Network Working Group
Internet-Draft
Intended status: Experimental
Expires: January 30, 2014

R. Van Rein
OpenFortress B.V.

July 29, 2013

6bed4: Peer-to-Peer IPv6 on Any Internetwork

draft-vanrein-6bed4-01

Abstract

The purpose of 6bed4 is to support IPv6-only applications, even on IPv4-only networks. A specific area of concern is that of peer-to-peer protocols such as SIP or document exchange during a chat session. Such protocols are designed to run in any environment, which means that they cannot rely on IPv6 for themselves, or for their peers. The 6bed4 tunnel mechanism ensures that IPv6 can be assumed on all peers.

The 6bed4 mechanism is meant as a fallback mechanism for IPv6 connectivity on networks that do not support it natively, by running a tunnel over UDP and IPv4. The IPv4 address is used to support traceability of the traffic originator, which means that no user account or other configuration is needed.

The tunnel mechanism encapsulates IPv6 in UDP/IPv4 and builds on existing IPv6 mechanisms; it employs Stateless Address Autoconfiguration [RFC4862] to setup an IPv6 address on a 6bed4 Peer, and Neighbor Discovery [RFC4861] to find the most direct route to a remote 6bed4 Peer.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 30, 2014.

Table of Contents

1. Terminology

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 [RFC2119].

6bed4 Frame: A network frame or packet that consists of an IPv6 frame encapsulated into UDP and then into IPv4.

6bed4 Address: The combination of an IPv4 address and a UDP port number. This pair of coordinates is used to indicate a UDP/IPv4 endpoint the 6bed4-wrapped traffic. A Source 6bed4 Address is a 6bed4 Address comprising of a source IPv4 address and source UDP port; a Destination 6bed4 Address is a 6bed4 Address comprising of a destination IPv4 address and destination UDP port. A 6bed4 Address is called Direct if it represent the coordinates of an endpoint; a 6bed4 Address is called General if it addresses a 6bed4 server.

6bed4-Prefixed Address: An IPv6 address that begins with the standard prefix TBD1::/32 that defines it the rest of the address as holding a General and (usually) a Direct 6bed4 Address.

6bed4 Peer: An endpoint in the 6bed4 communication path. This is where IPv6 traffic is wrapped into 6bed4 Frames, or taken out of it. Note that configurations may exist where further communication over IPv6 takes place, but the peer remains the endpoint for 6bed4-embedded frames.

6bed4 Infrastrucure: The combination of 6bed4 Peers, 6bed4 Servers and routing announcements made to the Internet.

/n Prefix: When n is at most 32, we use this notation to refer to an IPv4 prefix. When n is over 32, we use this notation to refer to an IPv6 prefix.

/32+n Prefix: This is always an IPv6 prefix, with a length of 32+n bits, with n at least 0 and at most 32. This notation improves readability when /32+n prefixes in a 6bed4-Prefixed Address are compared to /n prefixes in IPv4 addresses.

Native IPv6 Address: An IPv6 address that is not routed over 6bed4, Teredo [RFC4380] or 6to4 [RFC3056]. In other words, an IPv6 address that matches none of the prefixes TBD1::/32, 2001:0000::/32, 2002::/16. Synonyms used in this document are Native Address, or even just "native".

Internal Address: An IPv4 address as specified in [RFC1918] or [RFC6598], or an IPv6 address specified as Link-Local in [RFC4291].

Public Address: An address as observed by any arbitrary observer on the Internet, so outside of any layer of Network Address Translation. Formally, it is defined as any address that is not an internal address.

UDP/IPv4 Stream: A term that loosely refers to the back-and-forth exchanges of UDP frames over IPv4 between two specific 6bed4 Addresses. Note that the 6bed4 protocol sends keepalive messages to keep a started stream open with respect to filtering and address/port translation.

6bed4 Connection: The exchange of 6bed4 Frames between two 6bed4 endpoints. This is run over an UDP/IPv4 stream. When the endpoints of that stream are both Direct 6bed4 Addresses, then the 6bed4 connection is also called direct. Otherwise, the 6bed4 connection may be referred to as a general connection to make explicit that frames travel through at least one 6bed4 server.

Endpoint-Dependent Mapping: Any mapping between 6bed4 Addresses that is not an endpoint-independent mapping according to [RFC4787]. In terms of this RFC, for some values of Y2:y2 the value of X1':x1' differs from X2':x2'.

2. Wire Format of 6bed4 Frames

A 6bed4 Frame is constructed from an IPv6 frame by embedding it into UDP and IPv4. In that form, it is transmitted over the Internet. The source IPv4 address and source UDP port together are referred to as the frame's Source 6bed4 Address; the destination IPv4 address and destination UDP port together are referred to as the frame's Destination 6bed4 Address.

The embedded contents of a 6bed4 Frame always represents an IPv6 frame. The source IPv6 address encodes the Source 6bed4 Address, and the destination IPv6 address encodes the Destination 6bed4 Address. In fact, there are two variants of each 6bed4 Address depending on how the destination is reached, namely a Direct 6bed4 Address and a General 6bed4 Address; both are included in the IPv6 addresses.

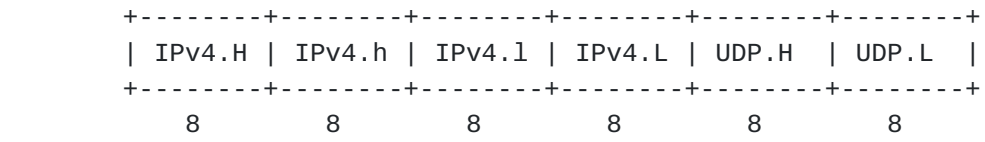
Since 6bed4 embedding supports general IPv6 traffic, there is a well-defined form for carrying around ICMPv6 messages, including Neighbor Discovery messages. These are actively used to negotiate with 6bed4 Peers. Router Solicitation and Advertisement is used to determine a public IPv6 address; Neighbor Solicitation and Advertisement is used to test and confirm if a remote 6bed4 Peer can be reached at the Direct 6bed4 Address.

