Network Working Group	W. Cerveny
Internet-Draft	Arbor Networks
Intended status: Informational	R. Bonica
Expires: January 6, 2016	Juniper Networks
	July 5, 2015

Benchmarking IPv6 Neighbor Cache Behavior

draft-ietf-bmwg-ipv6-nd-00

Abstract

This document is a benchmarking instantiation of RFC 6583: “Operational Neighbor Discovery Problems” [RFC6583]. It describes a general testing procedure and measurements that can be performed to evaluate how the problems described in RFC 6583 may impact the functionality or performance of intermediate nodes.

Requirements Language

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].

Status of This Memo

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

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

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

This Internet-Draft will expire on January 6, 2016.

Copyright Notice

Table of Contents

1. Introduction

This document is a benchmarking instantiation of RFC 6583: “Operational Neighbor Discovery Problems” [RFC6583]. It describes a general testing procedure and measurements that can be performed to evaluate how the problems described in RFC 6583 may impact the functionality or performance of intermediate nodes.

2. Terminology

Intermediate Node
: A router, switch, firewall or any other device which separates end-nodes. The tests in this document can be completed with any intermediate node which maintains a neighbor cache, although not all measurements and performance characteristics may apply.

Neighbor Cache
: The neighbor cache is a database which correlates the link-layer address and the adjacent interface with an IPv6 address.

Neighbor Discovery
: See Section 1 of RFC 4861 [RFC4861]

Scanner Network
: The network from which the scanning tester is connected.

Scanning Interface
: The interface from which the scanning activity is initiated.

Stale Entry Time
: This is the duration for which a neighbor cache entry marked "Reachable" will continue to be marked "Reachable" if an update for the address is not received.

Target Network

The network for which the scanning tests is targeted.

Target Network Destination Interface
: The interface that resides on the target network, which is primarily used to measure DUT performance while the scanning activity is occurring.

3. Overview of Relevant NDP and Intermediate Node Behavior

In a traditional network, an intermediate node must support a mapping between a connected node's IP address and the connected node's link-layer address and interface the node is connected to. With IPv4, this process is handled by ARP [RFC0826]. With IPv6, this process is handled by NDP and is documented in [RFC4861]. With IPv6, when a packet arrives on one of an intermediate node's interfaces and the destination address is determined to be reachable via an adjacent network:

1. The intermediate node first determines if the destination IPv6 address is present in its neighbor cache.
2. If the address is present in the neighbor cache, the intermediate node forwards the packet to the destination node using the appropriate link-layer address and interface.
3. If the destination IPv6 address is not in the intermediate node's neighbor cache:
    1. An entry for the IPv6 address is added to the neighbor cache and the entry is marked "INCOMPLETE".
    2. The intermediate node sends a neighbor solicitation packet to the solicited-node multicast address on the interface considered on-link.
    3. If a solicited neighbor advertisement for the IPv6 address is received by the intermediate node, the neighbor cache entry is marked "REACHABLE" and remains in this state for 15 to 45 seconds.
    4. If a neighbor advertisement is not received, the intermediate node will continue sending neighbor solicitation packets every second until either a neighbor solicitation is received or the maximum number of solicitations has been sent. If a neighbor advertisement is not received in this period, the entry can be discarded.

There are two scenarios where a neighbor cache can grow to a very large size:

1. There are a large number of real nodes connected via an intermediate node's interface and a large number of these nodes are sending and receiving traffic simultaneously.
2. There are a large number of addresses for which a scanning activity is occuring and no real node will respond to the neighbor solicitation. This scanning activity can be unintentional or malicious. In addition to maintaining the "INCOMPLETE" neighbor cache entry, the intermediate node must send a neighbor solicitation packet every second for the maximum number of socicitations. With today's network link bandwidths, a scanning event could cause a lot of entries to be added to the neighbor cache and solicited for in the time that it takes for a neighbor cache entry to be discarded.

An intermediate node's neighbor cache is of a finite size and can only accommodate a specific number of entries, which can be limited by available memory or a preset operating system limit. If the maximum number of entries in a neighbor cache is reached, the intermediate node must either drop an existing entry to make space for the new entry or deny the new IP address to MAC address/ interface mapping with an entry in the neighbor cache. In an extreme case, the intermediate node's memory may become exhausted, causing the intermediate node to crash or begin paging memory.

At the core of the neighbor discovery problems presented in RFC 6583 [RFC6583], unintentional or malicious IPv6 traffic can transit the intermediate node that resembles an IP address scan similar to an IPv4-based network scan. Unlike IPv4 networks, an IPv6 end network is typically configured with a /64 address block, allowing for upwards of 2**64 addresses. When a network node attempts to scan all the addresses in a /64

address block directly attached to the intermediate node, it is possible to create a huge amount of state in the intermediate node's neighbor cache, which may stress processing or memory resources.

