Network Working Group
Internet-Draft
Intended status: Experimental
Expires: May 10, 2008

B. Carpenter
Univ. of Auckland

November 7, 2007

# Shimmed IPv4/IPv6 Address Network Translation Interface (SHANTI) draft-carpenter-shanti-01

## Status of this Memo



## Copyright Notice



## Abstract

There is a pragmatic need for a packet-level translation mechanism between IPv4 and IPv6, for scenarios where no other mode of IPv4 to IPv6 interworking is possible. The mechanism defined here uses a shim in both the translator and the IPv6 host to mitigate the problems introduced by stateless translation.

Table of Contents

1. Introduction . . . . . 3
1.1. Disclaimer . . . . . 3
1.2. Summary of operation . . . . . 3
1.3. Requirements notation . . . . . 5
2. Scenario of addresses and ports . . . . . 5
3. General walkthroughs . . . . . 8
4. Placement of the shim . . . . . 9
5. DNS . . . . . 9
6. ICMP . . . . . 10
7. Unresolved issues . . . . . 10
8. Security Considerations . . . . . 13
9. IANA Considerations . . . . . 14
10. Acknowledgements . . . . . 14
11. Change log [RFC Editor: please remove this section] . . . . . 14
12. References . . . . . 14
12.1. Normative References . . . . . 14
12.2. Informative References . . . . . 14
Author's Address . . . . . 15
Intellectual Property and Copyright Statements . . . . . 16

Dedication

A few days before his tragic death, itojun (Jun-ichiro Itoh Hagino) responded to a comment that "I absolutely don't like to see ::FFFF/96 on the wire" by writing "then we'd have to deprecate SIIT at least. still, you cannot be sure that ::ffff:0:0/96 are not on the wire." This directly inspired the idea behind SHANTI. This proposal is dedicated to itojun.

## 1. Introduction

### 1.1. Disclaimer

This proposal is incomplete. It is posted to seek comments on plausibility; much more work is needed to make it implementable.

### 1.2. Summary of operation

There has long been a defined mechanism for stateless translation betweeen IPv4 and IPv6 packet formats, named SIIT [RFC2765]. Its intended use is any scenario where dual stack coexistence between IPv4 and IPv6, possibly accompanied by dual stack application level proxies, is insufficient. In the most stringent case, this will occur when communication is needed between unmodified ("legacy") IPv4 hosts and IPv6-only hosts that have no IPv4 code, and no dual stack proxy is available for the application protocol of interest. Thus the scenario of interest is one where an IPv6-only host is modified (with the inclusion of a shim and DNS resolver changes) to allow it to leverage a separate device (the translator) to access IPv4-only sections of the Internet.

The previously proposed solution for this requirement, NAT-PT [RFC2766], has known issues and has been deprecated [RFC4966]. The present proposal does not resolve all of those issues; a later section will identify the issues believed to remain open. This proposal aims to resolve those issues that can be handled if the IPv6 protocol stack communicating with a translator can obtain information about the translation. The objectives are to ensure that

- from the IPv4 host's point of view, nothing is worse than in the case of an IPv4-to-IPv4 translation
- from the IPv6 host's point of view, no special code is generally required in the transport layer or above. However, information about the translation is available in the IPv6 host's network stack, as needed. This is the crucial difference from NAT-PT.
- IPv6 routing is not affected in any way, and there is no risk of importing "entropy" from the IPv4 routing tables into IPv6.

To achieve these goals, a shim is inserted in the protocol stack at both the IPv6 host and at the translator. Its objective is to allow the IPv6 stack at the host to be aware of the presence of the translator, of the addresses involved in the translation, and of any other information known by the translator that may be of value to the IPv6 host. A shim model is chosen, as in SHIM6 [I-D.ietf-shim6-proto], so that upper layer protocols (ULPs) have no need to be aware of anything unusual. The mechanism is known as SHimmed Address Network Translation Interface (SHANTI, which means "inner peace" in Sanskrit).

As in SHIM6, ULPs are presented with an upper layer identifier (ULID) in the form of an IPv6 address which is independent of any manipulation of addresses in the shim or translator.

Additionally, packets that flow over the IPv6 network all have quite normal IPv6 addresses, with no topological constraints. The same applies on the IPv4 side. This means that the translator may be positioned anywhere that is operationally convenient (e.g., on the IPv6 host's own site, within its ISP's network, or much closer to the IPv4 host). The only requirement is that there exists an IPv6 path between the IPv6 host and the translator, and an IPv4 path between the translator and the IPv4 host.