Router Advertisements in 6bed4 Frames always announce /114 prefixes. This leaves 14 bits that can be used locally to distinguish hosts under the same prefix, possibly managed through static assignment or DHCPv6. Note that the completion with all zero bits indicates [RFC4291] the router, or more precisely, the sender of the Router Advertisement.

3. 6bed4-Prefixed Address Format

A 6bed4 Address is in concept a pair of an IPv4 address and a UDP port. A representation of this information as drawn below would show these parts in that order, and each represented in network byte order. Every 6bed4-Prefixed Address is known to at least contain a General 6bed4 Address; furthermore, somewhat dependent on a few flags as explained below, the same address usually also holds a Direct 6bed4 Address.

Conceptual Representation of a 6bed4 Address:



In places where link-scoped addresses [RFC4291] are used, they look like a standard fe80::/10 prefix followed by zero bit padding and ending in the six bytes IPv4.H, IPv4.h, IPv4.l, IPv4.L, UDP.H and UDP.L.

The representation of the General 6bed4 Address is founded on the knowledge that each 6bed4 Server runs on the standard UDP port TBD2. This information is not mentioned in the 6bed4-Prefixed Address, but the IPv4 address is completely shown in the top 64 bits of the 6bed4-Prefixed Address. The resulting sequence is:

- The four bytes IPv4.H, IPv4.h, IPv4.l and IPv4.L.

Assuming it is there, the representation of the Direct 6bed4 Address is also not entirely straightforward; it goes into the lower half of the 6bed4-Prefixed Address, but it must respect the two bits that must be set in the EUI-64 interface identifier (Section 4.2.1 of [RFC4291]). The bits in those position are placed behind the rest of the Direct 6bed4 Address. The resulting sequence in the representation is, in network byte order:

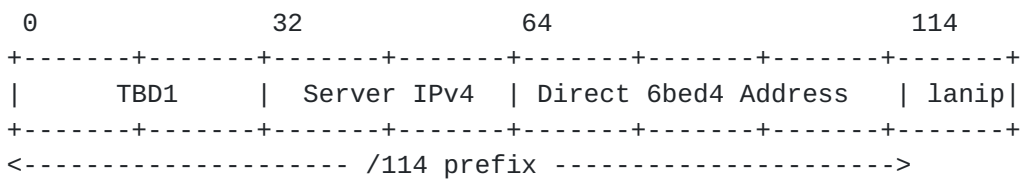
- IPv4.H with the lowest two bits replaced by the EUI-64 required bits;

- The three bytes IPv4.h, IPv4.l and IPv4.L;
- The two bytes UDP.H and UDP.L;
- The overruled 2 bits from IPv4.H.

The last part of the 6bed4-Prefixed Address concerns the 14 bits that follow after the /114 from the Router Advertisement. This part is left to the local 6bed4 endpoint to fill. The only thing to be mindful about is that the continuation with all zero bits is reserved (Section 2.6.1 of [RFC4291]) for the on-link address of the router that sent the prefix. This part will be denoted as "lanip" in the following diagram.

The general format of a 6bed4-Prefixed address that can be embedded as 6bed4 is as follows:

The IPv6 address for 6bed4:



The well-known TBD1::/32 prefix is important; it enables software to recognise this format in IPv6 addresses, and realise that traffic may be sent to either of the 6bed4 Addresses contained in the IPv6 address.

Note how it takes a /32+32 prefix to describe the route to a particular 6bed4 Server; however, when the IPv4 address of this server is contained in a /16 prefix it is possible to publish a /32+16 prefix, which is rather common. More on global routing follows in Section 7.

The prefix TBD1::/32 and port TBD1 are only reserved for a period of ten years, starting at the date of publication of this specification. Implementations that are aware of time MUST NOT implement 6bed4 after the final date; an extension of the final date is only possible through an updating RFC that at least passes through expert review for both allocated resources. This supports five years for new applications to roll out IPv6 applications based on 6bed4, and another five years for the use of 6bed4 to cease to exist and native IPv6 to replace it. Effectively, 6bed4 supports the assumption of IPv6 available anywhere, which should help the shift from IPv4 to IPV6.

This specification ensures that a 6bed4 Server can always be reached, and will always receive the frames sent to its /64 prefix. It cannot place constraints on the routers that act as intermediaries, though.

3.1. Invitations for Direct Connections

One special condition in a 6bed4-Prefixed IPv6 Address can be seen as an invitation for Direct 6bed4 Connections, namely when the General and Direct 6bed4 Address are the same. In terms of the address format, this means that the IPv4 addresses in each are the same, and that the UDP port in the Direct 6bed4 Address is the 6bed4 standard port TBD2.

This special format can be exploited to always make Direct 6bed4 Connections to this 6bed4-Prefixed Address; details are given in Section 8.4.

3.2. Meaning of EUI-64 flag bits

Note that it would be possible to assign meaning to non-zero values of the two EUI-64 flag bits, but this specification does not detail how. It could however be useful in an extension to 6bed4; multicast media streams are more likely with the larger address space of IPv6, and 6bed4 could facilitate it because it makes IPv6 possible everywhere and because its peers could attempt to spread the load through some clever application-layer protocol. The EUI-64 flags could also be useful to support globally unique EUI-64 identifiers when the General 6bed4 Address can address hosts locally.

Mindful of the current specification and these possible extensions, this specification states that the two flag bits SHOULD be set to 0. The lower half of the IPv6 address MUST NOT be interpreted to contain a Direct 6bed4 Address if the local/unique flag is set to indicate uniqueness; in that case, it MAY be interpreted as a modified EUI-64 address and potentially used for local routing if the upper half of the IPv6 address is in local use. The unicast/multicast flag MUST be ignored until this specification is refined with details of its interpretation towards multicast facilities.

4. Network Infrastructure and Protocol Overview

This specification does not refer to parts of the 6bed4 Infrastructure as tunnel clients and tunnel servers, but rather as 6bed4 Peers and 6bed4 Servers. This reflects the intention of making the endpoints, or peers, use Direct UDP/IPv4 Streams as their preferred transport for 6bed4 Connections.

