INTERNET-DRAFT
Intended Status: Proposed Standard
Expires: January 26, 2012

Rick van Rein
OpenFortress
July 25, 2011

IPv6 Tunneling for Embedded Systems (6bed4)
draft-vanrein-v6ops-6bed4-00

Abstract

Given the limited resources available to a lot of embedded systems, dual-stack solutions are not always feasible for such hosts. A mechanism that supports a direct transition from IPv4-only to IPv6-only may prove beneficial in getting the smallest hosts to make a transition to IPv6 at a much earlier stage than would otherwise be possible. This calls for tunnels, but no current tunnel technique appears to be optimal for embedded systems.

This specification details an IPv6 tunneling technique over UDP and IPv4. The technique is specifically designed to benefit embedded systems, and to work without end user configuration. The working principle for obtaining a routable IPv6 address is through stateless autoconfiguration from an anycast tunnel service.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

Copyright and License Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Table of Contents

1. Introduction

The 6bed4 mechanism is a tunnel that encapsulates IPv6 packets in UDP and IPv4. The mechanism assumes a remote tunnel service at a well-known IPv4 address and UDP port, effectively making the tunnel independent of DNS [RFC1034], and capable of traversing NAT [RFC2766]. The local IPv4 address is acquired through traditional techniques such as DHCPv4 [RFC2131] or manual IPv4 configuration, and the local UDP port can be chosen in any way that makes sense locally.

The tunnel service can be implemented at many locations simultaneously, each announcing a route to their well-known IPv4 address over BGP [RFC4271], following the anycast [RFC4786] principle. The routing infrastructure will cause tunnel traffic to be relayed to a nearby instance of the 6bed4 service. In addition to classically routed service, the well-known tunnel service address enables en-route translation.

Embedded systems obtain a routable IPv6 address over the tunnel through stateless autoconfiguration [RFC4862]. The router, always with interface identifier 0, can be reached over the tunnel on the fixed link-local IPv6 address fe80:: or on the all-routers address ff02::2 or on the all-nodes address ff02::1. The tunnel client, being the only other participant in the tunnel, can choose non-zero interface identifiers at will to complete autoconfiguration.

The routable prefix offered by the 6bed4 tunnel service includes the IPv4 address and UDP port as seen from the Internet. Any NAT layer crossed by the tunnel may influence the client-side IPv4 address and UDP port, which is why the tunnel service must provide this data. Autoconfiguration serves to assign an IPv6 address to the tunnel client that incorporates the way it looks from the perspective of the Internet.

1.1 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].

1.2. Well-known Addresses

This specification allocates a number of well-known numbers for addressing. These numbers can be fixated into an embedded system, and serve to address an anycast-published tunnel as a fallback for local IPv6 facilities.

IPv4 address for the remote tunnel service: TBD1

IPv6 prefix for the remote tunnel service: TBD2::/64

UDP port for the remote tunnel service: TBD3

The IPv4 address and UDP port are used to refer to the anycasted tunnel service. The IPv6 prefix is the default prefix for 6bed4 service, although local implementations MAY sometimes use another prefix. The IPv6 prefix is configured as a /64 on tunnel servers, composed of an IANA-registerd /48 and 16 zero bits.

2. Design Overview

The most important desktop systems of today support IPv6 as well as IPv4, and have IPv6 enabled by default. This means that the Internet as a whole can now make the transition to IPv6, without the risk that users will be thrown out. Indeed, this is gradually happening.

Embedded systems however, show a vastly different situation. The support for IPv6 is lagging behind, and numerous manufacturers are not even considering support.

2.1. Rationale - Why Embedded Systems are Different

As long as a majority of Internet users still use IPv4, at least some parties may see no serious economic incentive to migrate to IPv6. Considering that IPv4 is not scheduled to be deprecated anytime soon, this situation may persist for many years to come. This limited incentive to transit to IPv6 applies equally to all host implementations, and will usually be weighed with other arguments.

Specifically for embedded systems, a second force applies. The resources available to such systems are often limited; not only technical limitations such as available memory space, but also economic limitations such as lack of development time and the cost efficiency of support on more complicated implementations.

