Network Working Group R. Van Rein
Internet-Draft OpenFortress B.V.
Intended status: Informational July 9, 2020
Expires: January 10, 2021

6bed4: IPv6 Anywhere in support of Reliable Peering
draft-vanrein-6bed4-04

Abstract

The purpose of 6bed4 is to support IPv6-only networks, hosts and applications. It passes IPv6 frames over UDP between IPv4 sites. Peers connected over 6bed4 can switch to direct routes over UDP/IPv4 after deducing that this will be reliable.

6bed4 lets peer-to-peer applications benefit from transparant addressing in IPv6 and delegates NAPT concerns to 6bed4. It is possible to use 6bed4 as a fallback for IPv6, or as an additional route. Servers can be setup as IPv6-only servers with NAT64 for IPv4-only customers who only need client-server facilities, and add a 6bed4router to also facilitate reliable peer-to-peer protocols.

Status of This Memo

Copyright Notice

Table of Contents

1. Terminology 3
2. Introduction 3
   2.1. Address Format 3
   2.2. Protocol Description 4
   2.3. Use Cases 5
3. 6bed4 Network Components 6
   3.1. IPv6 Address Validation 6
   3.2. Router Solicitation and Advertisement 6
   3.3. Network Prefixes 7
      3.3.1. Native /64 Prefixes 8
      3.3.2. Locally Routed fc00::/7 Prefixes 8
      3.3.3. The 6bed4 fc64::/16 and TBD1::/32 Prefixes 9
4. The 6bed4peer Component 10
   4.1. 6bed4peer Forwarding 10
   4.2. 6bed4peer Filtering 12
5. The 6bed4router Component 13
   5.1. 6bed4router Filtering 14
   5.2. 6bed4router Forwarding 15
   5.3. Combining 6bed4peer and 6bed4router Functions 16
6. Peering Policies 16
7. Reliable Peering 19
   7.1. Reliable 6bed4router Uplinks 19
   7.2. Probing for Direct Peering 20
   7.3. Sending and Receiving the Seen Flag 20
   7.4. Peer NAPT State 21
   7.5. State/Timer Diagram 22
   7.6. Differentiation through Peering Policies 23
   7.7. Bindable Local Prefix 24
   7.8. Benefiting from Adaptive Flags 25
8. Global Routing of TBD1::/32 26
9. IANA Considerations 26
10. Security Considerations 27
11. Acknowledgements 28
12. Normative References 29

Author's Address 31

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].

2. Introduction

Several tunnels for IPv6 have been proposed [RFC7059]; the novelty of 6bed4 is that it allows the assumption that IPv6 is everywhere; not only can it be implemented for hosts or networks, but even inside an application. As a result, application developers can rely on transparant IPv6 addressing, even when their code is distributed over a network that may include IPv4-only users. Currently unsupported use cases such as SIP/RTP, direct file transfer and other peer-to-peer protocols can benefit from the relative simplicity of this model and the knowledge that traffic either transfers directly between peers, or reflects through a relay of the user's choosing.

To carry IPv6 anywhere, 6bed4 transports it as a UDP payload, contained in UDP/IPv4. The UDP port and IPv4 address are derived from the IPv6 address on sending, and validated to match upon arrival. This means that much of the 6bed4 infrastructure can be stateless, like a router. In addition, IPv4 attackers are still traceable when they use 6bed4 to step up to IPv6.

The 6bed4 network consists of any number of 6bed4peers running IPv6 applications and usually a 6bed4router that defines a prefix under which it reliably connects peers, and through which it may route to and from native IPv6 addresses. To optimise routing, one 6bed4peer may choose to access multiple 6bed4routers, but automatic detection of crossover between 6bed4peers under different 6bed4routers is also possible.

2.1. Address Format

The structure of a 6bed4 address may involve a specialised top half structure in the /64 prefix and/or a specialised bottom half structure following it. The format of a 6bed4 address with both specialised parts is:

```
| 32 bits |         32         |          50            |  14   |
+----------+---------------------+-------------------------+-------+
|  prefix  | 6bed4router address | direct 6bed4peer address| lanid |
+----------+---------------------+-------------------------+-------+
```

Whether the top half is a 6bed4 address depends on the prefix. The prefix TBD1::/32 globally defines a 6bed4 address, and supports

routing across the Internet. Prefixes fc64:<netid>::/32 for any <netid> are interpreted as 6bed4 addresses when they occurs on the 6bed4 network, but this interpretation cannot be generally prescribed. These /32 prefixes permit the interpretation of the top half, where bits 32..64 hold the IPv4 address of a 6bed4router that can further relay the traffic. Any IPv6 router may pass TBD1::/32 by forward the IPv6 packet over UDP/IPv4 to port TBD2 and the address in the IPv6 top half.

Any /64 prefix that is a destination address on the 6bed4 network must adhere to this format for the lower half. These are prefixes announced by 6bed4router and 6bed4peer components, extended to a /114 in a Router Advertisement and having the L and A flags set. Traffic with a 6bed4peer source address never came from a routed backend, so source addresses arriving over direct peering instead of from a subscribed-to 6bed4router can also be considered to have a 6bed4 lower half. This lower half contains the IPv4 address and UDP port, as observed by an addressed party.

2.2. Protocol Description

The role of a 6bed4router is to be the defining home for one /64 prefix and potentially connect to other IPv6 addresses. It binds a UDP socket to a static IPv4 address and the standard UDP port TBD2.

Every 6bed4peer opens a UDP socket and is identified by one or more external pairs of an IPv4 address and UDP port. The UDP socket is seen as a /64 prefix, usually obtained from a 6bed4router, extended into to as many /114 prefixes as the 6bed4 network has external address/port pairs for its UDP/IPv4 socket.

When a 6bed4peer desires to run under the /64 prefix of a given 6bed4router, it sends a Router Solicitation to its UDP/IPv4 address. The response is an initial Router Advertisement that offers a /114 prefix, consisting of the /64 of the 6bed4router externded with the UDP/IPv4 address of the 6bed4peer, as observed by the 6bed4router. This /114 must be used as the source address in future communications with that 6bed4router. A 6bed4peer sends Keepalive messages to keep the NAPT mapping towards the 6bed4router open as a reliable bidirectional routing path.

When attempting direct peering, the UDP/IPv4 destination address of a remote 6bed4peer is collected from its /114 and a direct UDP/IPv4 message is attempted. This may or may not succeed, so the initial attempt is usually a Probe that may result in a future report of a Seen flag. In reliable modes of operation, the initial traffic is sent through the 6bed4router until it learns that direct peering is

possible; more agressive but less reliable peering policies are defined as alternatives.

