Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 26, 2015

S. Perreault
Jive Communications
T. Tsou
Huawei Technologies (USA)
C. Zhou
Huawei Technologies
P. Fan
China Mobile

March 25, 2015

Gap Analysis for IPv4 Sunset
draft-ietf-sunset4-gapanalysis-06

Abstract

Sunsetting IPv4 refers to the process of turning off IPv4 definitively. It can be seen as the final phase of the migration to IPv6. This memo enumerates difficulties arising when sunsetting IPv4, and identifies the gaps requiring additional work.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 26, 2015.

Table of Contents

1. Introduction

The final phase of the migration to IPv6 is the sunset of IPv4, that is turning off IPv4 definitively on the attached networks and on the upstream networks.

Some current implementation behavior makes it hard to sunset IPv4. Additionally, some new features could be added to IPv4 to make its sunsetting easier. This document analyzes the current situation and proposes new work in this area.

The decision about when to turn off IPv4 is out of scope. This document merely attempts to enumerate the issues one might encounter if that decision is made.

2. Related Work

[RFC3789], [RFC3790],[RFC3791], [RFC3792], [RFC3793], [RFC3794], [RFC3795] and [RFC3796] contain surveys of IETF protocols with their IPv4 dependencies.

Additionally, although reviews in RFCs 3789-3796 ensured that IETF standards then in use could support IPv6, no IETF-wide effort has been undertaken to ensure that the issues identified in those drafts are all addressed, nor to ensure that standards written after RFC3100 (where the previous review efforts stopped) function properly on IPv6-only networks.

The IETF needs to ensure that existing standards and protocols have been actively reviewed, and any parity gaps either identified so that they can be fixed, or documented as unnecessary to address because it is unused or superseded by other features.

First, the IETF must review RFCs 3789-3796 to ensure that any gaps in specifications identified in these documents and still in active use have been updated as necessary to enable operation in IPv6-only environments (or if no longer in use, are declared historic).

Second, the IETF must review documents written after the existing review stopped (according to RFC 3790, this review stopped with approximately RFC 3100) to identify specifications where IPv6-only operation is not possible, and update them as necessary and appropriate, or document why an identified gap is not an issue i.e. not necessary for functional parity with IPv4.

This document does not recommend excluding Informational and BCP RFCs as the previous effort did, due to changes in the way that these documents are used and their relative importance in the RFC Series. Instead, any documents that are still active (i.e. not declared historic or obsolete) and the product of IETF consensus (i.e. not a product of the ISE Series) should be included. In addition, the reviews undertaken by RFCs 3789-3796 were looking for "IPv4 dependency" or "usage of IPv4 addresses in standards". This document recommends a slightly more specific set of criteria for review: review should include consideration of whether the specification can operate in an environment without IPv4. Reviews should include guidance on the use of 32-bit identifiers that are commonly populated by IPv4 addresses. Reviews should include consideration of protocols on which specifications depend or interact, to identify indirect dependencies on IPv4. Finally, reviews should consider how to migrate from an IPv4 environment to an IPv6 environment.

3. Remotely Disabling IPv4

3.1. Indicating that IPv4 connectivity is unavailable

PROBLEM 1: When an IPv4 node boots and requests an IPv4 address (e.g., using DHCP), it typically interprets the absence of a response as a failure condition even when it is not.

PROBLEM 2: Home router devices often identify themselves as default routers in DHCP responses that they send to requests coming from the LAN, even in the absence of IPv4 connectivity on the WAN.

3.2. Disabling IPv4 in the LAN

PROBLEM 3: IPv4-enabled hosts inside an IPv6-only LAN can auto-configure IPv4 addresses [RFC3927] and enable various protocols over IPv4 such as mDNS [I-D.cheshire-dnsext-multicastdns] and LLMNR [RFC4795]. This can be undesirable for operational or security reasons, since in the absence of IPv4, no monitoring or logging of IPv4 will be in place.

PROBLEM 4: IPv4 can be completely disabled on a link by filtering it on the L2 switching device. However, this may not be possible in all cases or may be too complex to deploy. For example, an ISP is often not able to control the L2 switching device in the subscriber home network.