The possible lack of an economic incentive to adopt IPv6, combined with limited resources, may lead to a situation where embedded systems are the last ones to support IPv6. This is probably aggravated by the fragmented market for embedded systems, compared to the desktop and router markets.

2.2. Solution - IPv6-only on Any Network

The 6bed4 tunnel was created to answer to the specific situation on the market of embedded systems. To resolve any problems related to limited resources, the proposed transition is to go directly from IPv4-only to IPv6-only. The resulting setup can lead to simpler embedded code than in a dual-stack system; it can even save effort otherwise spent on NAT-traversal issues. Finally, any infrastructure built around IPv6-only devices is likely to be simpler than a dual-stack infrastructure; specifically peer-to-peer infrastructure may benefit greatly from the transparency of IPv6 addressing. We expect these motivations to lead to a faster adoption of IPv6 in embedded systems.

If embedded systems are to operate in IPv6-only mode, they should be able to do that on any network, including IPv4-only networks. That calls for a tunneling technique. Several tunnels have been defined, but none would be very useful to convince programmers of embedded systems to transit from IPv4-only to IPv6-only. Because of this, 6bed4 is introduced as an alternate tunneling technique for this specific application area.

The following code fragment shows a procedure for automatic network bootstrapping over IPv6 if it is locally available, with a tunnel fallback based on IPv4 if this is needed. Not shown are alternate options, such as manually configured local IPv4 addresses. The simplicity of this network bootstrapping code means that only minor resources are needed to make a networking application always run on (an illusion of) an IPv6-only network.

```
// Redistribution and use in source and binary forms, with
// or without modification, are permitted provided that the
// following conditions are met:
//
// * Redistributions of source code must retain the above
//   copyright notice, this list of conditions and the
//   following disclaimer.
//
// * Redistributions in binary form must reproduce the
//   above copyright notice, this list of conditions and
//   the following disclaimer in the documentation and/or
//   other materials provided with the distribution.
//
// * Neither the name of Internet Society, IETF or
//   IETF Trust, nor the names of specific contributors,
//   may be used to endorse or promote products derived
//   from this software without specific prior written
//   permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
// CONTRIBUTORS _AS_IS_ AND ANY EXPRESS OR IMPLIED
```



```
// WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
// WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
// PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL
// THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY
// DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
// CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
// PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
// USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
// HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER
// IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
// NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE
// USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
// POSSIBILITY OF SUCH DAMAGE.

err = try_autoconfig6 (&cfg);
if ((err != 0) || managed_flag_dhcp6 (&cfg)) {
   err = try_dhcp6 (&cfg);
   if (err != 0) {
      err = try_dhcp4 (&cfg);
      if (err == 0) {
         use_tunnel (&cfg);
         err = try_autoconfig6 (&cfg);
      }
      if (err) {
         panic ();
      }
   }
}
```

2.3. Requirements - Specifically for Embedded Systems

Following is a list of criteria for tunneling techniques that are necessary to make IPv6 workable for embedded environments; no previously designed tunnel is able to fulfill them all yet:

Standard.
A tunneling approach requires interoperability between tunnel clients and servers; therefore, open standards must be followed.

Simple.
To support embedded systems with limited resources, the tunnel should be as simple as possible. Combined with the addressing transparency of IPv6, the transition to IPv6 can actually be easier than using IPv4, and thus attractive for newly created or upgraded firmware.

Any router.
A tunnel technique that works for embedded systems should not make

any assumptions about the routers on its path; in other words, it should pass through any type of router, including those who rely on symmetrical NAT.

Zero configuration.
Any requirement to manually configure a tunnel will seriously impair the possibilities of rolling out an embedded system to end users. As a result, such a tunnel would not be used in many products. Only if IPv6 works straight out of the box, without manual configuration, is it likely to enter mainstream devices. In technical terms, this means that the tunnel mechanism cannot rely on device or end user credentials.