Traffic on the 6bed4 network is validated to hold an IPv6 source address with a lower half mentioning the source UDP/IPv4 address. Failure to match rejects the message with a corrective Router Advertisement in return. This allows a 6bed4peer to communicate directly with any 6bed4peer, and to learn of another external UDP/IPv4 address for its UDP socket in relation to other network components.

2.3. Use Cases

The 6bed4peer can be run for a network, a machine or even a single application. It can add IPv6 to any machine that supports IPv4, whether it already supports IPv6 or not.

The 6bed4router can share a /64 prefix to any number of 6bed4peer nodes. It may offer additional routes to network regions that can route back to this /64 prefix, including an offering of a default route if the shared prefix is globally routable.

Applications may benefit from the certainty that IPv6 is available, especially when they run their own 6bed4 network. They would normally want to connect to a 6bed4router at its specified IPv4 address and UDP port TBD2 and obtain a /114 prefix holding their own IPv4/UDP address. The final 14 lanid bits allow a range of addresses to divide as the 6bed4peer sees fit.

Peer-to-peer applications may prefer to evade the 6bed4router and prefer direct peering, at least when both peers are known to use 6bed4 address bottom halves. This may not always succeed, mostly dependent on NAPT properties which cannot generally be solved. The desired peering policy can be specified in each frame, and peering success or failure can be learnt from received frames, so preferences among alternative routes could be based on peering properties.

The reliable and desired-direct approaches can be mixed, because the peering policy is separately set in the traffic class for each IPv6 frame. An application might connect (perhaps with SIP) through a 6bed4router and use a direct-peering data path (perhaps with RTP) afterwards. It is possible to bind to another address for RTP than for SIP, and the addition of Keepalive messages can even support non-symmetric RTP streams; all this can help to find direct routes where they are possible, and thus rely on a minimum of fallback routing through the 6bed4router.

3. 6bed4 Network Components

This section describes common aspects that apply to 6bed4peers as well as 6bed4routers. Later sections specify aspects that these components add.

Every component on the 6bed4 network opens a UDP port over which it sends and receives IPv6 frames. The further processing of these frames depends on the component.

3.1. IPv6 Address Validation

Upon arrival of an IPv6 frame over UDP/IPv4, the IPv6 source (and destination) address is verified. This usually involves testing an IPv6 address to have a lower half containing an IPv4 address and UDP port, almost in network bit order. The lower half must follow this format:

```
 |      6      |  2   |     24      |  16  |      2      |   14  |
-+-------------+------+-------------+------+-------------+-------+
 | IPv4 [0..5] | EIU64 | IPv4 [8..31] |  UDP  | IPv4 [6..7] | lanid |
-+-------------+------+-------------+------+-------------+-------+
```

The IPv4 address is split into portions [N..M] with bits N to M, counting from 0 for the high end. The IPv4 address effectively has two bits taken out to conform to the EUI-64 address format [RFC3513]. The overlaid IPv4 address bits follow after the UDP port. The last 14 bits form the lanid, which can be freely used on the component bound to the given UDP port and IPv4, except for the value 0, which is reserved for the 6bed4router for this IPv6 address.

3.2. Router Solicitation and Advertisement

When a local 6bed4 component intends to connect to a remote 6bed4 component, it may send a Router Solicitation [Section 4.1 of [RFC4861]] to the IPv4 address and UDP port of the remote. The prefix supplied can be used after the remote sends back a Router Advertisement [Section 4.2 of [RFC4861]], the composition of which is

- Flags [Section 3 of [RFC5175]] are M=0, O=0, H=0, P=0.

- A prefix of length /114, where the /64 portion defines the 6bed4 network and the added 50 bits hold the local component's UDP port and IPv4 address as observed by the remote 6bed4 component. The trailing 14 bits are the lanid; lanid 0 is reserved for the

  router, but the other 16383 values may be used freely by the local component, perhaps through DHCPv6.

- Any number of routes, possibly including a default route. Any route MAY be ignored by the local 6bed4 component. Routes only make sense when the reachable prefixes can route traffic back to the remote 6bed4 component; a 6bed4peer SHOULD NOT offer routes because it is considered a terminal in the 6bed4 network, rather than an authoritative source of routes.

Every component on the 6bed4 network SHOULD send a Router Advertisement to correct an invalid bottom-half Section 3.1 in an IPv6 source address. This allow the sending 6bed4 component to correct its own idea of its externally observed UDP/IPv4 address, at least towards the designated recipient. The bits that would change are the bottom-half bits holding its IPv4 address and UDP port; note that the lanid is always zero in a /114 prefix.

3.3. Network Prefixes

The 6bed4 network uses /114 prefixes for its destination addresses. These addresses contain a UDP/IPv4 address in their bottom half, following a /64 prefix in the address top half. The top half is considered the identity of a 6bed4 network segment, as ususally defined by a 6bed4router and sometimes by a 6bed4peer. The top half never changes while processing a corrective Router Advertisement; it does however get assigned by the initial Router Advertisement that follows up on a Router Solicitation.

Even if a 6bed4peer obtains a /64 prefix from a 6bed4router as part of a /114 in an initial Router Advertisement, and even though it is not a router for that address, it may nonetheless use the /64 to construct a /114 towards other 6bed4peers. This bottom-half logic is a point where 6bed4 destination addresses have more semantics than general IPv6. Two 6bed4peers may use Router Solicitation and/or Router Advertisements to learn about each other's view on their addresses. This behaviour is not required for reliable exchanges, but it can help to pierce through more kinds of NAPT router; it is why the /64 is said to describe a 6bed4 network, rather than just the component that introduces it.

Typical for IPv6, there is some variation in use cases for different prefix kinds. We distinguish native prefixes, locally routed prefixes and specific 6bed4 prefixes.

3.3.1. Native /64 Prefixes

Every 6bed4 component can export a native /64 prefix, provided that it can route to it and that the native prefix is routed back to it. This facility allows sharing that prefix to connecting other components; do note that no access control exists, but the bottom half of an IPv6 address under the prefix reveals the validated UDP/IPv4 address of a source.

The top-half of a native prefix is not recognised as a 6bed4 address, as it is not possible to extract a 6bed4router IPv4 address from it. As a result, its traffic usually passes over native IPv6 routes. The benefit of a publicly routable IPv6 address is that it can be used in connections to arbitrary other IPv6 addresses that are also globally routable.