The purpose of a 6bed4 Server is to provide information about the Public IPv6 Address of a 6bed4 Peer, and to permit fallback to General 6bed4 Streams as a transport mechanism between 6bed4 Peers between which Direct UDP/IPv4 Streams are not feasible. Finally, the 6bed4 Server is needed to connect a host that can only do 6bed4 to one that can only do Native IPv6 Addresses; precautions to avoid that situation wherever possible follow.

To be reachable to the outside world on a Public IPv6 Address, a 6bed4 Peer sends a Router Solicitation to a 6bed4 Server, and receives a Router Advertisement with a /114 prefix in response. As shown in Section 2, the returned prefix includes both the General and Direct 6bed4 Address for the local endpoint, and leaves some room for local address assignment. It is then the local 6bed4 Peer's task to keep open the UDP/IPv4 Stream to the 6bed4 Server, so as to ensure that the IPv6 address remains constant. If it were to change, then the server would respond to sent traffic with a new Router Advertisement, offering a new /114 prefix and retracting the previous one. This situation is usually avoided by sending regular keepalive messages.

When initiating a new 6bed4 Connection, in other words when first contacting a remote 6bed4 Peer, it is also possible to send a Router Solicitation to the General 6bed4 Address found in the remote 6bed4 Address. This results in an alternate 6bed4-Prefixed Address that can be setup locally for that contact attempt, avoiding trapezium-shaped traffic between 6bed4 Peers. This specification ensures that this mechanism will always work, but a similar responsibility to keep the link to the remote 6bed4 Server open falls upon the local system. To further optimise traffic, it is also possible to send a Router Advertisement to the Direct 6bed4 Address contained in the remote IPv6 address; this may work and entirely remove the dependency on a 6bed4 Server. It is up to the application which options are tried, an in which order. Note that it is not generally possible to rely on certain connectivity between 6bed4 Peers without using at least one 6bed4 Server per 6bed4 Peer as a fallback mechanism to reach it.

Finally, for the most agressive approach towards peer-to-peer connections, it is even possible to send the Router Solicitation to the Direct 6bed4 Address of the approached peer. Note that it is also possible for peers inviting this method to copy their Direct 6bed4 Address in the General 6bed4 Address, and ensure that traffic always ends up on the 6bed4 endpoint. It should be noted that it is not generally possible to rely on just trying the Direct 6bed4 Address if certainty of the connection is required; the General 6bed4 Address MUST also be used to achieve this certainty, except when it is the same as the Direct 6bed4 Address, in which case the Router Solicitations would be the same.

At any time before or during communication with a 6bed4 Peer through a 6bed4 Server, it is possible to try using a Direct UDP/IPv4 Stream by sending a Neighbor Solicitation from a local Direct 6bed4 Address to the peer's Direct 6bed4 Address, and observing if it is matched by a Neighbor Advertisement over the opposite path. Note that "matched" means that it MUST be certain that the Neighbor Advertisement was not sent in response to a Neighbor Solicition that went out over another channel. In Section 8 a few alternatives to this standard method are presented.

When such direct requests lead to matching direct responses, then it is safe to assume that a Direct UDP/IPv4 Stream is possible between the 6bed4 Peers, at least for some time following. This is because UDP/IPv4 has worked in both directions, and the fact that it contained ICMPv6 cannot be inferred by the intermediate NAT routers or firewalls because the UDP format does not tag its contents. To ensure that this assumption continues to hold, firewalls and NAT routers MUST NOT give frames from and to the standard 6bed4 port TBD2 any special treatment.

The knowledge that a remote 6bed4 Address can be reached over a certain UDP/IPv4 Stream MUST NOT be assumed to also apply to communication with any other IPv6 Address. This is vital, as it evades the trap of making inductive assumptions about the behaviour of NAT or firewalls. See Section 5 for details of the deductive approach of 6bed4 regarding regarding NAT and firewalls.

It is common for IPv6 hosts to have multiple IPv6 addresses, and so the question which local and remote addresses to use. This has been answered in [RFC6724]. In relation to 6bed4, native-to-native traffic MUST precede 6bed4-to-6bed4. In addition, 6bed4-to-6bed4 SHOULD be preferred over either native-to-6bed4 and 6bed4-to-native, because it is more desirable to use a Direct UDP/IPv4 Stream than a General UDP/IPv4 Stream through a 6bed4 Router. Every 6bed4 Router is ultimately a shared resource and thereby a potential bottle neck for routing efficiency. This choice implies that 6bed4 Routers are offloaded. Local interface and routing table configurations are usually the places to configure these rules.

In addition to this optimisation, higher-layer applications MAY also incorporate knownledge of 6bed4; for instance, a SIP proxy [RFC3261] could observe a remote peer's 6bed4-Prefixed Address and offer media exchange over a locally available 6bed4-Prefixed Address instead of a Native IPv6 Address, once more so as to offload the shared resource of a 6bed4 Server, and to achieve lower roundtrip delays and jitter. It could even aim to construct a more optimal 6bed4-Prefixed Address for the connection using the techniques mentioned above.

5. NAT and Firewall Traversal

There is a wide variety of NAT router implementations, each with subtly different characteristics. A similar thing applies to firewalls. This means that any approach to peering through NAT and firewalls that is to work everywhere must be founded on deductive reasoning-from-facts, and not induce information from an incomplete survey of NAT routers and firewalls. Specifically, STUN [RFC3489] used to make such inductions which were later relativated (Section 14.3 [RFC5389]). An IPv6 tunneling mechanism based on this work [RFC4380] has indeed shown to not work in general [POTAROO].

Nothing in 6bed4 makes assumptions like a behavioural classification of NAT routers or firewalls. Instead, 6bed4 simply tries if IPv6 traffic between peers is possible by sending a frame directly to a peer and matching it with a return frame. Since these are carried as part of a UDP/IPv4 stream with no content tagging, a NAT router cannot treat this test exchange any differently from "real" data, and so it is possible to exchange traffic directly for as long as the UDP /IPv4 stream is considered to be mapped. This may be for a limited time, but that time can be extended through keepalive messages if so desired.

Firewalls and NAT routers are assumed to not inspect the contents of an UDP/IPv4 stream. This is a fair assumption because of the lacking content tagging in UDP. Network components that alter the contents of UDP frames have been reported [RFC4380] but are downright broken. They may need to be replaced before 6bed4 can function; unlike Teredo, 6bed4 will not reduce the risk of running into what are bugs by obfuscation of addresses and ports contained in 6bed4 Frames.