Traceable.
Even without configurable credentials, a tunnel should not become a portal for network abuse. It should therefore be possible to isolate a misbehaving network node and take corrective measures. One way of making IPv4-registered users traceable in IPv6 is by revealing their IPv4 address as part of their IPv6 address.

Stateless.
If possible, a stateless mechanism for the tunnel service would be ideal. This would make it easy to handle downtime on an instance, because traffic can be diverted elsewhere. Also, there is no need to pass through the same server on bidirectional traffic; this perfectly matches the stateless routing properties of IP-level traffic.

Anycast.
If possible, it would be ideal to have well-known addresses for the tunnel service, so they can be anycasted over BGP4. This leads to a network service that is resilient and that can roughly spread the load over instances. A requirement for this to work would be that the tunnel service is stateless.

2.4. Design - Choices Made

Following is the train of thought that explains how the 6bed4 tunneling mechanism can implement all requirements from the previous subsection:

Not anonymous.
The tunnel user should not be anonymous so abusers can be traced. This is easily achieved by incorporating a public IPv4 address in any IPv6 addresses that is usable through the tunnel.

No registration.
Given that a tunnel user can always be traced back through their

IPv4 address, it is possible to handle tunnel requests without registration of user accounts, or login. The tunnel service will not aggravate problems of address spoofing if it verifies that the IPv6 addresses sent out do indeed match the sender's IPv4 address. That way, egress filtering on IPv4 applies to the IPv6 addresses as well.

Stateless.
Given that there is no registration, a check on traffic from the IPv4 side to the IPv6 side is needed, and a reconstruction of IPv4 and UDP headers when traffic returns from IPv6 to IPv4. If the UDP information (notably, the client-side UDP port on what will often be a NAT router) is also present in the IPv6 addresses used, then the tunnel service has no need for storing any state about the tunnel. It is a stateless, packet-at-a-time translation service.

Anycast.
Stateless services that only perform simple translations of headers can easily be made into anycast services. Indeed, allocating a /64 on the IPv6 side and a /24 on the IPv4 side suffices. Broadcasting these routes from multiple BGP speakers and hosting them in multiple autonomous systems should not be any problem.

3. Link-local Profile

The 6bed4 tunnels add detail with respect to the autoconfiguration mechanism described in RFC 4862. Most importantly, the parameter N for the number of bits in the interface identifier is 16. The preceding 112 bits are filled with a /64 prefix for the IPv6-side of the tunnel service, 32 bits worth of IPv4 address and 16 bits of UDP port number:

```
 0                            63              95     111     127
+-----------------------------+---------------+-------+-------+
|      IPv6-side /64 PREFIX   | Public v4addr |UDPport| if-id |
+-----------------------------+---------------+-------+-------+
```

Or, as seen from the tunnel client perspective:

```
 0                                                   111     127
+---------------------------------------------------+-------+
|                 IPv6-side /112 PREFIX              | if-id |
+---------------------------------------------------+-------+
```

The interface identifier (if-id) for the router is always 0 and only 0, which means that it can always be reached at its link-local address fe80::/128. The tunnel client can choose its own interface identifier(s) at will from the range 1 to 65535 inclusive. The tunnel client MAY fixate interface identifiers in its firmware.

The point-to-point nature makes it possible for tunnel clients and servers to ignore the interface identifier altogether without malfunctioning. Upon sending however, a node MUST use a valid interface identifier to accommodate peers that do check it.

The value of DupAddrDetectTransmits defaults to 0 for this kind of link, meaning that no neighbor discovery is required for 6bed4 links. This is possible because of the static allocation of interface identifiers to endpoint. Note that RFC 4862 requires the tunnel client and server provide a facility for overriding this setting. For this reason, the tunnel endpoints MUST support neighboring requests.

4. Tunneling Traffic

The following describes how to pass traffic down from the IPv6-side of the tunnel to the IPv4-tunneled side; how to pass it up in the opposite direction; and, as an optimisation, how a tunnel may directly connect two tunnels by bouncing traffic to the other side.