Some hosts have only one /64 available, and may want to mix 6bed4 with services. Although invalid 6bed4 bottom halves (for instance, UDP port 0) could be allocated for other uses than 6bed4, this is NOT RECOMMENDED because it would break connectivity for other 6bed4 components when the prefix is passed through the 6bed4 network via Router Advertisements. Instead, the RECOMMENDED procedure would be to setup IPv6 addresses as available to a locally run 6bed4peer on a fixated public IPv4 address and UDP port; the 6bed4 component software may optimise handling for these purposes.

3.3.2. Locally Routed fc00::/7 Prefixes

Some IPv6 addresses have been allocated for local routing [RFC4193]. The fc00::/7 prefix is divided into administratively assigned fc00::/8 prefixes and randomly completed fd00::/8 prefixes. With the exception of fc64::/16 discussed below, these addresses are locally routed, also on a 6bed4 network.

Local routes cannot be used to communicate with native IPv6 address, unless they happen to be aware of how to return the traffic. Such native routes in the direct backend of a 6bed4router can be explicitly mentioned in its Router Advertisement, even for a fc00::/7 prefix.

Locally routed addresses can only be communicated with the 6bed4 component that defines them. Because of this, they MUST NOT be further transmitted through Router Advertisement; connections are 1:1 only. Whether this is a 6bed4peer to a 6bed4router (reliable) or a 6bed4peer to another 6bed4peer (not reliable) is a matter of application or network configuration.

It is quite possible for a 6bed4peer to setup a random /64 prefix based on fd00::/8, and peers might even form networks between such addresses, possibly based on a distributed hash table. In such networks, redundancy can be helpful to overcome unreliable direct peering connections. As explained below, 6bed4 can support the detection of reliability.

3.3.3. The 6bed4 fc64::/16 and TBD1::/32 Prefixes

The prefixes fc64::/16 and TBD1::/32 mark what are called 6bed4 prefixes. The fc64::/16 prefix receives an additional network identifier <netid> to form fc64:<netid>::/32. These /32 prefixes are used in a top half, whose format is completed with the IPv4 address of a 6bed4router, in network byte order:

```
|               32                |               32                |
+---------------------------------+---------------------------------+-
|      fc64:<netid> or TBD1       |   IPv4 address of 6bed4router   |
+---------------------------------+---------------------------------+-
```

The 6bed4router is responsible of knowing its public IPv4 address. The fixed UDP port on which the 6bed4router provides its service MUST be TBD2. As a result, a network component that can interpret the prefix as a 6bed4 prefix can route the traffic over the 6bed4 network. For the TBD1::/32 prefix, this can be any party on the Internet; for fc64::/16 it can only be assumed when the address is found on the 6bed4 network. The added value of TBD1::/32 over fc64::/16 therefore is that it expands the IPv6-everywhere facilitation of 6bed4 from an overlay network to a globally routed network.

The 6bed4router is vital in the reliability of the 6bed4 network:

- o The 6bed4router is always reachable at its IPv4 address and the fixed UDP port TBD2;
- o The 6bed4router can always reach every 6bed4peer that subscribes to its serviced prefix.

This means that a conservative route can always be made through a 6bed4router. This is also possible when the prefixes differ, either in the IPv4 address or also in the /32 part. Within the constraints of source address validation, it is possible to route traffic through the 6bed4router covering the source prefix and the 6bed4router covering the destination prefix; or it is possible to route traffic

through the 6bed4router of a destination node, after a source 6bed4peer (also) connects to the destination's 6bed4router.

Another value of the 6bed4 prefix is that they represent end points on the 6bed4 network. This means that the bottom half can also be interpreted and used for direct 6bed4 traffic. This usually works best when the source address is then also chosen to be a 6bed4 address, which can be achieved under normal address binding rules when a low-priority interface offers routes for fc64::/16 or TBD1::/32 with a prefix under either.

4. The 6bed4peer Component

The 6bed4peer opens a UDP socket over which it sends and receives IPv6 frames. The UDP remote end can be any number of 6bed4peers and/or 6bed4routers. A basic configuration would communicate with a single 6bed4router which may be its default route to native IPv6 addresses, and as many 6bed4peers as it can connect to directly.

To be able to route under a 6bed4router's prefix, the 6bed4peer sends a Router Solicitation and awaits an initial Router Advertisement Section 3.2 to learn about its /114 prefix, which includes the /64 prefix provisioned by the 6bed4router. The 6bed4peer MAY configure any additional routes provided in a Router Advertisement from a 6bed4router. Through the Router Advertisement, the 6bed4peer learns about the external address of its UDP socket, as observed by the 6bed4router. Over this route, the 6bed4peer will send regular Keepalive messages to keep the NAPT traversal open and guarantee a reliable incoming route through the 6bed4router. A generally advised minimum frequency for Keepalive messages is once in 30 seconds.

4.1. 6bed4peer Forwarding

To submit an IPv6 frame on the 6bed4 network, a 6bed4peer first determines whether it can forward to a 6bed4router or directly to a 6bed4peer:

- When source and destination share the same /64 prefix, consider direct routing as well as the 6bed4router.

o When the destination is a 6bed4 prefix and the source and destination have different /64 prefixes, consider direct routing as well as the 6bed4router.

o When the destination has another prefix (that was offered and accepted as a route from a 6bed4router), consider going through the 6bed4router.

When a direct route is considered, the default peering policy Section 6 only uses direct peering when it is known to be reliable Section 7. Other peering policies provide variations on a frame-by-frame basis, to allow for maximum flexibility. When direct routing is selected, any considerations of going through the 6bed4router are dropped.

When considering the 6bed4router, the /64 prefix of the source address determines which one to use. When the source and destination have 6bed4 addresses with different /64, the traffic bounces through both 6bed4routers, which is useful to validate the traffic for not forging IPv6 addresses. To avoid this "trapeziums-shaped" routing, it is also possible for a 6bed4peer to bind an address under the destination address's 6bed4router by sending a Router Solicitation and acquiring a /114 prefix to work from. The result would be only one 6bed4router bouncing the traffic instead of two. Whether or not this is done, direct peering may be discovered as reliable alternative and either shape bypassed completely.

The IPv6 frame can now be sent over the 6bed4 network, from the UDP socket held by the 6bed4peer. The remote UDP port and IPv4 address are learnt from the bottom half of the destination IPv6 address. Since a 6bed4peer is not supposed to route traffic, any source address is supposed to be locally bound, therefore be a 6bed4 address, and so its bottom half is supposed to contain the external view on its UDP port and IPv4 address.