Section 7.1 of RFC 6583 recommends how intermediate nodes should behave when the neighbor cache is exceeded. Section 6 of RFC 6583 [RFC6583] recommends how damage from an IPv6 address scan may be mitigated. Section 6.2 of RFC 6583 [RFC6583] discusses queue tuning.

4. Test Setup

The network needs to minimally have two subnets: one from which the scanner(s) source their scanning activity and the other which is the target network of the address scans.

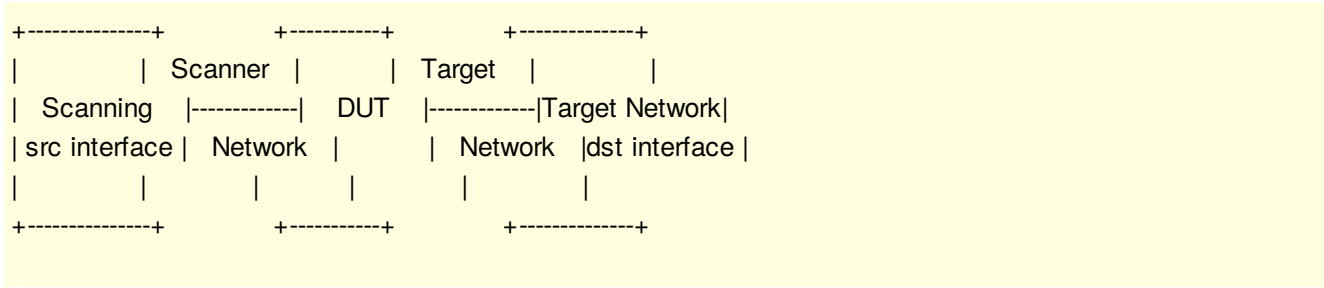
It is assumed that the latency for all network segments is neglible. By default, the target network's subnet shall be 64-bits in length, although some tests may involve increasing the prefix length.

Although packet size shouldn't have a direct impact, packet per second (pps) rates will have an impact. Smaller packet sizes should be utilized to facilitate higher packet per second rates.

For purposes of this test, the packet type being sent by the scanning device isn't important, although most scanning applications might want to send packets that would elicit responses from nodes within a subnet (such as an ICMPv6 echo request). Since it is not intended that responses be evoked from the target network node, such packets aren't necessary.

At the beginning of each test the intermediate node should be initialized. Minimally, the neighbor cache should be cleared.

Basic format of test network.



4.1. Testing Interfaces

Two tester interfaces are configured for most tests:

- Scanning source (src) interface: This is the interface from which test packets are sourced. This interface sources traffic to destination IPv6 addresses on the target network from a single link-local address, similar to how an adjacent intermediate node would transit traffic through the intermediate node.
- Target network destination (dst) interface: This interface responds to neighbor solicitations as appropriate and confirms when an intermediate node has forwarded a packet to the interface for consumption. Where appropriate, the target network destination interface will respond to neighbor solicitations with a unique link-layer address per IPv6 address solicited.

5. Modifiers (Variables)

5.1. Frequency of NDP Triggering Packets

The frequency of NDP triggering packets can be as high as the maximum packet per second rate that the scanner network will support (or is rated for). However, it may not be necessary to send packets at a particularly high rate. In fact, a non-benchmarking goal of testing could be to identify if the DUT is able to

withstand scans at rates which otherwise would not impact the performance of the DUT.

Optimistically, the scanning rate should be incremented until the DUT's performance begins deteriorating. Depending on the software and system being used to implement the scanning, it may be challenging to achieve a sufficient rate. Where this maximum threshold cannot be determined, the test results should note the highest rate tested and that DUT performance deterioration was not noticed at this rate.

The lowest rate tested should be the rate for which packets can be expected to have an impact on the DUT — this value is of course, subjective.

6. Tests

6.1. Stale Entry Time Determination

This test determines the time interval when the intermediate node (DUT) identifies an address as stale.

RFC 4861, section 6.3.2 [RFC4861] states that an address can be marked "stale" at a random value between 15 and 45 seconds (as defined via constants in the RFC). This test confirms what value is being used by the intermediate node. Note that RFC 4861 states that this random time can be changed "at least every few hours."