This specification assumes that outgoing UDP is supported. This may not work for every service that states to offer Internet service, but such statements are false and will lead to more functional problems. The only imagined exception to the assumption of functioning UDP is due to manual override, where an local administrator has opted to control UDP traffic based on port numbers. In such cases, the administrator must be contacted to manually install support for 6bed4, precisely as s/he desires it.

Note however, that the inability to recognise UDP traffic in lieu of content tagging so the firmware in NAT routers and firewalls cannot make any assumptions on the contents of UDP frames, and so it is not possible to filter UDP traffic with a general mechanism. As a result, an entire NAT router or firewall is limited to one default behaviour for all UDP traffic. Otherwise, UDP as a general protocol would fail.

When sending UDP out through a NAT router, it will usually substitute the source IPv4 address and UDP port with an external IPv4 address and UDP port. For UDP protocols to behave normally in the presence of this translation, the same substitutions must applied when future frames are sent over the same UDP stream, so some mapping between internal and external IPv4 address and UDP port can safely be assumed. The NAT router can select values from the source and destination IPv4 addresses and UDP ports to lookup what mapping to apply, but as long as all these values match, the same UDP/IPv4 stream is recognised and the same mapping must be applied.

Firewalls have a similar mapping, albeit not for substituting an address and port, but to recognise if traffic has been sent out over a UDP/IPv4 stream. Other than this difference, it generally behaves under the same constraints as a mapping in a NAT router.

To handle all mappings that are possible, 6bed4 makes no assumption about sharing the same mapping for different UDP/IPv4 Streams; a Public IPv4 Address and UDP Port may be the same for two remote peers, or they may differ. As far as 6bed4 concerns, they may be separately administered mappings. And if they happen to overlap then it is still safe to treat them separately, as the protocol components are idempotent.

UDP has no formal end marker for a UDP/IPv4 Stream, so the only thing that a router can do is guess when a stream has ended. It is assumed by this specification that a mapping is kept active for a minimum time, and revived at least when traffic is sent out along the stream, as that is the only thing on which an implementation can base such revivals without making UDP insecure. To overcome timeouts on such mappings, a keepalive mechanism is needed in a number of places, and the timer triggering that mechanism must be a setting that the end user can influence. Experiments [RFC4380] have shown that a default setting of 30 seconds is quite likely to work. To avoid being presumptious, implementations SHOULD permit user configuration of this timeout value.

UDP in general must support bidirectional streams, so when outbound traffic is passed over an UDP/IPv4 Stream, then reply traffic within that same same stream must also be accepted. Otherwise, a firewall or NAT router would break the UDP protocol. Note once more that there is no tagging of content in UDP, so there can only be one default policy. There is only one way that a NAT router can redirect such traffic internally, and that is by applying the mapping for the UDP/IPv4 stream in reverse. This will send the reply traffic back to the internal host that initiated the outbound side of the UDP/IPv4 stream, so the damage done to the UDP/IPv4 Stream is unnotice to the local endpoint (as long as it does not make its endpoint coordinates explicit to the outside World, of course).

6. Filtering 6bed4 Traffic

Any tunnel should guard against being abused for claims on addresses "inside" the tunnel based on clients "outside" the tunnel that should not be able to make such claims. In the case of 6bed4, this means that the 6bed4 Address from which a 6bed4 Frame arrives must match with the Sender IPv6 Address.

When a filter rejects a 6bed4 Frame on account of a mismatch between the Source 6bed4 Address and the suggested IPv6 address, it should respond with a Router Advertisement that retracts the /114 prefix used in the IPv6 address, and offer a new /114 prefix that would have matched to replace it.

Both 6bed4 servers and 6bed4 Peers MUST silently discard any 6bed4-embedded frame that is not an proper IPv6 message; example conditions to implement that would be:

- the embedded frame length is less than that of an IPv6 header
- the embedded frame does not start with nibble 6
- the embedded frame length differs from the total IPv6 frame length
- there are checksum errors in TCP, ICMPv6 or UDP headers

If the recipient is a 6bed4 Server, then the Source 6bed4 Address of an acceptable 6bed4 Frame MUST match one of the following:

- the Direct 6bed4 Address embedded in the Source 6bed4 Address;
- the General 6bed4 Address of a 6bed4 Server known to announce TBD1::/32, but only if the recipient's 6bed4 Address matches the Destination 6bed4 Address of the 6bed4 Frame.

Note that 6bed4 Frames may occasionally travel through a router announced for TBD1::/32 but not when they are sent from a 6bed4-Prefixed Address, in which case it would have been forwarded to a 6bed4 Address.

If the recipient is a 6bed4 Peer, then the Source 6bed4 Address of an acceptable 6bed4 Frame MUST match one of the following:

- the Direct 6bed4 Address embedded in the Source IPv6 Address, but this variation fails if it can be retrieved;
- the General 6bed4 Address embedded in the Source IPv6 Address.

In addition, the Destination 6bed4 Address of an acceptable 6bed4 Frame MUST match the following:

- the Direct 6bed4 Address in the Desintation IPv6 Address, but this check also succeeds if it can be retrieved.

Finally, the destination IPv6 address SHOULD be assigned by the 6bed4 Server set as the Generic 6bed4 Address as part of the destination IPv6 address.

In cases where the lower half of the IPv6 address indicates a globally unique lower half, no Direct 6bed4 Address can be retrieved from the IPv6 address, and so it is not possible to make the aforementioned filtering matches. A remote peer could still accept a 6bed4 Frame on account of the General 6bed4 Address in the IPv6 address, but it MUST NOT try to read a Direct 6bed4 Address from the IPv6 address. In situations like these the 6bed4 Server would follow other rules because it would be aware of the way to forward (and receive) 6bed4 Frames from 6bed4 Peers.

A 6bed4 Server MAY apply additional filtering to limit its use to a particular subset of 6bed4 Peers; for instance, the users of an ISP or a commercial 6bed4 service. To this end, it would ensure that 6bed4 Frames are passed either from or to a Direct 6bed4 Address that is accepted. This MUST NOT be done on a 6bed4 Server that announces routability for the TBD1::/32 prefix.

