received from Tony Hain, Brian E. Carpenter, and Svend Moeller Nielsen. Prototype implementation and tests have been performed by Fernando Anton.

This work has been partly supported by the European Union, under IST projects GCAP and LONG, and also under COST 263 "Quality of Future Internet Services".

References

[1] Deering et. al., S., "Internet Protocol Version 6 Specification", RFC 2460, December 1998.

[2] Nichols et. al., K., "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

[3] Reynolds, J. and J. Postel, "Assigned Numbers", RFC 1700, October 1994.

[4] Postel, J., "Internet Protocol", RFC 791, September 1981.

[5] Srisuresh , P. and K. Egevang , "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.

[6] Ramakrishnan, K. and S. Floyd, "A Proposal to add Explicit Congestion Notification (ECN) to IP", RFC 2481, January 1999.

Authors' Addresses

Arturo Azcorra
Universidad Carlos III de Madrid
Av. Universidad 30
Leganes, Madrid 28911
SPAIN

Phone: +34 91 6248778
EMail: azcorra@it.uc3m.es
URI: http://www.it.uc3m.es

Alberto Garcia-Martinez
Universidad Carlos III de Madrid
Av. Universidad 30
Leganes, Madrid 28911
SPAIN

Phone: +34 91 6248782
EMail: alberto@it.uc3m.es
URI: http://www.it.uc3m.es

Marcelo Bagnulo
Universidad Carlos III de Madrid
Av. Universidad 30
Leganes, Madrid 28911
SPAIN

Phone: +34 91 6249500
EMail: marcelo@it.uc3m.es
URI: http://www.it.uc3m.es

Internet-Draft IPv64 Specification March 2002

Full Copyright Statement

Acknowledgement

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