6.1.1. General Testing Procedure

1. Send a packet from the scanning source interface to an address in target network. Observe that the intermediate node sends a neighbor solicitation to the solicited-node multicast address on the target network, for which tester destination interface should respond with a neighbor advertisement. The intermediate node should create an entry in neighbor cache for the address, marking the address as "reachable". As this point, the packet should be forwarded to the tester destination interface.
2. After the neighbor advertisement from the destination tester interface in step one, no more neighbor advertisements from the tester destination interface should be allowed.
3. Continue sending packets from the scanning source interface to the same address in the target network.
4. Note the time at which the DUT no longer forwards packets. The stale timer value will be the period of time between when the DUT received the first neighbor advertisement above and the point at which the DUT no longer forwards packets for this flow to the tester destination interface.

6.2. Neighbor Cache Exhaustion Determination

Discover the point at which the neighbor cache is exhausted and evaluate intermediate node behavior when this threshold is reached. If possible, the stale timer value should be locked down to a large value. A side-effect of this test is to confirm that intermediate node behaves correctly; in particular, it shouldn't crash.

Note that some intermediate nodes may restrict the frequency of allowed neighbor discovery packets transmitted. The maximum allowed packets per second must either be set to a value which doesn't impact the outcome of the test must allow for this restriction.

6.2.1. General Testing Procedure

1. At a very fast rate, send packets incrementally to valid unique addresses in the target network, within stale entry time period. Simultaneously, send packets for addresses previously added to the neighbor cache. The neighbor cache has been exhausted when previously added addresses must be re-discovered with a neighbor solicitation (within the stale entry time period).

2. Observe what happens when one address greater than the maximum neighbor cache size ("n") is reached. When "n+1" is reached, if either the first or most recent cache entry are dropped, this may be acceptable.
3. Confirm intermediate node doesn't crash when "n+1" is reached.

6.3. Dropped Flows Per Second

This test determines the rate that which flows are dropped once the neighbor cache size is exceeded. The metric for this test is the number of flows which are dropped in a minute.

6.3.1. General Testing Procedure

1. Send packets incrementally to unique valid addresses in the target network, within stale entry time period. The number of unique valid addresses may be as high as the size of the neighbor cache, but may be the number of nodes that would be expected in a deployed network. Continue sending packets to previously cached addresses.
2. Send packets incrementally to unique invalid addresses (addresses without valid node in target network), until the intermediate node crashes, packets are no longer accepted or existing flows to unique valid addresses are dropped.

7. Measurements Explicitly Excluded

These are measurements which aren't recommended because of the itemized reasons below:

7.1. DUT CPU Utilization

This measurement relies on the DUT to provide utilization information, which is subjective.

7.2. Malformed Packets

This benchmarking test is not intended to test DUT behavior in the presence of malformed packets.

8. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

9. Security Considerations

Benchmarking activities as described in this memo are limited to technology characterization using controlled stimuli in a laboratory environment, with dedicated address space and the constraints specified in the sections above.

The benchmarking network topology will be an independent test setup and MUST NOT be connected to devices that may forward the test traffic into a production network, or misroute traffic to the test management network.

Further, benchmarking is performed on a "black-box" basis, relying solely on measurements observable external to the DUT/SUT. Special capabilities SHOULD NOT exist in the DUT/SUT specifically for benchmarking purposes.

Any implications for network security arising from the DUT/SUT SHOULD be identical in the lab and in production networks.

10. Acknowledgements

Helpful comments and suggestions were offered by Al Morton, Joel Jaeggli, Nalini Elkins, Scott Bradner, Ram Krishnan, and Marius Georgescu on the BMWG e-mail list and at BMWG meetings. Precise grammatical corrections and suggestions were offered by Ann Cerveny.

11. References

11.1. Normative References

[RFC0826] Plummer, D., "Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware", STD 37, RFC 826, November 1982.

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

[RFC2544] Bradner, S. and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", RFC 2544, March 1999.

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

[RFC5180] Popoviciu, C., Hamza, A., Van de Velde, G. and D. Dugatkin, "IPv6 Benchmarking Methodology for Network Interconnect Devices", RFC 5180, May 2008.

[RFC6583] Gashinsky, I., Jaeggli, J. and W. Kumari, "Operational Neighbor Discovery Problems", RFC 6583, March 2012.

11.2. Informative References

[RFC7048] Nordmark, E. and I. Gashinsky, "Neighbor Unreachability Detection Is Too Impatient", RFC 7048, January 2014.

Authors' Addresses

Bill Cerveny
Arbor Networks
2727 South State Street
Ann Arbor, MI 48104
USA
EMail: wcerveny@arbor.net

Ron Bonica
Juniper Networks
2251 Corporate Park Drive
Herndon, VA 20170
USA
EMail: rbonica@juniper.net