There are two cases to consider:

1. A new flow of packets is started by an IPv6 host. In this case, the principle of operation is that the shim in the IPv6 host exchanges information with the shim in the translator before the first packet of the new flow is released from the sending buffer. The result of the information exchange is that the shim knows what addresses and ports will be used for both IPv6 and IPv4, and can appropriately manipulate the packets before sending them to the translator via IPv6.
2. A new flow of packets is started by an IPv4 host. In this case, the principle of operation is that the shim in the translator sends the first packet to the IPv6 host with a shim header defining what addresses and ports will be used for both IPv6 and IPv4. The shim in the IPv6 host can appropriately manipulate the packets before delivering them to the upper layer protocol.

In neither case is any IPv4 component aware of any difference from a normal IPv4 packet stream.

The reader is assumed to have a general understanding of SHIM6. Although this early draft does not assume that the SHIM6 mechanisms defined in [I-D.ietf-shim6-proto] would be used unchanged, they form a proof of concept for the type of communication required between two network-layer shims.

It should be noted that this mechanism adds complexity to an IPv6-only host. This has to be balanced against the complexity of a dual-stack host. In this model, no residual IPv4 code is needed in the IPv6 host. The shim has to handle the rewriting of addresses and port numbers, but nothing else.

## 1.3. Requirements notation

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 [RFC2119].

## 2. Scenario of addresses and ports

Consider an IPv6-only host X and and IPv4-only host Y.

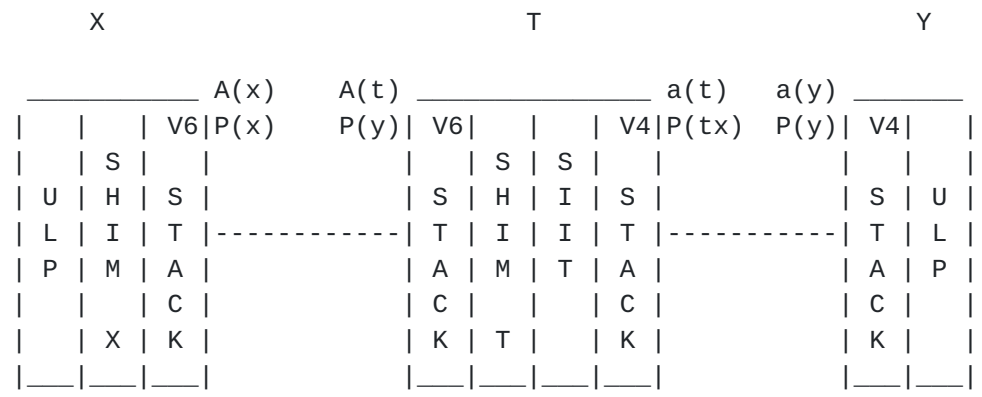
Let A(x) be an IPv6 address for X, and let a(y) be an IPv4 address for Y. Let the port in use at X be P(x) and at Y be P(y).

We will observe later that it is irrelevant whether a(y) is translated by an IPv4 NAT, and whether P(y) is translated by an IPv4 NAPT.

Additionally, consider a translator T between X and Y. On the IPv6 side it has address A(t) and on the IPv4 side it has address a(t). If port translation is in effect, P(x) will become P(tx) on the IPv4 side. We will observe later that the A(t) address can be chosen from an address pool. We cannot assume that a(t) can be chosen from a pool, which is why port translation will be needed.

Thus A() is always an IPv6 address and a() is always an IPv4 address.

A diagram of the solution follows:




We will refer to the shim in X as SHIMX, and the shim in T as SHIMT.

The address set used by the shims for X is conceptually {a(t),A(x)}, and for Y it is conceptually {a(y),A(t)}. In other words the ULP at X sees its own ULID as a(t) and Y's ULID as a(y), both filled out to 128 bits. On the wire, the IPv6 packets between X and T use A(x) and A(t) as the actual address pair. The IPv4 packets between T and Y use a(t) and a(y). P(y) can be used everywhere, but we must assume that P(x) will be used on the IPv6 side and P(tx) on the IPv4 side.

