RFC6296: NPTv6 to Standards Track

Margaret Cullen, Fred Baker, Nick Buraglio, Ole Trøan

Terminology

- NPTv6 = This proposal
  - 1:1, stateless, algorithmic, checksum-neutral prefix translation
  - Primary use cases are address independence and site multi-homing for sites using provider-allocated addresses
- NAPT = Stateful NAT with port translation
  - Widely used in v4 for address sharing, address independence, site multi-homing, firewall-like properties, topology hiding ,etc.
- NAT = Stateful address translation, without port translation
  - May be used for other NAT benefits when address sharing isn't needed

- **NAT66 = Ambiguous term sometimes used to refer to NPTv6, sometimes used to refer to v6-to-v6 NAT or NAPT. Try not to use it in today's discussion.**

Background

- IPv4 Address shortage was only one of the scaling issues threatening the continued growth of the Internet in ~2010
  - An increase in the number of IP addresses is a key feature of IPv6, which was already available to address this issue.
- There was widespread concern about the growth of the global routing tables
  - Most of the growth was caused by the use of Provider-Independent (PI) prefixes and site multi-homing, both of which required long prefixes in BGP that limited route aggregation
- We moved toward provider-allocated (PA) addresses to slow the growth
  - But edge networks still needed address independence and multi-homing, both of which are often achieved in IPv4 by using IPv4 NAPT with associated tradeoffs.

Schools of Thought

- The community realized that these issues were largely caused by the fact than IP Address conflates the “location” of a node with the “identity” of a node, resulting several ID/Locator split proposals (8+8, GSM, LISP, 1:1 NAT, etc.)
- These proposals fall into two categories:
  - Address Rewriting solutions (8+8, GSM, 1:1 NAT, etc.)
  - Tunneling proposals (LISP, etc.)
- Ultimately, these categories form a continuum, because address rewriting mechanisms are similar to degenerate tunnels with the ID/Locator mapping information stored in a middlebox at the edge of the local network instead of being transmitted in every packet.
- NPTv6 is an address remapping mechanism that eliminates most of the issues caused by tunnels or other address-rewriting proposals, because it uses a stateless, algorithmic, checksum-neutral, 1:1 address remapping mechanism

Address Rewriting vs Tunnel Tradeoffs (before NPTv6)

Address Rewriting

- Nodes do not know their global addresses
  - Requires that applications use DNS names or have ALGs to translate application-layer addresses
- Transport layer checksum corrections needed
  - Requires rewriting of upper layers that use IP pseudo-header checksums (UDP, TCP, etc)
- Stateful translation
  - Creates brittleness – all connections drop if router reboots or state times out
  - Complicates dynamic routing, asymmetrical routing and multi-homing
- Need for topology-aware or split DNS

TUNNELS

- Need for mapping function between inner addresses (IDs) and outer addresses (Locators)
- MTU Issues
- Not incrementally deployable – benefit only realized when the other side of a connection has a tcompatible unnel endpoint
- Need for topology-aware or split DNS

NPTv6 Tradeoffs

1:1 NATs

- Nodes do not know their global addresses
  - Requires that applications use DNS names or have ALGs to translate application-layer addresses
- ~~Transport layer checksum corrections needed~~
  - ~~Requires rewriting of upper layers that use IP pseudo-header checksums (UDP, TCP, etc)~~
- ~~Stateful address translation~~
  - ~~Creates brittleness -- all connections drop if router reboots or state times out~~
  - ~~Complicates dynamic routing, asymmetrical routing and multi-homing~~
- Need for topology-aware or split DNS

TUNNELS

- ~~Need for mapping function between inner addresses (IDs) and outer addresses (Locators)~~
- ~~MTU Issues~~
- ~~Not incrementally deployable -- benefit only realized when the other side of a connection has a tcompatible unnel endpoint~~
- Need for topology-aware or split DNS

Document History:

- Originally published as NAT66 in an individual draft in 2008
- Discussed in the behave WG through 2009
- Republish in 2010 as NPTv6, discussed in 6man, v6ops and elsewhere
  - Name change emphasized the significant differences between NPTv6 and traditional IPv4 Network Address/Port Translation (NAPT)
  - Added code and proof that the algorithm would produce 1:1 results
- Published in 2011 as an IETF Experimental RFC (RFC 6296)
  - As was typical at that time, no "Experiment" was explicitly defined in the RFC,
  - The document says: “it is published for examination, experimental implementation, and evaluation.”

How did the experiment go?

- Over the last 12 years, NPTv6 has been implemented by many vendors, and widely used to solve the problems it was intended to solve
- NPTv6 has been implemented by Cisco, Juniper, Huawei, VyOs, Palo Alto, H3C, A10, OPNsense, pfSense, Check Point, Microtik, Linux (various), NetBSD, and others.
- NPTv6 has been widely used to protect edge networks from ISP renumbering and ISP changes, and to simplify deployment of multi-homed edge networks.without the need for PI addresses
  - No need to renumber internal nodes, access control lists, logs, etc.
  - No need to inject long prefix routes into the global routing tables
  - Incrementally deployable – one site that deploys it locally reaps the benefits
- There is no solution to these problems without compromises, and NPTv6 is far less disruptive than NAPT66

How did the experiment go?

- Over the la endors, and widely use(
- NPTv6 has Palo Alto, H3C, A10, ious), NetBSD, a
- NPTv6 has enumbering and ISP ch e networks.w
  - No neec
  - No neec
- There is nc NPTv6 is far less dis

Why RFC6296bis?

- Why not just write a short document changing the status of existing RFC?
- Some minor document changes are needed:
  - Add section on ICMPv6 error handling (RFC5508 equivalent)
  - Cleanup a couple of editorial issues (a few typos and one sentence fragment)
  - Incorporate errata (1)

Why reclassification as standards track?

- Experiment has been succesful
- The mechanism is technically mature and widely deployed.
- The specification is well understood.

- A document update is planned, and it makes sense to publish the updated RFC on the standards track to reflect its widespread implementation and deployment

Is there general agreement to move this document to standards track and work on it here in 6man?

(or at least not violent disagreement)