The only messages exempted from all aforementioned filtering are Router Solicitation frames targeting a link-local IPv6 address, and having their IPv6 Hop Limit set to 255. Those messages receive a Router Advertisement frame in reply, containing the /114 prefix that matches the 6bed4 Address of the originator of the solicitation. Specifically note that even a commercially exploited 6bed4 Server MUST welcome Router Solicitation from unaffiliated peers that intend to avoid trapezium-shaped routing.

7. Routing 6bed4 Traffic

In backbone networks, IPv6 connectivity providers exchange routes through the Border Gateway Protocol (BGP). There is no standard for prefix lengths supported, and principally everyone makes their own filtering rules. In practice [ref] however, a /48 is a mostly supported length, unlike anything that is longer. The small percentage that does not support the /48 is usually capable of routing through a covering /32 IPv6 prefix. The best approach to global routeability appears to be the combination of a /32+16 with a covering /32+0 prefix.

One MUST own the IPv4 range over which routing is announced, so the announcement of a /32+16 is only possible for a 6bed4 Address that falls in a /16 prefix that is owned by the anonymous system that runs the 6bed4 server. An exception is made for the /32+0 prefix, which serves as a fallback routing facility. This fallback range can be used as anycast, so this MAY be announced by any party implementing a 6bed4 server. What this means is that control over return traffic is only possible through announcement of the /32+16, with a minute portion of the Internet still routing it through the /32 announcement.

Note that routing practice [ref] indicates that there is no global routing use in announcing prefixes longer than /32+16 if a /16 prefix is not held around the 6bed4 server; such longer prefixes usually do not make it to the backbone. Also note that there is no use in announcing shorter prefixes than /32+16 if a shorter prefix than a /16 is available around the 6bed4 server address. This is because only a sinlge IPv4 address is addressed, and because the /32+16 prefix is the most likely to make it to the backbone.

Anyone announcing a /32+16 MAY also announce the /32+0; anyone announcing the /32+0 MUST respond to all non-local 6bed4 traffic (targeted at 6bed4 server addresses outside locally administered IPv4 ranges) by relaying the traffic to the General 6bed4 Address embedded in the top half of the IPv6 destination address and the sending server's address as the source address [TODO:why_accept?]. This relaying is done inside a 6bed4 Frame over UDP, with the default 6bed4 port as both the source and destination port.

Announced /32+16 prefixes SHOULD be retracted during downtime of the 6bed4 service; announced /32 prefixes MUST be retracted during downtime of the 6bed4 service. It is advised to relay routing messages that establish these routes through the 6bed4 service, to the effect that service downtime implies route announcement retraction. This would not normally interfere with intentions to setup redundant 6bed4 service.

7.1. Network Endpoint Routing Options

Note that the availability of /32+0 announcements as an anycasted service makes it possible for any node with a single IPv4 address and access to the 6bed4 port to offer 6bed4 service. The quality level of this service is lower due to the dependency on the anycast range for more than just fallback routing. Having said that, it does enable a model where an endpoint can run a 6bed4 service to cover a local network. In this case it is possible to use the local numbering scheme, as long as the respective bit in the EUI-64 address is zero. This specifically implies that Stateless Address Autoconfiguration cannot be used to assign a 6bed4 Address on such a LAN, but DHCPv6 or manual schemes, or local-bit-flipping autoconfiguration could all work.

8. Opportunistic Peering

An explicit design goal of 6bed4 is to exploit Direct UDP/IPv4 Streams between 6bed4 Peers. This is achieved by simply trying an exchange over UDP/IPv4 and relying on the uninterpreted transport of UDP payloads to infer that the entire UDP/IPv4 stream must be possible if a single exchange works bidirectionally.

It is vitally important that UDP/IPv4 Streams follow the same path in both directions. This is because it is only safe to assume that outgoing traffic in a UDP/IPv4 Stream keeps NAT router and firewalls open for return traffic. Specifically, these network components need to see the same UDP ports and IPv4 addresses for the traffic in both directions, albeit with their source and destination roles exchanged. To ensure this, a 6bed4 Peer that receives direct traffic from a remote peer that it would address through a 6bed4 server MUST start one of the following methods to setup direct peering. In such cases, it may not give up before it has tried the Neighbor Discovery method, which is the only obliged method.

The one thing left to specify is how peering is initiated. This is done by one of the following methods to setup direct peering, applied opportunistically. This can basically be done at any time deemed fitting, but it is suggested to employ some form of rate limiting for opportunistic peering attempts. A few useful places to do this would be to send a Router Advertisement to a direct 6bed4 Address before sending it to its general 6bed4 Address; or to send Neighbor Discovery sometime during the normal exchange; or to do it during the setup of a TCP/IPv6 stream.

The following subsections introduce a general framework for setting up direct peering. It continues with a few concrete examples, some of which must be implemented with 6bed4. There are certainly

opportunities for other specific implementations of the general opportunistic mechanism.

8.1. Abstract Framework for Direct Peering

The abstract framework for direct peering setup assumes that 6bed4 Frames may be lost, a common assumption when routing frames over the Internet. This assumption allows the 6bed4 Peer to simply try sending a frame directly, without having to maintain resending queues. Usually, lost frames will be resent by higher protocol layers, at which time they would be sent through the 6bed4 Server. Alternatively, extra frames can be generated in the 6bed4 Peer.

Another aspect of the abstract framework is that it must be possible to relate an incoming direct reply to an outgoing direct request, and only a direct outgoing request. This can be established with a nonce if a frame is generated for the purposes of 6bed4 Peer detection; or if a higher protocol layer generates the message, then the time between the first send and a later re-send is the window during which the direct response is considered a reliable sign of bidirectional traffic.

8.2. Neighbor-based Direct Peering

One mechanism that MUST be implemented in all 6bed4 Peers is that of peering setup through Neighbor Discovery. This involves a Neighbor Solictation generated by the 6bed4 Peer over a Direct UDP/IPv4 Stream, and receiving a reply in the form of a Neighbor Advertisement. In short, this is standaard reachability detection for IPv6. The usual ICMPv6 requirement of a Hop Limit equal to 255 applies here, and is not invalidated by the underlying UDP/IPv4 transport.