Although it makes less sense for UDP than for TCP, NAPT middleware may have imposed an Endpoint-Dependent Mapping [RFC4787], which means that the external UDP port and IPv4 address observed by the 6bed4router form a reasonable initial guess when communicating directly with other 6bed4peers, but there may be a need to correct these aspects when communicating with such 6bed4peers. To this end, a remote peer might send a corrective Router Advertisement and the IPv6 frame would be lost. This should not happen when reliability rules are obeyed, but it may happen for frames that select another peering policy than the default.

Note how this opens degrees of freedom to the an application acting as/via a 6bed4peer. For general TCP or SCTP, a first SYN or INIT frame might be sent under a peering policy that enforces direct peering, and fall back to reliable routing when resent. And a application could send SIP messages over reliable patterns, but work towards direct peering for RTP; a SIP/SDP offer can welcome RTP traffic on a 6bed4 address, possibly using a 6bed4router in a SIP/SDP offer that was just received; many SIP networks are closed, and could opt to be IPv6-only networks, with 6bed4 as a reliable fallback option.

4.2. 6bed4peer Filtering

Anyone might send packets to the UDP socket of a 6bed4peer, but not all traffic should pass. Specifically, filtering on the IPv4/UDP source address in relation to the IPv6 source address is necessary to stop peers from claiming arbitrary IPv6 addresses.

Traffic without a full IPv6 header is exceptional, and considered to be a Probe; an attempt to reach our UDP socket through direct peering. This is not routed any further, but it counts as successful input under direct peering. Usually sent before or after an IPv6 frame via the 6bed4router, it permits flagging that this direct input succeeded on return traffic to that same 6bed4peer.

There are two reasons why an IPv6 header's source address might be acceptable; it could be from a 6bed4router to which the receiving 6bed4peer maintains an uplink, or it might be a direct connection from another 6bed4peer under the same 6bed4router. On top of this, a third reason to accept an IPv6 header is when its source and destination address together indicate that a bypass of a trapezium route is made.

The 6bed4peer should know the 6bed4routers to which it maintains an uplink; it needs this information for sending Keepalives that maintain an open NAPT mapping. Each 6bed4router can be uniquely identified by matching their IPv4 address and UDP port against the UDP/IPv4 source address of an incoming frame.

A frame sent directly from another 6bed4peer working under the same 6bed4router will use a /64 prefix that we got from that 6bed4router. As a result, we can rely on the bottom half address to contain a IPv4 address and UDP port, which MUST then match the source of the incoming frame. When the /64 prefix matches but the bottom half does not, a corrective Router Advertisement SHOULD be sent (possibly with a limited frequency). TODO:REQUIRE_SRC/64_IS_DST/64?

The bypass for a trapezium route is more complicated. In this case, the frame came from a 6bed4peer acting under another 6bed4router. The frame was first passed to the source's 6bed4router and then the destination's. When the latter delivers the frame at the 6bed4destination, the source and destination IPv6 address are used. First, both addresses MUST be 6bed4 addresses, so have either the fc64::/16 or TBD1::/32 prefix, in any combination. Second, the 6bed4router addresses in the top half are considered; the source 6bed4router address is assumed to have been validated by our 6bed4router; the destination 6bed4router address MUST be found as one to which we maintain an uplink. Third, the source IPv4 address and UDP port MUST be set in the bottom half of the source IPv6 address.

Fourth, our IPv4 address and UDP port according to our 6bed4router (which is mentioned in the top half of the IPv6 destination address) MUST match the bottom half of the IPv6 destination address. Failure on any of these four conditions leads to discarding of the frame as a suspect frame.

5. The 6bed4router Component

The 6bed4router is a stateless component. It can be reached reliably on a static IPv4 address on the fixed UDP port TBD2, so it can be reached by all 6bed4 components. If a NAPT mapping applies, it is a static mapping.

Every 6bed4router defines its own /64 prefix and when this is a 6bed4 address its static IPv4 address is contained in the IPv6 address top half that forms this prefix. In addition to the offered prefix, a 6bed4router may also exchange IPv6 frames with locally routable prefixes and/or with globally routable prefixes anywhere on the

Internet.

Among the routes offered may be the default route ::/0. This is just one of many possible routes.

Another seemingly special route is fc64::/16, which captures more than just a specific fc64:<netid>::/32 that the 6bed4router may use as its prefix. A route fc64::/16 states that other <netid> values can be routed to the 6bed4router's connected network. Only the one announced in the prefix is handled directly by the 6bed4router. Again, there is no need to treat such cases in any special way. TODO: BUT WE DO TREAT IT ESPECIALLY; IF NOT OUR IPV4 IS USED, WE RELAY SUCH A PREFIX. SAME FOR TBD1::/32 BY THE WAY.

Traffic that arrives over the 6bed4 network is first filtered to be properly formed. After this, one possible forwarding option is to bounce the frame back into the 6bed4 network, but to another IPv4 address and UDP port. This is done to relay traffic between two 6bed4peers that share the 6bed4router's /64 prefix but that are not (yet) able or willing to connect directly. It is also done to relay traffic between two 6bed4routers if their 6bed4peers use different /64 prefixes and are not (yet) able or willing to connect directly; this situation is called trapezium routing and is effectively a bypass for routing over IPv6, mostly to permit fc64::/16 prefixes to work across 6bed4routers' prefixes.

5.1. 6bed4router Filtering

The 6bed4router is stateless, but it is careful about filtering traffic before agreeing to route it. Different filtering rules are applied on the IPv6 side of the 6bed4router than on the side of the 6bed4 network.

Traffic arriving at the IPv6 side is called return traffic. Traffic arriving over the 6bed4 network can be routed for either local, backend or first and second stage of trapezium routing.

Return traffic MUST be rejected unless:

- o The destination IPv6 address matches the 6bed4router's defined /64 prefix;
- o The bottom half of the destination IPv6 address does not mention IPv4 address 0.0.0.0;
- o The bottom half of the destination IPv6 address does not mention UDP port 0.

Local routing applies when the IPv6 source and destination addresses both use the /64 prefix defined by the 6bed4router. Local routing MUST be rejected unless:

- o The bottom half of the IPv6 source address mentions the IPv4 address and UDP port over which the frame arrived from the 6bed4 network;
- o The bottom half of the IPv6 destination address does not mention IPv4 address 0.0.0.0;
- o The bottom half of the IPv6 destination address does not mention UDP port 0.