When a(t) and a(y) are filled out to 128 bits, an appropriate /96 prefix must be used. This must checksum to zero when 16-bit transport checksums are computed. In SIIT, the ::ffff:0:0/96 IPv4-mapped format is used to fill out addresses for IPv4 hosts. Also in SIIT, an "IPv4-translated" address format is introduced to represent a synthetic IPv4 address for the IPv6 host, with the ::ffff:0:0:0/96 prefix. This format, which is not in the IPv6 address architecture [RFC4291], could be used as the ULID for X. But since the shim has explicit knowledge of the addresses in use, is there any reason to use this format in preference to the IPv4-mapped format? The latter is assumed here for simplicity.

Further to this, because these addresses never appear on the IPv6 wire in SHANTI, there seems to be no reason in principle why the deprecated ::/96 "IPv4-compatible" prefix could not be used for further simplicity. However, this has been avoided to respect the deprecation.

If there's an IPv4 NAT with routable address a(n) on the IPv4 path, it won't know anything is special, and a(y) will be represented by a(n). X, Y and T won't know that the NAT is there. X and T will not know if Y has a private [RFC1918] address or if additional port translation takes place.

T must have a large pool of A(t) addresses, and should have a complete /64 to itself for maximum flexibility.

SHIMX is configured with knowledge of a default A(t) to start any new exchange with SHIMT, and with knowledge of a(t). SHIMX will catch all packets sent to ::ffff:0:0/96 by any ULP in X. When a ULP sends a first packet to ::ffff:0:0:a(y)/128, we need to start a SHIM6-like process. SHIMX will carry out a message exchange with SHIMT to discover the relevant A(t) and P(tx) values. It can then update the port number and recompute a transport checksum if needed, rewrite the addresses as A(t),A(x), and send the packet on to A(t). Subsequent packets in the same flow will not require a shim message exchange.

Note that the network stack in X will use the ULID ::ffff:0:0:a(t)/128 as the source address for checksum purposes. Source address selection MUST choose this when the destination address matches

::ffff:0:0/96. This is why a(t) must be configured in SHIMX. Checksum recomputation by SHIMX will only be needed if P(tx) != P(x). The NAT-like code for this will require data sharing between the transport protocols and SHIMX.

T needs to select a specific A(t) and P(tx) for each new flow, and exchange SHIM6-like messages with X, to tell SHIMX the values of A(t) and P(tx) . This should create enough state in both shims to know what to do with outbound and return packets. If T has a full /64 to work with, it can create a new A(t) for each new X or even for each new flow if that turns out to be needed.

Note that unlike SHIM6, SHANTI must perform the shim exchange before sending the first packet of an outbound traffic flow from X. This is because SHIMX must learn if P(tx) is unequal to P(x). A consequence of this is that SHIMX should buffer packets of a new outbound flow until it has completed its shim exchange with T. For this to scale, it is important that the translator has adequate capacity for the number of IPv6 hosts it serves, and adequate network connectivity to them, so as to minimize buffering requirements.

When a data packet reaches T from X, there will already be shim state established. The shim will pass the packet on to SIIT for translation and IPv4 transmission.

Once the shim state is established, the ULPs in both X and Y will work as normal. Since T uses a specific A(t) for each X, and the shim at X is aware of that A(t), all port numbers are available in each direction on the IPv6 side. Port mapping, if required, will only affect the IPv4 side of T. Also, SHIMX is aware that the ULP in Y believes it is using the address pair {a(t), a(y)} and the ports {P(tx), P(y)}. Thus, address and port dependent fix-ups can be performed, if needed, by SHIMX. This means that TCP and UDP checksums do not need to be fixed up by T. This has scaling advantages compared to NAT-PT.

Additionally, with this knowledge being available in the host rather than being hidden in the translator as in NAT-PT, it is in principle possible for any address and port dependencies in the ULP to be fixed up in the host itself, precluding the need for Application Level Gateways (ALGs). Although this would introduce a layer violation, it is in principle a more robust design than associating ALGs with a "stateless" translator. In particular, it means that new applications on the IPv6 host do not require new ALG code in the translator, removing a strong dependency in deployment scenarios.