In order to ensure that a response matches a request sent directly, as well as to thwart attemts to overtake traffic, a nonce following [RFC6496] MUST be sent as part of the Neighbor Solicitation. A 6bed4 Peer MUST respond to Neighbor Solicitation with Hop Limit 255 and its own address information as destination with a Neighbor Advertisement with Hop Limit 255, and it MUST include the nonce it found in the Neighbor Solicitation, if any.

A 6bed4 Peer wanting to initiate a Direct 6bed4 Connection may use this mechanism at any time; it is safe because it does not interfere with the actual data stream between the peers. These internally generated Neighbor Discovery messages SHOULD NOT be sent to the 6bed4 Server. Only when a direct Neighbor Solicitation results in a Neighbor Advertisement with the same nonce may the originator of the exchange conclude that a Direct 6bed4 Connection can be used. The

remote peer MUST NOT draw that conclusion, as it cannot be sure if the direct Neighbor Advertisement arrived. Also in the interest of security, it should initiate its own exchange.

8.3. Routing-based Direct Peering

Another mechanism that SHOULD be implemented in 6bed4 Peers, and that MUST be implemented in 6bed4 Servers, is responding to Router Discovery messages with the Hop Limit set to 255. The response should be composed of the /64 prefix that applies to the 6bed4 server in use, and the lower half should be composed from the 6bed4 Address over which the request was received. If a nonce is included, it MUST be handled as for Neighbor Discovery. A 6bed4 Peer MUST include a nonce if it sends out a Router Discovery message.

8.4. Invitation-based Direct Peering

The simplest possible form of opportunistic direct routing is when the 6bed4-Prefixed address of a remote peer has the same values set as its General and Direct 6bed4 Address. This means that the opportunistic route can be tried immediately, as the 6bed4 Router function requires accessibility over the standard UDP port TBD2 that is apparently also in use for Direct 6bed4 Connections.

Note that it is usually necessary to obtain a 6bed4-Prefixed Address to use for communication with such a party, so if no local IPv6 address with the same /64 as the remote peer's IPv6 address exists, then one should be requested from the remote peer through Router Solicitation. This will inform the local peer how its public address looks to the inviting peer.

8.5. TCP-based Direct Peering

This mechanism MAY be incorporated into 6bed4 implementations that run TCP over 6bed4. It piggy-backs on the SYN and ACK flags exchanged while setting up a TCP connection to a remote 6bed4 Peer, and introduces peering opportunistically and, quite possibly, without any explicit messaging for setting it up.

TCP-based peering is based on the SYN flag sent initially by a new TCP connection being setup, and the ACK flag sent to acknowledge it. Furthermore, it assumes that the TCP stack wil resend a failed first SYN attempt.

The first SYN sent to a remote 6bed4 Peer would be sent to the Direct 6bed4 Address of the remote peer; if it reports back with the ACK flag set, then a direct connection was clearly possible. If this response does not arrive before the TCP-stack re-sends the SYN fame,

then this and further attempts are sent to the General 6bed4 Address, until further inspiration triggers another attempt at direct peering.

The endpoint information in TCP as well as window offsets are used to recognise the SYN attempt, and later pairing the ACK to it. This means that the conditions are available for certainty that a direct attempt is paired with a direct response.

Note that this mechanism works in both directions; as the passive side responds to a SYN with ACK, it usually sends its own SYN flag in hope of an ACK back from the active side. This second exchange can follow the same rules to detect bidirectional connectivity from the other side. Sending that first ACK along with the second SYN also means that a first attempt is made through direct peering.

This facility is useful for servers that have configured their NAT and Firewalls to open a UDP port, so any direct contact attempts are certain to succeed. This can be used to construct a fixed 6bed4 Address, which would be suitable for publication in DNS. Although the Invitations of Section 8.4 are a special form of this approach, the TCP-based approach covers many more situations.

8.6. SCTP-based Direct Peering

The Opportunistic Peering method for SCTP [RFC4960] is a variation on the method for TCP. Note that SCTP does not have the usual acceptance problems when used between 6bed4 Peers, since 6bed4 traffic is largely unfiltered and tunnels through the NAT routers that currently tend to block this protocol.

SCTP follows an association initiation protocol with four frames instead of the three of TCP. The last two may carry data chunks, but this would be stored in a separate chunk, permitting the network layer some freedom to manipulate the frames if desired.

The advantage of four messages over three is that the first chunk (INIT) can be sent first to the Direct 6bed4 Address, then to the General 6bed4 Address of the remote peer, which can then assume that outgoing UDP traffic has opened a hole in NAT and Firewalls. This means that the chances of entry for a reply message are optimal. Therefore, the opportunistic method for SCTP relies on the remaining three chunks of association initiation.

The INIT-ACK chunk counts as an opportunistic attempt from the remote to the local peer when it arrives over the Direct UDP/IPv4 Stream; the COOKIE chunk sent over the Direct UDP/IPv4 stream signals acceptance to the remote peer and is at the same time an opportunistic attempt from the local peer to the remote; the COOKIE

ECHO chunk sent over the Direct UDP/IPv4 Stream signals acceptance to the local peer.

In short, new SCTP associations can send the first attempt of each of the chunks for association initiation to the Direct 6bed4 Address of a peer. With the exception of INIT, the arrival of a chunk over the General UDP/IPv4 Stream indicates that the SCTP association must continue through the 6bed4 Server.

8.7. SIP-supported Direct Peering

Another example of opportunistic peering that can be advantageous for a specific application is SIP [RFC3261]. A SIP exchange consists of a request and one or more responses. The application software is well aware of the distinction between those, and can prove useful to facilitate 6bed4 peering if it integrates with the network layers.

SIP messages, when sent over UDP, are prone to resends if a response is not received for some time. As a result, it is possible to first attempt sending to the remote peer's Direct 6bed4 Address, and send later retries to the General 6bed4 Address. As with TCP, it is possible to treat as proof of direct peering any incoming direct responses between the intial request and its re-sends through the General 6bed4 Address.