Backend routing applies when the IPv6 source address uses the /64 prefix defined by the 6bed4router, while the IPv6 destination address matches a route announced in Router Advertisements. TODO:CONSIDER_TRAPEZIUM_ROUTING_FOR_6BED4ADDRS_WITH_DIFFERENT_TOP_IPV4ADDR. Backend routing MUST be rejected unless:

- o The bottom half of the IPv6 source address mentions the IPv4 address and UDP port over which the frame arrived from the 6bed4 network.

The first stage of trapezium routing applies when the IPv6 source address uses the /64 prefix defined by the 6bed4router, but the

desination IPv6 address neither suggests local or backend routing. The first stage of trapezium routing MUST be rejected unless:

- o The source and destination IPv6 addresses are both 6bed4 addresses, though not necessarily the same;

- o The IPv6 source address matches the /64 prefix of the 6bed4router;

- o The bottom half of the IPv6 source address mentions the IPv4 address and UDP port over which the frame arrived from the 6bed4 network;

- o The bottom half of the IPv6 destination address does not mention IPv4 address 0.0.0.0;

- o The bottom half of the IPv6 destination address does not mention UDP port 0.

The second stage of trapezium routing applies when the IPv6 destination address uses the /64 prefix defined by the 6bed4router, but the source IPv6 address neither suggests local or backend routing. The second stage of trapezium routing MUST be rejected unless:

- o The source and destination IPv6 addresses are both 6bed4 addresses, though not necessarily the same;

- o The IPv4 address over which the frame arrived from the 6bed4 network matches the 6bed4router address in the top half of the IPv6 source address;

- o The UDP port over which the frame arrived from the 6bed4 network is the standard port TBD2;

- o The bottom half of the IPv6 destination address does not mention IPv4 address 0.0.0.0;

- o The bottom half of the IPv6 destination address does not mention UDP port 0.

5.2. 6bed4router Forwarding

The 6bed4router terminates certain IPv6 destination addresses. These addresses match the /64 prefix, and the bottom half defines the IPv4 address at which the 6bed4router can be reached, along with its standard UDP port TBD2. Furthermore, any bottom half that specifies lanid value 0 is considered to terminate at the 6bed4router. Such addresses may have some special treatment for ICMPv6 traffic, but most other traffic would be relayed to a local interface binding to that address. In absense of such a binding, an ICMPv6 error may be sent back.

Return traffic is relayed over the 6bed4 network to the IPv4 address and UDP port found in the bottom half of the IPv6 destination address.

Local traffic is relayed back over the 6bed4 network to the IPv4 address and UDP port found in the bottom half of the IPv6 destination address.

Backend traffic is relayed into the backend, normally using the routing table under which the 6bed4router operates.

The first stage of trapezium routing is relayed over the 6bed4 network to the 6bed4router address in the top half of the destination IPv6 address and the standard UDP port TBD2.

The second stage of trapezium routing relays over the 6bed4 network to the IPv4 address and UDP port found in the bottom half of the IPv6 destination address.

5.3. Combining 6bed4peer and 6bed4router Functions

It is possible for a single component to act both as a 6bed4peer and a 6bed4router. The individual requirements for each apply, specifically resolving potential conflicts with:

- The component MUST be externally reachable on a static IPv4 address at the standard UDP port TBD2;
- When sending a Router Advertisement, it MUST NOT provide additional routing information.

6. Peering Policies

The IPv6 frame that travels over the 6bed4 network, so between 6bed4 components, can express its preferred peering policy. This is expressed through the Traffic Class, which is spread over two bytes in the IPv6 header. The value of the Traffic Class is generally considered [RFC2474][RFC3168] to follow this structure:

```
|       6       |  2  |
+---------------+-----+
|      DS       | ECN |
+---------------+-----+
```

Part of the DS field is interpreted [RFC2474] as a Class Selector CS:

```
|    3    |  3  |  2  |
+---------+-----+-----+
|   CS    | 000 | ECN |
+---------+-----+-----+
```

On the 6bed4 network, the interpretation of CS is further split into a two-bit peering policy PP and a Seen flag S:

```
|  2 | 1 |  3  |  2  |
+----+---+-----+-----+
| PP | S | 000 | ECN |
+----+---+-----+-----+
```

The PP and S values may be retained when a frame exits a 6bed4peer and arrive at an application. The Seen flag expresses that direct peering from this side to the source address recently succeeded, even if just as a Probe. This means that direct peering to the IPv6 source address is considered reliable for 27 seconds from the arrival of the frame with the Seen flag. The 6bed4peer normally takes note of this flag, and modifies its peering behaviour accordingly.

Before entry into the 6bed4 network, so in the application that constructs the IPv6 frame to be relayed through a 6bed4peer, the peering policy PP indicates the desired peering policy, but the Seen flag has no meaning yet; its place is taken by Adaptive flag A:

```
|  2 | 1 |  3  |  2  |
```

```
+----+---+-----+-----+
| PP | A | 000 | ECN |
+----+---+-----+-----+
```

The Adaptive flag expresses that the IPv4 address and UDP port in the bottom half of the source IPv6 address may be modified at will. When local NAPT performs Endpoint-Dependent Mapping it would map the same

UDP socket to different external address/port combinations for different remote peers. Normally, this makes the reliable traffic fall back on the 6bed4router, because there is no basis of trust for the remote that two seemingly different peers map to the same UDP socket and hence to the same 6bed4peer. Through Adaptive source addresses, a prefix supplied through a corrective Router Advertisement in the past can be used to construct a suitable source address. Other than the Adaptive flag, the 6bed4peer needs no instructions to do this. The application may learn the new address from replies to a frame sent with the Adaptive flag. It is then free to adopt the address and continue without an Adaptive flag, or to continue and even allow connected updates to the address when NAPT changes state. If and when this can work is up to the application logic.

The four peering policy (PP) values defined for 6bed4 are:

Proper Peering has bit value 00 or decimal value 0 and is the default. This policy routes through the source's 6bed4router when direct peering has not been detected to be reliable, but as soon as it is considered reliable it switches to direct peering.

Prohibited Peering has bit value 01 or decimal value 1. This policy prohibits direct peering and will always route through the source's 6bed4router.

Presumptious Peering has bit value 10 or decimal value 2. At the expense of reliability, this policy tries direct peering for a few seconds. If reliability is not achieved within these seconds, it falls back to the same behaviour as Proper Peering, which can also route through the 6bed4router. Until that fallback however, traffic may be lost.

Persistent Peering has bit value 11 or decimal value 3. At the expense of reliability, this policy tries direct peering for a few seconds. If reliability is not achieved within these seconds, it persists in this behaviour but will report errors, either with return values or through occasional ICMPv6 errors. This policy is the most likely to drop traffic.