## 3. General walkthroughs

Consider first an IPv6 client attempting to contact an IPv4 server via this mechanism. The main steps that must occur are:

1. ULP in X obtains Y's IPv4-mapped address ::ffff:0:0:a(y)/128. See DNS discussion below.
2. ULP sends unsolicited packet to that address.
3. SHIMX recognises the packet as needing attention.
4. SHIMX creates local state for a(y), P(x), and buffers the packet. Also, it creates a packet to send to T. This is a packet containing nothing but a shim header indicating that a first packet is ready from A(x):P(x) to a(y):P(y).
5. SHIMT receives this shim header and checks for existing state for {A(x):P(x),a(y):P(y)}.
6. If no such state exists, assign an A(t) value from the pool, and create state. Includes the ports. If P(x) is already in use by T, assign a P(tx). Otherwise, P(tx)=P(x).
7. SHIMT creates a packet to return to X. This is a packet containing nothing but a shim header indicating the assigned A(t) and P(tx).
8. SHIMX records this additional state, including P(tx) as the translated port.
9. SHIMX now applies the following process to buffered and future packets sent from ::ffff:0:0:a(t), port P(x) to ::ffff:0:0:a(y), port P(y).
    1. If P(tx) != P(x), recompute transport checksum as for addresses DA=::ffff:0:0:a(y), SA=::ffff:0:0:a(t) and ports DP=P(y), SP=P(tx).
    2. Rewrite destination address as A(t).
    3. Rewrite source address as A(x).
    4. Rewrite destination port as P(tx).
    5. Send packet.
10. SHIMT rewrites the addresses as DA=::ffff:0:0:a(y), SA=::ffff:0:0:a(t), and hands the packet off to SIIT.
11. SIIT translates the packet to IPv4 and sends it on (destination = a(y), source = a(t)).
12. When an IPv4 return packet comes into SIIT, SIIT translates the packet to IPv6 and hands it to SHIMT.
13. The shim performs port demultiplexing on the destination port (which will be P(tx)) to identify the A(x) involved.
14. The shim rewrites the addresses as A(x), A(t) and sends the packet on to A(x).
15. The shim at X receives the packet, rewrites the header to restore the original ULIDs and P(x), and sends the packet on up the stack.

Now consider an IPv4 client attempting to contact an IPv6 server via

T. The main steps that must occur are:

1. T must be pre-configured to admit traffic for P(x) and forward it to A(x). This is a normal port-forwarding issue, to be solved as for NATs or perhaps as proposed in [I-D.woodyatt-ald]. It cannot be performed without pre-existing state. Assuming T has only one a(t), a given P(x) can only have one IPv6 listener.
2. ULP in Y obtains an IPv4 address for T (believing it to be the actual server X).
3. Y sends an unsolicited packet from a(y) to a(t), port P(x).
4. It is passed to SIIT in T, translated to IPv6 format, and passed on to SHIMT.
5. SHIMT performs port demultiplexing and determines that the packet is destined for A(x). It creates state if none exists.
6. SHIMT inserts a shim header that will tell X the translation in effect, translates the addresses, and sends the packet from A(t) to A(x).
7. SHIMX receives the packet, and translates the addresses to ::ffff:0:0:a(t)/128 and ::ffff:0:0:a(y)/128. This should checksum OK. SHIMX creates state if none exists.
8. The packet is delivered to the ULP, minus the shim header.

Subsequent packets will flow as in the previous case.

Shim state will be torn down (deleted) using inactivity timers, as for SHIM6 and typical NATs.

## 4. Placement of the shim

In SHIM6 the shim is logically placed below both the transport and IPsec layers, so that their checksums do not need recalculation. In SHANTI, the transport layer checksum does need to be recalculated by the shim, rather in the manner that a NAT behaves. However, this cannot be done for cryptographic checksums for obvious reasons. The shim should perhaps be regarded as logically below transport, but a better implementation would be for each transport layer to invoke the shim in-line prior to executing its checksum calculation.

## 5. DNS

It is required that the IPv6 hosts "behind" a SHANTI translator either have a resolver that maps A records into AAAA records expanded with ::ffff:0:0/96, or a DNS server that actually stores such records, or performs this transformation on the fly. On the assumption that hosts behind a translator will need to be configured in any case, in order to activate the shim, a mapping resolver seems

likely to be the most robust choice, applying the fate-sharing principle. It would also work in a network with a mixture of SHANTI and dual-stack hosts. The former would see A records mapped as AAAA, and the latter would see native A records.

This illustrates that SHANTI is an all-or-nothing approach. It doesn't seem plausible to activate SHANTI on a dual stack host since DNS entries are either mapped, or they aren't. But why would it be needed?