PROBLEM 5: A host with only Link-Local IPv4 addresses will "ARP for everything", as described in Section 2.6.2 of [RFC3927]. Applications running on such a host connected to an IPv6-only network will believe that IPv4 connectivity is available, resulting in various bad or sub-optimal behavior patterns. See [I-D.yourtchenko-ipv6-disable-ipv4-proxyarp] for further analysis.

Some of these problems were described in [RFC2563], which standardized a DHCP option to disable IPv4 address auto-configuration. However, using this option requires running an IPv4 DHCP server, which is contrary to the goal of IPv4 sunsetting.

4. Client Connection Establishment Behavior

PROBLEM 6: Happy Eyeballs [RFC6555] refers to multiple approaches to dual-stack client implementations that try to reduce connection setup delays by trying both IPv4 and IPv6

paths simultaneously. Some implementations introduce delays which provide an advantage to IPv6, while others do not [Huston2012]. The latter will pick the fastest path, no matter whether it is over IPv4 or IPv6, directing more traffic over IPv4 than the other kind of implementations. This can prove problematic in the context of IPv4 sunsetting, especially for Carrier-Grade NAT phasing out because CGN does not add significant latency that would make the IPv6 path more preferable. Traffic will therefore continue using the CGN path unless other network conditions change.

PROBLEM 7: getaddrinfo() [RFC3493] sends DNS queries for both A and AAAA records regardless of the state of IPv4 or IPv6 availability. The AI_ADDRCONFIG flag can be used to change this behavior, but it relies on programmers using the getaddrinfo() function to always pass this flag to the function. The current situation is that in an IPv6-only environment, many useless A queries are made.

5. Disabling IPv4 in Operating System and Applications

It is possible to completely remove IPv4 support from an operating system as has been shown by the work of Bjoern Zeeb on FreeBSD. [Zeeb] Removing IPv4 support in the kernel revealed many IPv4 dependencies in libraries and applications.

PROBLEM 8: Completely disabling IPv4 at runtime often reveals implementation bugs. Hard-coded dependencies on IPv4 abound, such as on the 127.0.0.1 address assigned to the loopback interface. It is therefore often operationally impossible to completely disable IPv4 on individual nodes.

PROBLEM 9: In an IPv6-only world, legacy IPv4 code in operating systems and applications incurs a maintenance overhead and can present security risks.

6. On-Demand Provisioning of IPv4 Addresses

As IPv6 usage climbs, the usefulness of IPv4 addresses to subscribers will become smaller. This could be exploited by an ISP to save IPv4 addresses by provisioning them on-demand to subscribers and reclaiming them when they are no longer used. This idea is described in [I-D.fleischhauer-ipv4-addr-saving] and [BBF.TR242] for the context of PPP sessions. In these scenarios, the home router is responsible for requesting and releasing IPv4 addresses, based on

snooping the traffic generated by the hosts in the LAN, which are still dual-stack and unaware that their traffic is being snooped.

PROBLEM 10: Dual-stack hosts that implement Happy-Eyeballs [RFC6555] will generate both IPv4 and IPv6 traffic even if the algorithm end up choooosing IPv6. This means that an IPv4 address will always be requested by the home router, which defeats the purpose of on-demand provisioning.

PROBLEM 11: Many operating systems periodically perform some kind of network connectivity check as long as an interface is up. Similarly, applications often send keep-alive traffic continuously. This permanent "background noise" will prevent an IPv4 address from being released by the home router.

PROBLEM 12: Hosts in the LAN have no knowledge that IPv4 is available to them on-demand only. If they had explicit knowledge of this fact, they could tune their behaviour so as to be more conservative in their use of IPv4.

PROBLEM 13: This mechanism is only being proposed for PPP even though it could apply to other provisioning protocols (e.g., DHCP).

7. IPv4 Address Literals

IPv4 addresses are often used as resource locators. For example, it is common to encounter URLs containing IPv4 address literals on web sites [I-D.wing-behave-http-ip-address-literals]. IPv4 address literals may be published on media other than web sites, and may appear in various forms other than URLs. For the operating systems which exhibit the behavior described in [I-D.yourtchenko-ipv6-disable-ipv4-proxyarp], this also means an increase in the broadcast ARP traffic, which may be undesirable.