The link between a SIP request and its response is easily made with the identifying parts of the message; these are embedded in text and would thus rely on application integration with 6bed4. The parts are contained in the Call-ID: header, From: tag and optional To: tag. In addition, the top Via: header contains a branch parameter that identifies the exchange. All this is the knowledge domain of the application, and could take too much from the 6bed4 tunneling code, so instead of support such protocols directly it is a sign that an extended API between the 6bed4 stack and the application can be advantageous.

9. Requirements for 6bed4 Infrastructure Components

This section describes minimum requirements for 6bed4 Servers and 6bed4 Peers.

9.1. Requirements for 6bed4 Servers

A 6bed4 Server MUST respond to well-formed Router Solicitations from any 6bed4 Peer, including non-local and non-member requests, by advertising the /114 prefix that starts with the well-known prefix TBD1::/32, followed by its own 6bed4 Address in the position of the General 6bed4 Address, and the requesting 6bed4 Address as the Direct

6bed4 Address. The 6bed4 Server MUST know its own 6bed4 Address as a combination of a configured public IPv4 address and the UDP port TBD2 and it MUST be reachable for anyone at the resulting 6bed4 Address.

A 6bed4 Server MUST filter incoming 6bed4 Frames as specified in Section 6. When a 6bed4 Frame is rejected on account of its Source 6bed4 Address, then the same Router Advertisement MUST be sent in response, with the extension that the falsely assumed /114 prefix from the source IPv6 address is also retracted.

When responding to Neighbor Solicitation or Router Solicitation, a 6bed4 Server MUST copy any nonce and timing information from the request into the response.

A 6bed4 Server MAY publish prefixes of its IPv6 address with lengths between /32+0 and /32+32 through the Border Gateway Protocol (BGP) with at least the 6bed4 Address that it assigns in Router Advertisements as the General 6bed4 Address, and MAY be published in BGP with a larger prefix. The 6bed4 Server MUST receive all native IPv6 traffic sent to this published prefix.

A 6bed4 Server SHOULD forward IPv6 frames sent to 6bed4-Prefixed Addresses after embedding them in a 6bed4 Frame; the Destination 6bed4 Address is the Direct 6bed4 Address if the General 6bed4 Address matches the server's 6bed4 Address, or otherwise the Destination 6bed4 Address is the General 6bed4 Address.

A 6bed4 Server SHOULD forward 6bed4 Frames destined for IPv6 Addresses that do not fall under the TBD1::/32 prefix. This is done by unpacking the 6bed4 Frames, or in other words, by removing the UDP and IPv4 headers.

A 6bed4 Server SHOULD forward 6bed4 Frames destined for a 6bed4-Prefixed Address to the next hop 6bed4 Address. If the destination IPv6 Address holds the 6bed4 Server's address as the General 6bed4 Address, then the next hop is the Direct 6bed4 Address found in the destination IPv6 address. Otherwise, the next hop is the General 6bed4 Address found in the destination IPv6 address.

The "SHOULD" conformance level in the last three paragraphs is a conscious limitation of the service, in support for commercial 6bed4 offerings. However, if the 6bed4 Server announces router prefixes of at least 32 bits valued TBD1, then any IPv6 Frames whois source or destination IPv6 Address matches one such prefix MUST be forwarded as described by the last three paragraphs.

9.2. Requirements for 6bed4 Peers

A 6bed4 Peer SHOULD respond to well-formed Router Solicitations from any 6bed4 Peer, and to invalid incoming 6bed4 Frames from any source with a Router Advertisement, by advertising the /114 prefix that starts with the well-known prefix TBD1::/32, followed by its own 6bed4 Address in the position of the General 6bed4 Address, and the requesting 6bed4 Address as the Direct 6bed4 Address. The 6bed4 Peer may obtain its own 6bed4 Address from the Destination 6bed4 Address of the incoming 6bed4 Frame.

A 6bed4 Peer MUST filter incoming 6bed4 Frames as specified in Section 6. When a 6bed4 Frame is rejected on account of its Source 6bed4 Address, then the same Router Advertisement MUST be sent in response, with the extension that the falsely assumed /114 prefix from the source IPv6 address is also retracted.

When setting up a 6bed4 Connection to a remote 6bed4 Address, a 6bed4 Peer SHOULD prefer any address for the local side of the 6bed4 Connection that starts with the same /64 prefix as the remote 6bed4 Peer's. If no such address is available locally, it is RECOMMENDED to acquire one by sending a Router Solicitation to the remote peer's General and/or Direct 6bed4 Address.

A 6bed4 Peer MUST respond to received Neighbor Solicitation messages with Neighbor Discovery messages, to implement Opportunistic Peering as specified in Section 8.1. It MAY also implement other mechanisms for Opportunistic Peering.

When communicating over a 6bed4 Connection, it is RECOMMENDED that a 6bed4 Peer attempts to setup a Direct 6bed4 Connection according to the Optimistic Peering procedures described in Section 8.1.

When receiving 6bed4 Frames over a Direct 6bed4 Connection when not being setup to perform direct 6bed4 Peering to the Source 6bed4 Address, a 6bed4 Peer MUST work towards Optimistic Peering, and Neighbor Solicitation MUST be one of the opportunistic mechanisms tried before considering failure. If attempts towards Optimistic Peering are already in motion for the same remote 6bed4 Peer, then its result may be awaited first.

A 6bed4 Peer SHOULD send a nonce and timing information [RFC6496] in any sent Neighbor Solicitation and Router Solicitation.

When responding to Neighbor Solicitation or Router Solicitation, a 6bed4 Peer MUST copy any nonce and timing information from the request into the response.

A 6bed4 Peer SHOULD send keepalive frames to keep the UDP/IPv4 Stream open for active 6bed4 Connections. A keepalive frames MAY be a valid

IPv6 frame, or it may be an empty message embedded in UDP and IPv4 as would have been done for an IPv6 frame. It MAY be sent with an IPv4 Time To Live that is so low that the keepalive frame just makes it to the public Internet, after having crossed local NAT routers and firewalls. The timing interval of a keepalive frame SHOULD be a user setting, and it MAY by default be set to a safe low value of 30 seconds.