"Outside" the translator, SHANTI hosts must be represented by an A record with the address of their translator. Specifically, the host's FQDN will have one or more AAAA records with its IPv6 address(es) and an A record with its translator's address. A dynamic DNS-ALG is not needed.

## 6. ICMP

In general, ICMP translation in both directions will proceed as defined in SIIT.

The pool of IPv4 addresses concerned (section 3.5 of [RFC2765]) will contain only a(t), and SHIMT will have to perform port demultiplexing in order to dispatch ICMP messages translated from IPv4 to the correct A(x). SHIMX will have to perform address or checksum rewriting as for other packets. (More details TBD).

## 7. Unresolved issues

This section comments on issues raised in [RFC4966] with regard to whether they are mitigated or resolved by the present specification. The relevant section headings from RFC 4966 are included for reference.

### 2.1. Issues with Protocols Embedding IP Addresses

In SHANTI, these can in principle be resolved within the IPv6 host, with no dependency on an up-to-date translator. This does require the protocol implementation in the IPv6 host to be SHANTI-aware. Also see issue 5 below.

### 2.2. NAPT-PT Redirection Issues

This concerns protocols where the port number is absent or encrypted, so port de-multiplexing is impossible. SHANTI cannot solve this problem; it is intrinsic in sharing one IPv4 address

among many IPv6 hosts. However, since it's an intrinsic problem of the NAPT model, SHANTI doesn't create this problem either; IPv4 hosts already have to live with it.

### 2.3. NAT-PT Binding State Decay

This concerns protocols whose idle times may exceed any reasonable tear-down timer, leading to a risk of P(tx) being reassigned while still in use. This risk should be mitigated in SHANTI, since the tear-down can be synchronized between SHIMX and SHIMT. It would even be theoretically possible for SHIMX to probe the application.

### 2.4. Loss of Information through Incompatible Semantics

This concerns inevitable loss of information such as the IPv6 Flow Label. SHANTI cannot solve this problem; it is intrinsic, as observed in [RFC1671] section B1.

### 2.5. NAT-PT and Fragmentation

Put simply, fragments can't be port-demultiplexed without reassembly. SHANTI cannot solve this problem; it is intrinsic in sharing one IPv4 address among many IPv6 hosts. Only applications that probe for the available MTU size can overcome this issue. However, since it's an intrinsic problem of the NAPT model, SHANTI doesn't create this problem either; IPv4 hosts already have to live with it.

### 2.6. NAT-PT Interaction with SCTP and Multihoming

SCTP includes alternative addresses in its messages. This is solved as in issue 2.1 above. SHANTI would remain a single point of failure for SCTP.

### 2.7. NAT-PT as a Proxy Correspondent Node for MIPv6

The problem is that MIPv6 route optimization cannot possibly be supported on the IPv4 network. This is intrinsic, but in SHANTI it would be possible for SHIMX to suppress messages and headers relating to changes of care-of addresses, including reverse routing checks, at their source, if they are sent to the ::FFFF:0:0/96 prefix.

### 2.8. NAT-PT and Multicast

SHANTI does not handle multicast translation.

Issues 3.1 through 4.5 are partly or completely related to NAT-PT's requirement for a DNS-ALG. SHANTI does require DNS entries for IPv4 hosts to be presented to the ULP as AAAA records, but this does not require a dynamic DNS-ALG to be colocated with the SHANTI translator (see Section 5). Thus, these issues are intrinsically mitigated by SHANTI.

### 3.1. Network Topology Constraints Implied by NAT-PT

Not relevant to SHANTI.

### 3.2. Scalability and Single Point of Failure Concerns

Compared to NAT-PT, a SHANTI translator has a simpler job since checksum calculations are left to the IPv6 host, and DNS-ALG is not needed. Scalability of performance is therefore less of a concern. SHANTI remains a single point of failure, unless a load sharing feature with failover is added. These issues are intrinsic to any translator scenario.

### 3.3. Issues with Lack of Address Persistence

In the absence of DNS-ALG, this appears to be identical to issue 2.3 above.

### 3.4. DoS Attacks on Memory and Address/Port Pools

In the absence of DNS-ALG, this appears to be a "standard" DoS threat to which almost any protocol is exposed. See Section 8.