PROBLEM 14: IPv6-only hosts are unable to access resources identified by IPv4 address literals.

8. Managing Router Identifiers

IPv4 addresses are often conventionally chosen to number a router ID, which is used to identify a system running a specific protocol. The common practice of tying an ID to an IPv4 address gives much operational convenience. A human-readable ID is easy for network operators to deal with, and it can be auto-configured, saving the work of planning and assignment. It is also helpful to quickly

perform diagnosis and troubleshooting, and easy to identify the availability and location of the identified router.

PROBLEM 15: In an IPv6 only network, there is no IP address that can be directly used to number a router ID. IDs have to be planned individually to meet the uniqueness requirement. Tying the ID directly to an IP address which yields human-friendly, auto-configured ID that helps with troubleshooting is not possible.

9. IANA Considerations

None.

10. Security Considerations

It is believed that none of the problems identified in this draft are security issues.

11. Acknowledgements

Thanks in particular to Andrew Yourtchenko, Lee Howard, Nejc Skoberne, and Wes George for their thorough reviews and comments.

Special thanks to Marc Blanchet who was the driving force behind this work and to Jean-Philippe Dionne who helped with the initial version of this document.

12. Informative References

[BBF.TR242]
Broadband Forum, "TR-242: IPv6 Transition Mechanisms for Broadband Networks", August 2012.

[Huston2012]
Huston, G. and G. Michaelson, "RIPE 64: Analysing Dual Stack Behaviour and IPv6 Quality", April 2012.

[I-D.cheshire-dnsext-multicastdns]
Cheshire, S. and M. Krochmal, "Multicast DNS", draft-cheshire-dnsext-multicastdns-15 (work in progress), December 2011.

[I-D.fleischhauer-ipv4-addr-saving]
Fleischhauer, K. and O. Bonness, "On demand IPv4 address provisioning in Dual-Stack PPP deployment scenarios", draft-fleischhauer-ipv4-addr-saving-03 (work in progress), August 2012.

[I-D.wing-behave-http-ip-address-literals]
Wing, D., "Coping with IP Address Literals in HTTP URIs with IPv6/IPv4 Translators", draft-wing-behave-http-ip-address-literals-02 (work in progress), March 2010.

[I-D.yourtchenko-ipv6-disable-ipv4-proxyarp]
Yourtchenko, A. and O. Owen, "Disable "Proxy ARP for Everything" on IPv4 link-local in the presence of IPv6 global address", draft-yourtchenko-ipv6-disable-ipv4-proxyarp-00 (work in progress), May 2013.

[RFC2563] Troll, R., "DHCP Option to Disable Stateless Auto-Configuration in IPv4 Clients", RFC 2563, May 1999.

[RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, February 2003.

[RFC3789] Nesser, P. and A. Bergstrom, "Introduction to the Survey of IPv4 Addresses in Currently Deployed IETF Standards Track and Experimental Documents", RFC 3789, June 2004.

[RFC3790] Mickles, C. and P. Nesser, "Survey of IPv4 Addresses in Currently Deployed IETF Internet Area Standards Track and Experimental Documents", RFC 3790, June 2004.

[RFC3791] Olvera, C. and P. Nesser, "Survey of IPv4 Addresses in Currently Deployed IETF Routing Area Standards Track and Experimental Documents", RFC 3791, June 2004.

[RFC3792] Nesser, P. and A. Bergstrom, "Survey of IPv4 Addresses in Currently Deployed IETF Security Area Standards Track and Experimental Documents", RFC 3792, June 2004.

[RFC3793] Nesser, P. and A. Bergstrom, "Survey of IPv4 Addresses in Currently Deployed IETF Sub-IP Area Standards Track and Experimental Documents", RFC 3793, June 2004.

[RFC3794] Nesser, P. and A. Bergstrom, "Survey of IPv4 Addresses in Currently Deployed IETF Transport Area Standards Track and Experimental Documents", RFC 3794, June 2004.