These values can be chosen on a frame-by-frame basis without damage to the logic that learns about the reliability of direct peering. Applications being aware of the meaning in a protocol of each frame, may choose to use less reliable delivery modes for application-specific purposes.

7. Reliable Peering

Given the nature of NAPT, there can be no completely reliable peering system without a fallback to a services that bounces traffic. This is why a 6bed4 network needs the fallback option of a 6bed4router to offer reliable routing. When NAPT is enhanced with port forwarding, this situation can be changed, and some applications may implement so many alternative routing options that full reliability might be waived. This is why special situations might be created to work reliably without a 6bed4router. However, when 6bed4 is built into an application that should "just work", it does need the fallback 6bed4router to be reliable.

The 6bed4 network can reliably switch to direct peering after a successful peering handshake. This is a deductive approach to NAPT traversal, and is achieved simply by trying direct peering and observing if it arrives. The lesson taken from prior inductive approaches [RFC4380] founded on classification of NAPT [RFC3489] was that such an approach can easily misclassify NAPT behaviour, ensuing in brittle IPv6 connectivity.

The decuction in 6bed4 is mostly through properties of UDP. Specifically, the absense of a protocol identifier disables interpretation of the protocol behaviour by NAPT, unless bold assumptions are made on the basis of a port number. To be workable, a NAPT therefore must keep an outward-sending UDP port open for response traffic. For some protocols, the response may come from other angles, which suggests that Endpoint-Dependent Mapping is not suitable for UDP, but this cannot be assumed in general. A certain amount of acceptable silence until the given connection closes can however be assumed, and it is commonly suggested that this should be at least 30 seconds. The only assumption made by 6bed4 is that these 30 seconds are a reasonable default, but of course this MUST be a setting that can be overridden by operators. NAPT software MUST NOT make any assumptions of passing 6bed4 on the basis of the standard port TBD2 if it is to comply with this specification.

7.1. Reliable 6bed4router Uplinks

Given that NAPT can only handle 6bed4peer traffic as generic UDP traffic, any outbound UDP traffic necessarily keeps a hole open for reverse traffic over a reasonable minimum time. When more traffic is sent before this period has passed, there is no basis for NAPT to assume that the connection can be severed, causing a reset of the timeout for the hole.

To allow reliable bidirectional traffic between a 6bed4peer and its 6bed4router, it therefore initiates with a Router Solicitation, and

then continues to send Keepalive messages with no smaller separating delay than the delay for UDP holes.

Keepalive messages need not make it to the 6bed4router to be effective. Their only need is to make it beyond the last NAPT or firewall. This can both help to offload a 6bed4router and to avoid that it detects the 6bed4peer still being online; since the 6bed4router is stateless, it has no use for this information.

The content of a Keepalive message can be the minimum that counts as UDP traffic, according to NAPT and firewalls; this is an empty UDP frame.

A 6bed4peer can keep up any number of 6bed4router uplinks, and would send Keepalive messages to each of them. This being a resource, the application may have to indicate explicitly what uplinks are considered to be useful (or a trivial setup such as a single uplink might be applied).

7.2. Probing for Direct Peering

When choosing how to forward an IPv6 frame, a 6bed4peer may consider direct routing, but refrain from it on account of reliable considerations. These are good moments to start working towards direct routing. Other moments are when a 6bed4peer forces direct peering through the Presumptious and Persistent Peering Policies.

While working towards direct routing, a few attempts to connect directly are made, by sending a Probe message if no direct traffic is sent at the same time. Probes are the smallest possible UDP frame, so an empty UDP message. They differ from Keepalives because their reach is not constrained but the message is sent with the intention of arriving at the remote 6bed4peer. The destination IPv4 address and UDP port are taken from the bottom half of the destination IPv6 address, which is only permitted for addresses known to be bound by the 6bed4 network.

When a Probe arrives on a 6bed4peer, it cannot be considered an IPv6 frame. It is however detected as an incoming UDP frame from a given IPv4 address and UDP port. This identifies a remote peer that appears to be able to make its way to the UDP socket, all the way through the NAPT and firewalls at the destination end.

7.3. Sending and Receiving the Seen Flag

When a Probe arrives at a 6bed4peer, it MUST look if the sending IPv4 address and UDP port have recently received any traffic. If so, the Probe will be processed into a timer/state product.

Starting with the time of arrival, a hole can be reliably assumed to be open in local NAPT, and that it remains open for 30 seconds (or a locally overridden timeout) after the last direct send to that remote, even if it was just a Probe or Keepalive.

The time for the open hole is subdivided as follows:

- 1 second accounts for a message delay in both directions;

- 27 seconds account for the time that the remote peer may send directly;

o the remaining time allows informing the other side that a Probe arrived.

when a NAPT hole is assumed to be open for 30 seconds, then this remaining time amounts to 2 seconds. This time period, added to the time of the last message sent to that remote, indicates a period in which the Seen flag can be sent to the remote. The remaining time may be negative, in which case the Sent flag is never sent to the remote. When the local NAPT holes are set to be open for longer periods, then the Seen flag is sent correspondingly longer.

The Seen flag can be set in any Traffic Class that adheres to the format for the 6bed4 network. It does not matter if it is relayed through one or two 6bed4routers, because it informs about the direct connectivity between terminating 6bed4peers, whose IPv4 address and UDP port must occur in the bottom half of addresses if direct peering is to be considered at all.

It is an event when the Seen flag arrives from a 6bed4peer, even from a 6bed4router, when it occurs in a Traffic Class that adheres to the format for the 6bed4 network. In case of this event, the destination 6bed4peer may conclude that it can use direct peering to the remote 6bed4peer for 27 seconds after arrival of the Seen flag. In active direct peering connections, the Seen flag is likely to be present on most frames, causing the regular reset of the 27-second timer.

7.4. Peer NAPT State

Every 6bed4peer keeps some state for its remote peers. In contrast, the 6bed4router is a simple stateless process. The state kept per remote peer is known as Peer NAPT State, and is indexed by the IPv4 address and UDP port for the remote peer.

The information kept for each remote peer consists of:

o A current state and related timeout

o A time for the last message sent

o A time for the last message sent directly

o A time until which the Seen flag is reported o A /64 or /114 prefix for local address binding (once known)

There is no explicit time until which direct peering is possible; this is instead derived from the current state.

This information differs from 6bed4router state, which minimally requires just a /114 prefix and a timer until the next Keepalive needs to be sent. If so desired, the Keepalive timer can be reset when a message is sent to the 6bed4router without risking to reduce reliability.

