behave
Internet-Draft
Intended status: Standards Track
Expires: May 3, 2012

Z. Li
G. Chen
China Mobile
Q. Zhao
X. Huang
Y. Ma
Beijing University of Posts and Telecommunications

October 31, 2011

Recommendation for DNS64-based NAT64 Round Robin Load-balancing
draft-li-behave-dns64-load-balancing-02

Abstract

This document compares some stateful prefix selection policies and suggests that the DNS64-based Round Robin Prefix Selection Policy be proper for the NAT64 load-balancing. This document also illustrates two recommended implementations of Round Robin Prefix Selection Policy in detail.

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 May 3, 2012.

Table of Contents

1. Introduction

The problems of bottlenecks and requirements on load balancing in translation technology have been stated in [I-D.li-v6ops-load-balancing-requirement]. Several approaches of NAT64[RFC6146] load-balancing are generally explained in [I-D.zhang-behave-nat64-load-balancing] and the approaches' pros and cons are discussed respectively.

This document generalizes the previous document and recommends that the DNS64[RFC6147] based round robin prefix selection policy be a proper method for the NAT64 load-balancing.

This document also gives the suggested implementations of DNS64 based Round Robin policy in detail.

2. 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 [RFC2119].

3. Terminology

The terminology used in this document is consistent with IPv6 Addressing of IPv4/IPv6 Translators[RFC6052], DNS64[RFC6147], Considerations on NAT64 Load-Balancing[I-D.zhang-behave-nat64-load-balancing]. Besides, the following terminology is defined in this document.

One-dimensional circular data structure (ODCDS): Each data record is stored in one of the elements organized as a circle. All the elements can be circularly and sequentially accessed. It can be implemented using one-dimensional array or circularly linked list.

Two-dimensional circular data structure (TDCDS): Each data record is stored in one of the elements that are organized as a circle (inner circle). Each such circle is one of the elements that are organized in another circle (outer circle). It can be implemented using two-dimensional array or circularly linked list.

Prefix64: An IPv6 prefix used for synthesizing IPv6 addresses representing IPv4 hosts in the IPv6 realm. See IPv6 Addressing of IPv4/IPv6 Translators[RFC6052] for more details on how IPv4-embedded IPv6 addresses are built.

4. Considerations on NAT64 Load-balancing Solutions

As described in [I-D.li-v6ops-load-balancing-requirement], there are four key factors that should be considered when designing the load-balancing solutions.

4.1. Target Device

Since NAT64 is a kind of translation technology, the target device should be the translation gateway - NAT64 translator and ALGs on it. The load-balancer will identify different NAT64 translators by Prefix64s.

4.2. Selection Policy

Compared with Round-Robin Selection Policy, the other policies listed in [I-D.li-v6ops-load-balancing-requirement] and [I-D.zhang-behave-nat64-load-balancing] have obvious shortages when applied on NAT64 load balancing.

- Anycast-Based Selection Policy: The policy can't be used to stateful translation technology such as NAT64.

- Source-Based Selection Policy: The policy's efficiency is constrained by the selection criteria. Besides, for the reason that the user can only access one NAT64, the whole system suffers from less robust capability.

- Destination-Based Selection Policy: Involve human to fairly assign certain IPv4 server to several different NAT64s. Otherwise, the NAT64 in charge of some popular web-site will suffer from much traffic.

- Dynamic Selection Policy: Introduce SNMP or similar techniques to collect the information of NAT64s'status will cause additional communication overheads and management complexity.

4.3. Load-balancer Location

The load-balancer can be implemented in Terminal/Host, Access Gateway or Network Servers. Since Access Gateway is not necessary in NAT64 enviroment and least influence should be made on Terminal/Host, load-balancer should be located in Network Servers.

4.4. Transport Protocol

The candidates are DNS, DHCP, ICMPv6 and RADIUS protocol. Since NAT64 relies on DNS64 which returns the synthesized IPv6 addresses,

it's reasonable to use DNS protocol to transport IPv6 addresses with Prefix64 and steer traffic toward different NAT64 translators.