[RFC3795] Sofia, R. and P. Nesser, "Survey of IPv4 Addresses in Currently Deployed IETF Application Area Standards Track and Experimental Documents", RFC 3795, June 2004.

[RFC3796] Nesser, P. and A. Bergstrom, "Survey of IPv4 Addresses in Currently Deployed IETF Operations & Management Area Standards Track and Experimental Documents", RFC 3796, June 2004.

[RFC3927] Cheshire, S., Aboba, B., and E. Guttman, "Dynamic Configuration of IPv4 Link-Local Addresses", RFC 3927, May 2005.

[RFC4795] Aboba, B., Thaler, D., and L. Esibov, "Link-local Multicast Name Resolution (LLMNR)", RFC 4795, January 2007.

[RFC6555] Wing, D. and A. Yourtchenko, "Happy Eyeballs: Success with Dual-Stack Hosts", RFC 6555, April 2012.

[Zeeb] "FreeBSD Snapshots without IPv4 support", <http://wiki.freebsd.org/IPv6Only>.

Appendix A. Solution Ideas

A.1. Remotely Disabling IPv4

A.1.1. Indicating that IPv4 connectivity is unavailable

One way to address these issues is to send a signal to a dual-stack node that IPv4 connectivity is unavailable. Given that IPv4 shall be off, the message must be delivered through IPv6.

A.1.2. Disabling IPv4 in the LAN

One way to address these issues is to send a signal to a dual-stack node that auto-configuration of IPv4 addresses is undesirable, or that direct IPv4 communication between nodes on the same link should not take place.

A signalling protocol equivalent to the one from [RFC2563] but over IPv6 is necessary, using either Router Advertisements or DHCPv6.

Furthermore, it could be useful to have L2 switches snoop this signalling and automatically start filtering IPv4 traffic as a consequence.

Finally, it could be useful to publish guidelines on how to safely block IPv4 on an L2 switch.

A.2. Client Connection Establishment Behavior

Recommendations on client connection establishment behavior that would facilitate IPv4 sunsetting would be appropriate.

A.3. Disabling IPv4 in Operating System and Applications

It would be useful for the IETF to provide guidelines to programmers on how to avoid creating dependencies on IPv4, how to discover existing dependencies, and how to eliminate them. Having programs and operating systems that behave well in an IPv6-only environment is a prerequisite for IPv4 sunsetting.

A.4. On-Demand Provisioning of IPv4 Addresses

No idea.

A.5. Managing Router Identifiers

Router IDs can be manually planned, possibly with some hierarchy or design rule, or can be created automatically. A simple way of automatic creation is to generate pseudo-random numbers, and one can use another source of data such as the clock time at boot or configuration time to provide additional entropy during the generation of unique IDs. Another way is to hash an IPv6 address down to a value as ID. The hash algorithm is supposed to be known and the same across the domain. Since typically the number of routers in a domain is far smaller than the value range of IDs, the hashed IDs are hardly likely to conflict with each other, as long as the hash algorithm is not designed too badly. It is necessary to be able to override the automatically created value, and desirable if the mechanism is provided by the system implementation.

If the ID is created from IPv6 address, e.g. by hashing from an IPv6 address, then naturally it has relationship with the address. If the ID is created regardless of IP address, one way to build association with IPv6 address is to embed the ID into an IPv6 address that is to be configured on the router, e.g. use a /96 IPv6 prefix and append it with a 32-bit long ID. One can also use some record keeping mechanisms, e.g. text file, DNS or other provisioning system like network management system to manage the IDs and mapping relations with IPv6 addresses, though extra record keeping does introduce additional work.

Authors' Addresses

Simon Perreault
Jive Communications
Quebec, QC
Canada

Email: sperreault@jive.com

Tina Tsou
Huawei Technologies (USA)
2330 Central Expressway
Santa Clara, CA 95050
USA

Phone: +1 408 330 4424
Email: tina.tsou.zouting@huawei.com

Cathy Zhou
Huawei Technologies
Huawei Industrial Base
Bantian, Shenzhen
China

Email: cathy.zhou@huawei.com

Peng Fan
China Mobile
32 Xuanwumen West Street
Beijing, Beijing
China

Email: fanp08@gmail.com