4.1. Sending Traffic Up the Tunnel

The embedded system can send IPv6 packets through a tunnel as soon as it has an assigned IPv6 address. To do this, it will prefix a UDP header with the well-known UDP port as a destination and the UDP port used locally for the tunnel as a source. It will then prefix an IPv4 header, containing the well-known IPv4 address of the tunnel as destination, and the classically obtained local IPv4 address as the sender.

This is shipped over the IPv4 network, may pass NAT routers, and will arrive on the tunnel server with possibly altered sender information in the IPv4 and UDP headers.

When traffic is sent to the all-routers multicast address ff02::2, the all-nodes multicast address ff02::1 or to the router's link-local address fe80::/128 it is subject to local handling according to RFC 4861 and RFC 4862. If however, the time-to-live is not 255, the packet MUST be dropped or rejected with Time Exceeded error sent with ICMPv6. Specifically, router solicitation will result in sending a router advertisement, and neighbor solicitation is handled as usual.

Before passing traffic through the tunnel, the time-to-live in the IPv6 packet MUST be decremented. If this is not possible because it

is already 0, then the packet MUST be rejected and a Time Exceeded error sent over ICMPv6 in response.

The tunnel server then verifies the correctness of the sending IPv6 address: The first 64 bits should match the fixed prefix assigned to the tunnel server; the following 32 bits should match the IPv4 address according to the incoming message; the following 16 bits should match the UDP port from which the incoming message came; the final 16 bits with the interface identifier may not be zero, as that is always the tunnel server's address.

If the match is good, the IPv6 payload will be taken out of its IPv4/UDP wrapper, and forwarded as normal traffic over the IPv6 network. In doing so, the DiffServ field SHOULD be preserved. A few exceptional forms of delivery are handled in later sections of this specification.

If the match is false, the tunnel server will send back an unsolicited router advertisement to the origin from which the failing tunnel traffic came, assuming that a 6bed4 endpoint is listening behind that address/port combination. This advertisement will revoke the prefix that the tunnel client tried to use by setting its preferred and valid lifetimes to 0. In the same message, it will advertise the new prefix that holds the external IPv4 address and UDP port of the client, and assign it an infinite preferred and valid lifetime value 0xffffffff.

Upon reception of a router advertisement, the tunnel client MUST immediately update its IPv6 addresses and it MUST NOT send out any further messages using an outdated IPv6 prefix. It MAY resend any unacknowledged messages that are being processed.

4.2. Sending Traffic Down the Tunnel

Traffic that is to be sent down through the tunnel, is routed to a tunnel server by routing to its IPv6 /64 prefix. If this is the well-known prefix, then many BGP speakers may be announcing it; the routing infrastructure would then find a suitable tunnel server nearby the IPv6 sender.

If an entering IPv6 packet has a destination address with a different /64 prefix than the prefix setup for the tunnel, then that packet MUST be dropped or rejected with an ICMPv6 message.

If an IPv6 packet is passed down through the tunnel, its time-to-live MUST be decremented. If this is not possible because the time-to-live is already 0, then the packet MUST be rejected with a Time Exceeded ICMPv6 error message.

The IPv6 address to which the message is sent contains an IPv4 address right after its /64 prefix, followed by 16 bits of client UDP port. The destination side of a UDP header and IPv4 header can be reconstructed from that, and the source side can be filled with the well-known values that have been defined for 6bed4. The DiffServ field of the incoming IPv6 header SHOULD be replicated in the newly constructed IPv4 header. At some point before shipping this message, the last 16 bits of the address, holding the interface identifier, MUST be checked to be non-zero. If the value is zero, the incoming traffic MUST be silently dropped.

4.3. Bouncing Traffic on the Tunnel's IPv4 Side

If a tunnel receives a tunneled package destined for an IPv6 address that begins with the well-known IPv6 /64 prefix for 6bed4 or perhaps a locally used alternative prefix for 6bed4, then it MAY optimize the flow of traffic by forwarding it as tunneled traffic to the IPv4 address and UDP port found in the remainder of the IPv6 destination address.