4.5. Conclusion

This document recommends DNS64-based round robin policy to balance the workload of NAT64 translators.

Prefix64s corresponding to NAT64 translators should be configured in a DNS64 server. On receiving AAAA query from user and no available AAAA record found, the DNS64 server circularly selects Prefix64 to be combined with available A records. All the synthesized AAAA records will be returned to user.

Since Prefix64s are used in round-robin way, users' traffic will routed to different NAT64 translators and the load of translation gateway is well balanced among many different NAT64 translators.

Note: In some cases, NAT64 might decouple with DNS64 to directly communicate with IPv4 servers. For example, IPv6 hosts get destination address through IPv4-literal in the application itself, other than DNS64. These hosts may synthesize NAT64 prefix with such IPv4 referral information. In this particular case, the recommended DNS64-based policy is not applied.

5. Recommended Implementations

Two specific implementations of DNS64-based NAT64 Round Robin load-balancing mechanism are recommended here. They are provided as referential implementations. Because the detailed implementation isn't visible to other devices, developers may use other better data structure, if any.

Solution 1:

Use two kinds of ODCDS which can be implemented by one-dimensional array or circularly linked list. One is used to store all the Prefix64s and the other is used to store all the returned A records. When AAAA query arrives, DNS64 dynamically generates AAAA records from one circularly selected Prefix64 and all the A records and returns them to user.

Solution 2:

Use one kind of TDCDS which can be implemented by two-dimensional array or circularly linked list. It stores all the synthesized AAAA records with the same Prefix64 as one of many inner circles. All the

inner circles are elements of a outer circle. When AAAA query arrives, DNS64 retrieves all the cached AAAA records with the same Prefix64 circularly and returns them to user.

5.1. ODCDS-Based Solution

This solution creates two kinds of ODCDS (O1 and O2) in DNS64 server. O1 stores all the Prefix64s when DNS64 starts up. O2 stores all the IPv4 addresses from returned results when upstream DNS server response to A query. O1's instance is only one but O2's instance can be multiple. Each O2's instance is corresponding to a different FQDN.

5.1.1. Operations in DNS64

When a user requires an AAAA record of a new FQDN and there is no AAAA record, the DNS64 will require A record. Once A records returned, an O2's instance is created. It caches all the returned IPv4 addresses (some IPv4 server's FQDN has more than one IPv4 address). Then, the following steps should be performed.

1. Fetch the Prefix64 in O1's current position and the IPv4 address in O2's current position.

2. Combine the Prefix64 with the IPv4 address as described in DNS64[RFC6147].

3. Move current position in O2 onto the next element. If current position in O2 has gone back to the start, move both current positions in O1 and O2 onto the next element together and then go to step 4. Else, hold the same position in O1 and then go to step 1.

4. Return all the synthetic AAAA records generated in every step 2 to user. These AAAA records have the same Prefix64.

If a user requires an AAAA record of a cached FQDN, start directly at step 1. Then, No O2's instances will be created.

5.1.2. Pros

+ Ensure reasonable distribution among a set of NAT64 devices and a set of business servers;

+ Only store one copy of IPv6 prefix and necessary IPv4 addresses to occupy less memory space.

5.1.3. Cons

- A given IPv6-only hosts may be redirected to distinct NAT64 devices. Therefore, distinct IPv4 address may be used to represent the IPv6-only host in the IPv4 realm [I-D.zhang-behave-nat64-load-balancing];

- A user can not be redirected to the NAT64 device where it has instructed its port forwarding entries [I-D.zhang-behave-nat64-load-balancing];

- Involve certain time cost while synthesizing all the AAAA records with the same Prefix64 every time.

5.2. TDCDS-Based Solution

This solution creates one kind of TDCDS (T1) in DNS64 server. T1 stores all the synthetic IPv6 addresses when upstream DNS server response to A query. T1's instance can be multiple. Each T1's instance is corresponding to a different FQDN.