7.5. State/Timer Diagram

The Peer NAPT State for a 6bed4peer follows the following state diagram with default progression timeouts to administer sender state:

```
                   First.Send
                      |
                      V
                       INIT --1s--> INIT --1s-->.<------+
                       |            |           |       |
                       V            V           |       |
   Seen.Flag         Probe        Probe         |      25s
       |                                        |       |
       V                                        V       |
        PEER---25s---> POLL --1s--> POLL --1s--> FAIL --+
                       |            |            |
                       V            V            V
                     Probe        Probe        Probe
```

This section describes the flow of this state/timer diagram for the Proper Peering, which is the default peering policy. The next section describes the modifications for the other peering policies.

The transitions marked 1s and 25s represent a default transition to occur after 1 and 25 seconds in the state, respectively. Timeouts for NAPT holes below 28 seconds MAY be rejected by a 6bed4peer; if not, it MUST split the 25 second delays into values less that the NAPT hole timeout and send Keepalives at the extra points. The total time MUST however be kept at 25 seconds.

First.Send marks the entry point to the diagram, used when initiating new Peer NAPT State for a remote 6bed4peer that had no such state managed yet. When entering FAIL state after a reasonable period with no traffic sent to the remote peer, the Peer NAPT State may be removed from administration. What counts as a "reasonable period" may be locally selected, but would normally be in the range of tens of seconds.

Seen.Flag marks the point of reset of the state when a Seen flag is set for traffic from this remote. When connections are actively used, this flag would occur on most frames, causing repeated resets of this state diagram.

Probe marks the side-effect of sending a Probe upon entry of a state. The intention of the Probe is to open a hole in outgoing NAPT and, at the same time, to reach the remote peer directly and trigger the return of a Seen flag.

The states in the diagram should be read as follows:

INIT marks the intention to work towards reliable direct peering. A total of 3 Probes is sent to allow reasonable opportunity for the remote peer to detect it and send a Seen flag.

PEER marks reliably direct peering for a duration of 27 seconds (continued into POLL states). Having established in hole in local NAPT, no need for further Probes exists.

POLL still marks reliable direct peering, but also starts to send Probes because the hole in NAPT is assumed to be timing out soon. The Probes are hoped to trigger new Seen flags, and are especially useful during one-directional transmissions. However, when no traffic is passed at all there is nothing to carry the Seen flag, and the state diagram continues.

FAIL marks durable failure to cause a Seen flag to be returned from the remote peer. This does not imply that the hole in NAPT towards that peer has been closed however; with 25 seconds between Probes, the hole would be kept open. TODO:SETTING. Because Probes have no IPv6 header, there is no vessel for Seen flags, so connections are setup to close after a reasonable time of inactivity.

7.6. Differentiation through Peering Policies

When a Traffic Class passes into and over the 6bed4 network, and its format is that defined for the 6bed4 network, then it may specify a peering policy. In other cases, the default peering policy applies, which is Proper Peering.

Peering Policies define how the choice between direct peering and the source's 6bed4router is made:

Proper Peering defines the default behaviour as described in the previous paragram.

Prohibited Peering does not interact with Peer NAPT State or its State/Timer diagram. Instead, traffic is always forwarded to the source's 6bed4router.

Presumptious Peering forces direct peering during the INIT states, but has the default behaviour in all other states. When the frame causes creation of Peer NAT State, the First.Send entry point is used, but the first Probe MAY be dropped because the frame causing it takes its place.

Persistent Peering forces direct peering in all states, but triggers errors instead of sending while residing in the FAIL state.

There is only one Peer NAPT State for a given remote peer; this same diagram applies to all the peering policies, and may be modified by it. This allows traffic from a variety of peering policies and a variety of connections to share one knowledge base about direct peering.

7.7. Bindable Local Prefix

The Peer NAPT State holds a prefix, which is either the unspecified address 0::0, a /64 prefix followed by 64 zeroed bits, or a /114 prefix.

The initial state depends on the purpose; when a 6bed4peer is contacted from an already-used source IPv6 prefix, then the full /114 may be copied to it. When initiating the contact through a Router Solicitation to the remote 6bed4peer, it is set to a /64 prefix to be shared, or to 0::0 if the prefix of the remote 6bed4peer is to be assumed.

Upon arrival of an initial Router Advertisement from a remote

6bed4peer, the /64 in the prefix is only updated when it was 0::0; the following IPv4 address and UDP port in the bottom half are always copied. The last 14 bits with the lanid are left zero.

Until the /114 prefix is complete, it is not possible to bind a local address for traffic to the remote 6bed4peer, let alone use it as a

source address. Only after the /114 prefix is known is it possible to select lanid values to complete the IPv6 address.

After binding has occurred, a corrective Router Advertisement may overwrite the IPv4 address and UDP port in the bottom half of this binding prefix. This marks a change, and any further attempts to send to this remote peer from a previously bound source IPv6 address with different IPv4 address and UDP port in its bottom half are rejected by the 6bed4peer. TODO:CODE

This undesirable situation blocks further direct peering and requires a fallback to the 6bed4router, whose connection is kept open through Keepalives. The same approach can only work between 6bed4peers when both send Keepalives to each other, but this may be wasteful in many scenarios, especially when normal traffic flows regularly. Also note that what worked for setting up direct peering once is likely to work again later.

7.8. Benefiting from Adaptive Flags

The 6bed4 network is concerned with routing and peering policy; this is used by applications, whose logic may be so courteous that it does not matter if a locally bound address changes. In terms of the Socket API this would be the case when neither bind() nor getsockname() are ever called.

Such ambivalence to the locally bound address can be a benefit when NAPT mappings alter on a given connection. It may help the number of direct peering successes to signal that such an address change does not matter to the application logic. To this end, the Adaptive flag can be set in frames submitted through a 6bed4peer, at the same time as the desired Peering Policy.

When this flag is set, the behaviour in case of a mismatched prefix for the remote peer is not to reject the traffic, but instead to substitute the IPv4 address and UDP port in the source IPv6 address of a frame; the /64 prefix should match as always, and the lanid is not altered.

The address change is not reversed in the destination IPv6 address of reply traffic. This means that the application would receive the altered address. If this could otherwise lead to confusion, the application might choose to match reply traffic on the basis of a unique lanid or a random Flow Label.

It is up to the application if this is possible, and in which frames sent. The safe default is to not allow Adaptive source IPv6

addresses, but as long as it is usually beneficial to set the Adaptive flag whenever it is of no concern to application logic.