In doing so, it MUST NOT bypass the comparison of the IPv6 prefix, IPv4 address and UDP port as mentioned in the source IPv6 address. If this comparison fails, the traffic MUST be treated as traffic trying to pass up through the tunnel with an incorrectly set IPv6 address.

5. Tunnel Service Profiles

Three different service profiles are expected to be useful for this tunnel mechanism. The first is public, available to users anywhere in the World. The second profile is local, intended to serve only a part of the Internet, such as the network of a particular provider. The third profile focusses on translation in routers through which the 6bed4-encapsulated traffic happens to flow.

5.1. Public Tunnel Service Profile

This profile describes the situation of a service made publicly available in the spirit of supporting the transition of embedded systems as proposed in this specification.

Such services are announced over BGP, which SHOULD be done for both the IPv4-side and IPv6-side addresses. Given the stateless nature of Internet routing, the upstream and downstream routes may well go through different services, and there can even be situation where traffic is evenly spread over more than one target. For this reason, the addresses announced MUST be the well-known address TBD1/24 for

the IPv4 side and TBD2::/48 for the IPv6 side.

Public Tunnel Service providers MUST register their tunnel service with their Regional Internet Registry, to support finding the Autonomous System that supports the 6bed4 service.

5.2. Local Tunnel Service Profile

This profile describes the situation of a local administrator at any scale from a home or company network to an ISP, offering 6bed4 service to hosts internal to the network.

Such services are announced over interior gateway routing protocols. On the IPv4 side, the well-known address TBD1/24 SHOULD be announced, to provide service any embedded device that was equipped to deal with standard 6bed4 facilities. On the IPv6 side a non-standard address range MUST be used, namely one that routes to the locally available 6bed4 service.

The reason that a local 6bed4 service MAY use a non-standard IPv4 address is that it may be desirable to support other applications than embedded systems, for instance desktops or routers, which MUST NOT put pressure on the public 6bed4 infrastructure. Using another IPv4 address however, releases the pressure that such applications would bring if the locally supported systems switch to external Internet access mechansims. Next to such alternative local use of 6bed4 it is still possible to run a normal 6bed4 service on TBD1/24, using any of the profiles sketched here.

Local tunnel service providers SHOULD register their IPv6 /64 range with their Regional Internet Registry, and MAY refer to this specification in abuse-handling comments.

5.3. En-route Translation Profile

This profile describes routers that translate 6bed4 traffic while it happens to pass through them. This can be done by any router that has both IPv4 and IPv6 connectivity, and that decides to terminate either TBD1/24 or the IPv6-side address range.

En-route translation need not be announced publicly, but it may help to direct traffic to such routers if there would otherwise be alternate paths around it. For example, if all gateways of an Internet Service Provider implement 6bed4 in an en-route profile, then it may not be necessary to announce this service on the local network or on the Internet at large.

A few variations exist regarding the prefix used on the IPv6 side of en-route translation. If all upstream traffic (for a targeted network) is guaranteed to pass through a router that performs en-route 6bed4 translation, then the IPv6 side SHOULD use an IPv6 address that routes to the local network, but it MAY use TBD2::/64 if downstream 6bed4 service is not locally available. On the other hand, if not all upstream traffic (for the target network) passes through an en-route translating router, then its IPv6 side MUST use the well-known prefix TBD2::/64 so that alternate routes and the en-route translation cannot be distinguished from the end user's perspective.

The major advantage of en-route translation is that it can avoid diversion of traffic through a third-party 6bed4 tunnel server. Instead of routing the traffic in two legs, going through an intermediary, it can be kept on the fastest route available, which is good for the routing budget of the Internet as a whole. It may even cause the IPv6 traffic to remain on the local network of the Internet Service Provider.

En-route translation providers SHOULD register their IPv6 /64 range with their Regional Internet Registry, and MAY refer to this specification in abuse-handling comments.

6. NAT-traversal Issues

Very often, a 6bed4 tunnel will have to pass through a layer of network address translation, or NAT. Even if 6bed4 recovers the transparency of network connections at the IPv6 level, there may still be simpler problems to resolve at an underlying IPv4 level. NAT will rewrite IPv4 and UDP-over-IPv4 source addresses. Handling these translations has been designed into 6bed4.