5.2.1. Operations in DNS64

When a user requires an AAAA record of a new FQDN and there is no AAAA record, DNS64 will require A record. Once A records returned, an T1's instance is created corresponding to this FQDN. Suppose the inner circle stores all the AAAA records with the same Prefix64 and many such circles are stored in the outer circle. Every Prefix64 available is bound with an element or an inner circle in outer circle. Then, the following steps should be performed.

1. Fetch the Prefix64 in current position of outer circle and all the returned IPv4 addresses.

2. Combine the Prefix64 with IPv4 addresses as described in RFC6147 [RFC6147] and store them circularly in the elements of inner circle.

3. Move current position in outer circle onto the next element. If current position in outer circle has gone back to the start, then go to step 4. Else, go to step 1.

4. Retrieve the AAAA record located by current position in outer circle and current position in corresponding inner circle. Move current position in inner circle to the next element.

5. If current position in inner circle has gone back to the start, return all the synthetic AAAA records retrieved at step 4 to user, move current position in outer circle onto the next inner circle and

move the current position in corresponding inner circle onto the element with exactly one element farther than the current position in previous inner circle. Else, hold the same position in the outer circle and go to step 4.

If a user requires an AAAA record of a cached FQDN, start directly at step 4. Then, No T1's instances will be created.

5.2.2. Pros

+ Ensure reasonable distribution among a set of NAT64 devices and a set of business servers;

+ Cache all the AAAA record to improve response speed.

5.2.3. Cons

- A given IPv6-only hosts may be redirected to distinct NAT64 devices. Therefore, distinct IPv4 address may be used to represent the IPv6-only host in the IPv4 realm [draft-zhang-behave-nat64-load-balancing];

- A user can not be redirected to the NAT64 device where it has instructed its port forwarding entries [draft-zhang-behave-nat64-load-balancing];

- Involve certain space cost while caching all the synthesized IPv6 addresses.

6. Security Considerations

No potential security problem is introduced in this document.

7. IANA Considerations

This document makes no request of IANA.

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

8. References

8.1. Normative References

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

8.2. Informative References

[I-D.li-v6ops-load-balancing-requirement]
Li, Z., Zhao, Q., Huang, X., Ma, Y., and K. Building, "Analysis on Load Balancing Requirement in IPv4/IPv6 Transition", draft-li-v6ops-load-balancing-requirement-01 (work in progress), July 2011.

[I-D.zhang-behave-nat64-load-balancing]
Zhang, D., Xu, X., and M. Boucadair, "Considerations on NAT64 Load-Balancing", draft-zhang-behave-nat64-load-balancing-03 (work in progress), July 2011.

[RFC6052] Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", RFC 6052, October 2010.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, April 2011.

[RFC6147] Bagnulo, M., Sullivan, A., Matthews, P., and I. van Beijnum, "DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers", RFC 6147, April 2011.

Authors' Addresses

Zhenqiang Li
China Mobile
Unit2, Dacheng Plaza, No. 28 Xuanwumenxi Ave, Xicheng District
Beijing 100053
P.R. China

Email: lizhenqiang@chinamobile.com

Gang Chen
China Mobile
Unit2, Dacheng Plaza, No. 28 Xuanwumenxi Ave, Xicheng District
Beijing 100053
P.R. China

Email: chengang@chinamobile.com

Qin Zhao
Beijing University of Posts and Telecommunications
Information Network Center, No. 10 Xitucheng Road, Haidian District
Beijing 100876
P.R. China

Email: zhaoqin@bupt.edu.cn

Xiaohong Huang
Beijing University of Posts and Telecommunications
Information Network Center, No. 10 Xitucheng Road, Haidian District
Beijing 100876
P.R. China

Email: huangxh@bupt.edu.cn

Yan Ma
Beijing University of Posts and Telecommunications
Information Network Center, No. 10 Xitucheng Road, Haidian District
Beijing 100876
P.R. China

Email: mayan@bupt.edu.cn