Finally, a 6bed4 Peer SHOULD deliver any 6bed4 Frames by unwrapping it (meaning, removing the IPv4 and UDP headers) and locally process the contained IPv6 frame, and it should accept local IPv6 frames originating at one of its local 6bed4-Prefixed Addresses, and wrap it into UDP and IPv4 to send to the destination, either to the Direct or General 6bed4 Address contained in the IPv6 destinationsaddres.

10. Implementation Concerns

One potential implementation of a 6bed4 tunnel interface would exploit the Neighbor Cache in an IPv6 host to facilitate storage and timing of the various neighboring relationships. Indeed, the timeouts of such relations are generally shorter than NAT mapping timeouts. It should however be noted that not all Neighbor Caches are designed for large-scale operation, and that an active host could choke on that. Furthermore, there is no keepalive mechanism built into such neighbor caches, which means that one or more peering relations could loose their address when no traffic is exchanged for some time.

For ISPs that do not provide 6bed4, there is no problem to reach out to more distant 6bed4 service providers. It is even thinkable that such a service would be offered on a commercial basis, and that traffic through the service would only be passed through when it either passes to or from a customer's registered address (range). Such a service provider obviously should not announce that it can route the TBD1::/32 prefix.

Although this specification speaks only of UDP as a transport for 6bed4, it is possible to add TCP as a fallback protocol if the code of the 6bed4 Peer and 6bed4 Server agree to that. Although SCTP would be more suitable, it is not likely to find a situation where that is present and UDP is not. What is common is that UDP is banned while TCP is not. Such cases could benefit from a fallback to TCP. The format of the messages exchanged would be precisely the same, except that the transmission happens to be ordered and guaranteed. As one more option to cator to failed Internet installations, one could even consider supporting TCP over port 80 or 443 instead of the standard port. In all cases, the 6bed4 Server should now be aware of all connected 6bed4 Peers, and choose to contact them over TCP

instead of UDP, as would normally be the case. This can be remedied in various ways, with differentiation through the 6bed4 Address of the server being the easiest.

Another option based on protocol extensions to which 6bed4 Server and 6bed4 Peer could agree, is to apply encryption to the information exchanged. Such facilities are not part of this specification.

A few situations call for coordination between 6bed4 Infrastructure components. This will take place under the domain name 6bed4.net for as long as the TBD1::/32 prefix and the TBD2 port are allocated for 6bed4. The precise methods are not detailed here because it does not concern the core communication protocols of the 6bed4 tunneling mechanism. One thing coordinated under 6bed4.net is an service where Autonomous Systems who announce TBD1::/32 routing must register all IPv4 addresses of 6bed4 Servers that could pass on 6bed4 traffic to a machine with a longer prefix that did not make it through to all parts of the Internet.

11. IANA Considerations

This specification reserves a 32-bit address prefix in the IPv6 address space for a period of ten years, starting from the date of publication of this specification.

This specification also reserves Port TBD2 from the pool of UDP ports and from the pool of TCP ports for a period of ten years, starting from the date of publication of this specification.

The prefix is TBD1::/32 and will be exclusively used for 6bed4 addresses. The port TBD2 will be the port on which 6bed4 Servers provide their services.

12. Security Considerations

Tunneling mechanisms must always be on their guard for wrapped packets containing false origins [RFC6169]. To shield against this, 6bed4 ensures that the source IPv4 address and UDP port of the together match either the Direct or General 6bed4 Address contained in the source IPv6 address.

Note that this facility works best when address filtering [BCP38] is applied. In lieu of authentication facilities for frame source, the best that the 6bed4 tunnel can do is to avoid worsening the problems of incomplete address filtering.

One exception arises with the possibility that a target 6bed4-Prefixed Address publishes a /32+16 prefix which is not seen

everywhere on the Internet; or that such a prefix is not actually published at all. In such situations, a 6bed4 Frame may be routed to a TBD1::/32 route, which passes it on to the General 6bed4 Address contained in the IPv6 destination address. The list of routers that can pass on such traffic will generally be limited, and will be maintained externally to this specification, but documented on 6bed4.net. Routes announced to larger IP space than owned by the publishing Anonymous System MUST be registered on 6bed4.net so as to distinguish them from abuse. The domain will publish processible information that helps 6bed4 Servers to recognise this distinction too.

It is important to realise that 6bed4 bypasses NAT and Firewalls. This is a feature inasfar as it enables peer-to-peer connectivity, but it also implies a responsibility to not lightly attach services to a 6bed4-Prefixed Address. There is no protection, other than what is being added. Specifically noteworthy is that the operator of the 6bed4 Server cannot implement filtering on behalf of their customers; the ability to use Direct 6bed4 Connections would bypass this, and since this could happen at any time such filtering could not even be realiably made connection-aware.

13. Normative References

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

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

[RFC3056] Carpenter, B. and K. Moore, "Connection of IPv6 Domains via IPv4 Clouds", RFC 3056, February 2001.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[RFC3489] Rosenberg, J., Weinberger, J., Huitema, C., and R. Mahy, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)", RFC 3489, March 2003.

[RFC4787] Audet, F. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, January 2007.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.

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

[RFC4380] Huitema, C., "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)", RFC 4380, February 2006.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.

[RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.

[RFC6169] Krishnan, S., Thaler, D., and J. Hoagland, "Security Concerns with IP Tunneling", RFC 6169, April 2011.

[RFC6496] Krishnan, S., Laganier, J., Bonola, M., and A. Garcia-Martinez, "Secure Proxy ND Support for SEcure Neighbor Discovery (SEND)", RFC 6496, February 2012.

[RFC6598] Weil, J., Kuarsingh, V., Donley, C., Liljenstolpe, C., and M. Azinger, "IANA-Reserved IPv4 Prefix for Shared Address Space", BCP 153, RFC 6598, April 2012.

[RFC6724] Thaler, D., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, September 2012.

[BCP38] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, May 2000.

[POTAROO] Huston, G., "Testing Teredo", April 2011, <http://www.potaroo.net/ispcol/2011-04/teredo.html>.

Author's Address

Rick van Rein
OpenFortress B.V.
Haarlebrink 5
Enschede, Overijssel 7544 WP
The Netherlands

Email: rick@openfortress.nl