A remaining concern with NAT is that UDP has no connection status, and its translation must therefore be flushed at some point. Although the 6bed4 tunnel will quickly recover when that happens, the higher protocols may not be as accommodating; notably, TCP connections over IPv6 would break if the translation changed suddenly.

For this reason, it may be necessary to send messages for the purpose of keeping the address translation active in any on-route NAT routers. This can either be achieved by an actively communicating protocol, or by explicit keep-alive messages.

Any such messages SHOULD NOT be sent if there is no application need for keeping the same IPv6 address on which the tunnel client can be reached. Keep-alive messages MUST NOT be sent more often than once in 25 seconds. Furthermore, randomization of the keep-alive message interval is important so as to offload the tunnel server from

synchronization of keep-alive messages after things like power outages.

Instead of sending keep-alive messages, a device MAY use other means to achieve the longevity of an open link. If it can communicate its wish for an open UDP port directed to its local endpoint directly, this is a much simpler method. The only disadvantage of this approach is usually that it cannot be relied upon as a general mechanism; but if it does, it will save energy and bandwidth, so it is certainly recommended.

7. Security Considerations

Any party that can convince a network of being the router for a given range of addresses will be able to attract the traffic for 6bed4 tunnels. This could open up such protocols for man-in-the-middle attacks.

The foreseeable means of doing this are either through BGP advertisements on the Internet, or through router advertisements on a local network. The issue of BGP advertisements is a general problem and is not commonly thought of as being hazardous; notwithstanding that, work is being done to solve the general problem at that general level.

When done at the local level using an interior gateway routing protocol, the attack is not much different from DHCP hijacking, a risk that is usually dealt with locally by monitoring client nodes to not exhibit server behavior, or by strict control over network access.

Although symmetric signatures are possible over neighbor discovery protocols, this is not usable for the 6bed4 system, because it is a global protocol and includes too many parties to be able to protect the shared secret keys. Any signature mechanism for 6bed4 would have to be asymmetrical.

8. IANA Considerations

This document allocates three resources from existing registries.

First, the well-known IPv4 address TBD1 under its own /24 prefix, to be registered in the IPv4 Special Purpose Address Registry. Although only a single address is required, the choice of a /24 integrates better with BGP routing practices. [TO BE REMOVED: 192.64.64.1/24 or 192.6.4.1/24 would be nice!] [TO BE REMOVED: This registration should take place at the following location: http://www.iana.org/assignments/iana-ipv4-special-registry/iana-ipv4-special-registry.xml]

Second, the well-known IPv6 prefix TBD2::/48, to be registered in the IANA IPv6 Special Purpose Address Registry. Although only a /64 prefix is needed, the choice of a /48 integrates better with BGP routing practices. The additional bits to come to a /64 prefix are filled with zeroes. [TO BE REMOVED: The address can be taken from 2001:0000::/23] [TO BE REMOVED: 2001:6:bed4::/48 would be nice!] [TO BE REMOVED: This registration should take place at the following location: http://www.iana.org/assignments/iana-ipv6-special-registry/iana-ipv6-special-registry.xml]

Thirdly, the well-known UDP port TBD3, to be registered in the Port Numbers Registry. [TO BE REMOVED: Suggest to share 3653 for comparable & interoperable TSP from RFC 5572; draft author contacted Marc Blanchet with this sharing proposal] [TO BE REMOVED: This registration should take place at the following location: http://www.iana.org/assignments/port-numbers]

9. References

9.1. Normative References

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

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

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, March 2006.

[RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.

[RFC4786] Abley, J. and K. Lindqvist, "Operation of Anycast Services", BCP 126, RFC 4786, December 2006.

[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, December 1998.

[RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de 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.

9.2. Informative References

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

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, March 1997.

Author's Address

Rick van Rein
OpenFortress Digital signatures
Haarlebrink 5
7544 WP Enschede
The Netherlands

email: rick@openfortress.nl