### 4.1. Address Selection Issues when Communicating with Dual-Stack End-Hosts

In the absence of DNS-ALG, there should be no problem in configuring IPv6 hosts to prefer native IPv6 addresses to IPv4-mapped addresses. Also, the resolver code (Section 5) could be designed to always return IPv4-mapped addresses last in the response to getaddrinfo().

### 4.2. Non-Global Validity of Translated RR Records

If an IPv4-mapped address known by host X in the above scenario is passed on to any other IPv6 host equipped with SHANTI, it will work, assuming that the IPv4 address is globally unique. If it's a private [RFC1918] address, it may fail, but that is intrinsic to private IPv4 addressing. Otherwise, in the absence of DNS-ALG, this issue is not applicable to SHANTI.

4.3. Inappropriate Translation of Responses to A Queries

In the absence of DNS-ALG, this is not applicable to SHANTI.

4.4. DNS-ALG and Multi-Addressed Nodes

In the absence of DNS-ALG, this is not applicable to SHANTI.

4.5. Limitations on Deployment of DNS Security Capabilities

In the absence of DNS-ALG, this is not applicable to SHANTI.

5. Impact on IPv6 Application Development

This is closely related to issue 2.1. As noted above, a SHANTI host is aware of the translation in effect. SHANTI will work "out of the box" for any application that runs through a traditional NAT or NAPT without problems *and* has been upgraded to AF_INET6 sockets. In other cases, the shimmed IPv6 stack can make an application aware of the both the ULIDs in use and of the translated port number, perhaps via socket options. Although modifying application code to take this into account may appear complex, application developers might prefer this to today's obscure failure modes caused by IPv4 NAPT or NAT-PT.

In conclusion, it seems that SHANTI overcomes or mitigates many of the issues noted with NAT-PT. Those that remain appear to be intrinsic to any translation scenario.

## 8. Security Considerations

As for NAT-PT, there is no obvious way to carry network layer IPsec across a SHANTI translator. There seems to be no reason IKE [RFC4306] cannot run in a SHANTI scenario, using its port agility intended for NAT tolerance. But that in itself isn't very useful. It seems likely that security solutions running above the transport layer will be required in order to protect a SHANTI session.

The use of a shim layer in SHANTI will raise some of the security issues considered for SHIM6 . More analysis of the potential spoofing and denial of service threats is needed to determine whether a cryptographic solution is needed, or if there is a straightforward way to prevent attackers taking over a session by impersonating the shim. It may be possible to find a simple method of arranging a shared secret between X and T, such that an elementary hash can be used to authenticate the shim headers.

## 9. IANA Considerations

This document has not yet been exhaustively checked for possible action by the IANA.

## 10. Acknowledgements

Vital comments on a very primitive version of this proposal were made by Marcelo Bagnulo Braun and Iljitsch van Beijnum. Contributions and comments by David Miles and others are gratefully acknowledged.

This document was produced using the xml2rfc tool [RFC2629].

## 11. Change log [RFC Editor: please remove this section]

draft-carpenter-shanti-01: added dedication, clarifications, bug fixes, added RFC4966 analysis, 2007-11-08

draft-carpenter-shanti-00: original version, 2007-10-28

## 12. References

### 12.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2765] Nordmark, E., "Stateless IP/ICMP Translation Algorithm (SIIT)", RFC 2765, February 2000.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.

### 12.2. Informative References

[I-D.ietf-shim6-proto]
Bagnulo, M. and E. Nordmark, "Shim6: Level 3 Multihoming Shim Protocol for IPv6", draft-ietf-shim6-proto-09 (work in progress), November 2007.

[I-D.woodyatt-ald]
Woodyatt, J., "Application Listener Discovery (ALD) for IPv6", draft-woodyatt-ald-01 (work in progress), June 2007.

[RFC1671] Carpenter, B., "IPng White Paper on Transition and Other Considerations", RFC 1671, August 1994.

[RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

[RFC2766] Tsirtsis, G. and P. Srisuresh, "Network Address Translation - Protocol Translation (NAT-PT)", RFC 2766, February 2000.

[RFC4306] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005.

[RFC4966] Aoun, C. and E. Davies, "Reasons to Move the Network Address Translator - Protocol Translator (NAT-PT) to Historic Status", RFC 4966, July 2007.

## Author's Address

Brian Carpenter
Department of Computer Science
University of Auckland
PB 92019
Auckland, 1142
New Zealand

Email: brian.e.carpenter@gmail.com

## Full Copyright Statement



## Intellectual Property



## Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).