8. Global Routing of TBD1::/32

TODO: SOLVE A FILTER ON AUTHORITATIVE GATEWAY ROUTERS

HINTS FILE OPTION: Distribute the IPv4 addresses of gateway routers as part of the 6bed4router software, as is done for DNS root name servers.

DNS OPTION: Define 32.gateway.6bed4.net to hold /32 gateways, and perhaps <IPv4/16>.48.gateway.6bed4.net to hold /48 gateways. These gateways are trusted by 6bed4routers. The only thing these gateways do is to relay native IPv6 prefix TBD1::/32 from/to the 6bed4 network. The IPv4/UDP is always that of the 6bed4router and the standard UDP port TBD2.

6TO4 OPTION??? Compare to https://labs.ripe.net/Members/emileaben/6to4-why-is-it-so-bad -- the main problem of 6to4 is firewalls not being setup for it, especially -- https://labs.ripe.net/Members/emileaben/6to4-how-bad-is-it-really -- if it is pushed through without asking users. Note that a 6bed4peer may also run on a LAN, but only announces an address prefix when Router Solicitation responds. The 6bed4router should however route properly... which is more easily established when it integrates into a router device where it has access to external interfaces and/or firewall rules. Given that, even using 2002::/16 instead of

TBD1::/32 and routing backend traffic over proto-41 might be an option? But it has fallen in disgrace, even if only the anycast/unicast mixup of router setups appears to have been a cause and anycast was deprecated for 6to4. Still, there is the issue of firewall traversal (that we solve by demanding a public IPv4 address and fixed UDP port for 6bed4routers).

9. IANA Considerations

This specification reserves a 32-bit address prefix TBD1::/32 in the IPv6 address space assigned to the IANA IPv6 Special-Purpose Address Registry. The prefix will be exclusively used for 6bed4 addresses, and further assigned as defined in Section 3.3.3. Addresses under this prefix can be used as source and destination addresses, as they are globally routable. The termination date for the allocation falls ten years after the publication date of this specification in the Request For Comments series; future standardisation work could modify the end date if this is considered useful.

This specification also reserves Port TBD2 from the pools of UDP ports, TCP ports and SCTP ports, subject to possible updates in follow-up specifications.

The port TBD2 will be the port on which 6bed4routers provide their services. These servers form a public resource, and remote peers have no mechanism other than the standardised port to know how to contact an arbitrary 6bed4 Server of which only the Fallback IPv4 Address was found in an IPv6 destination address falling under the 6bed4 prefix TBD1::/32.

[[CREF1: Requests to IANA / to be removed after processing: The requested assignment for TBD1 is 2001:64, so TBD1::/32 allocates 2001:64::/32 from 2001::/23 which is suggested (with the 6BONE example) in RFC 2928 and https://www.iana.org/assignments/ipv6-unicast-address-assignments/ipv6-unicast-address-assignments.xhtml The requested assignment for TBD2 is 25790, which is what we currently use in our code and experimental deployments. As per RFC 6335, this will probably be assigned under IETF Review or IESG Approval _or_ Expert Review, all defined in RFC 5226. Note that IETF Review is banned for independent submissions under https://www.rfc-editor.org/about/independent/ and IESG Approval is not a very common procedure. For Expert Review, it is necessary to document clearly that a Dynamic Port is required; this is the case because clients have no means to infer the port at which to contact a server or a peer. Especially for peers, whose addresses are inferred from their IPv6 address within the 6bed4 address space, there is no derivation mechanism for these ports; and it is the ability to communicate directly with peers that sets this mechanism apart as extremely scalable and apt for VoIP applications. --Rick]]

10. 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 Fallback 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 Fallback 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.

11. Acknowledgements

This work has evolved from a simple, and perhaps simplistic protocol to its current form. I owe gratitude to many who contributed.

Special thanks go to SURFnet, for generally supporting the idea of 6bed4 and providing long-term support for the first public 6bed4router, as well as supporting the expansion of the initial tunnel investigation work into a generally useful publication [RFC7059].

Thanks for financial support in developing phases of the specification and coding of 6bed4 are due towards NLnet Foundation, as well as SURFnet and SIDNfonds.

I owe gratitude to the NLUUG, ISOC NL and RIPE communities for discussing prior tunnel designs and pointing out scaling and routing problems; although the protocol has become more complex it also has become more reliable and, I think, more generally acceptable.

Many thanks to Henri Manson for supporting me over a long time in experimental development and many, many tests.

Finally, the work in this tunnel design called for a creative mind set in which technical concepts were highly fluid and changing. Although it would be difficult to point out a direct link or assign an economic figure, the ability to think in such a mode has clearly been influenced by independent, thought-provoking, boundary-breaking

expression by the many artists whose work I have been able to enjoy. I cherish art in all its forms for its incredible power to help us be imaginative and innovative in the pragmatic field of technology.

12. Normative References

[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.

[IEEE-EUI64]
IEEE, "Guidelines for 64-bit Global Identifier (EUI-64TM)", December 2013.

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

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

[RFC2372] Evans, K., Klein, J., and J. Lyon, "Transaction Internet Protocol - Requirements and Supplemental Information", RFC 2372, July 1998.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <https://www.rfc-editor.org/info/rfc2474>.

[RFC3095] Bormann, C., Burmeister, C., Degermark, M., Fukushima, H., Hannu, H., Jonsson, L-E., Hakenberg, R., Koren, T., Le, K., Liu, Z., Martensson, A., Miyazaki, A., Svanbro, K., Wiebke, T., Yoshimura, T., and H. Zheng, "RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed", RFC 3095, July 2001.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <https://www.rfc-editor.org/info/rfc3168>.

[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.

[RFC3513] Hinden, R. and S. Deering, "Internet Protocol Version 6 (IPv6) Addressing Architecture", RFC 3513, DOI 10.17487/RFC3513, April 2003, <https://www.rfc-editor.org/info/rfc3513>.

[RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005, <https://www.rfc-editor.org/info/rfc4193>.

[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.

[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.

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

[RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, September 2007.

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

[RFC5175] Haberman, B., Ed. and R. Hinden, "IPv6 Router Advertisement Flags Option", RFC 5175, DOI 10.17487/RFC5175, March 2008, <https://www.rfc-editor.org/info/rfc5175>.

[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.

[RFC6455] Fette, I. and A. Melnikov, "The WebSocket Protocol", RFC 6455, December 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.

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

[RFC7059] Steffann, S., van Beijnum, I., and R. van Rein, "A Comparison of IPv6-over-IPv4 Tunnel Mechanisms", RFC 7059, November 2013.

Author's Address

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

Email: rick@openfortress.nl