MMUSIC P. Martinsen
Internet-Draft T. Reddy
Intended status: Standards Track P. Patil
Expires: August 13, 2015 Cisco
February 9, 2015

ICE IPv4/IPv6 Dual Stack Fairness
draft-martinsen-mmusic-ice-dualstack-fairness-02

Abstract

This document provides guidelines on how to make Interactive Connectivity Establishment (ICE) conclude faster in multihomed and IPv4/IPv6 dual-stack scenarios where broken paths exist. The provided guidelines are backwards compatible with the original ICE specification.

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 August 13, 2015.

Copyright Notice

Copyright (c) 2015 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. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

Applications should take special care to deprioritize network interfaces known to provide unreliable connectivity. For example certain tunnel services might provide unreliable connectivity. The simple guidelines presented here describes how to deprioritize interfaces known by the application to provide unreliable connectivity. This application knowledge can be based on simple metrics like previous connection success/failure rates or a more static model based on interface types like wired, wireless, cellular, virtual, tunnelled and so on.

There is a also need to introduce more fairness in the handling of connectivity checks for different IP address families in dual-stack IPv4/IPv6 ICE scenarios. Section 4.1.2.1 of ICE [RFC5245] points to [RFC3484] for prioritizing among the different IP families. [RFC3484] is obsoleted by [RFC6724] but following the recommendations from the updated RFC will lead to prioritization of IPv6 over IPv4 for the same candidate type. Due to this, connectivity checks for candidates of the same type (host, reflexive or relay) are sent such that an IP address family is completely depleted before checks from the other address family are started. This results in user noticeable setup delays if the path for the prioritized address family is broken.

To avoid such user noticeable delays when either IPv6 or IPv4 path is broken or excessive slow, this specification encourages intermingling the different address families when connectivity checks are performed. Introducing IP address family fairness into ICE connectivity checks will lead to more sustained dual-stack IPv4/IPv6 deployment as users will no longer have an incentive to disable IPv6.

The cost is a small penalty to the address type that otherwise would have been prioritized.

The guidelines outlined in this specification are backward compatible with a standard ICE implementation. This specification only alters the values used to create the resulting checklists in such a way that the core mechanisms from ICE [RFC5245] are still in effect. The introduced fairness might be better, but not worse than what exists today.

2. Notational Conventions

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].

This document uses terminology defined in [RFC5245].

3. Improving ICE Multihomed Fairness

A multihomed ICE agent can potentially send and receive connectivity checks on all available interfaces. To avoid unnecessary delay when performing connectivity checks it would be beneficial to prioritize interfaces known by the agent to provide connectivity.

Candidates from a interface known to the application to provide unreliable connectivity SHOULD get a low candidate priority. This ensures they appear near the end of the candidate list, and would be the last to be tested during the connectivity check phase. This allows candidate pairs more likely to succeed to be tested first.

If the application is unable to get any interface information regarding type or unable to store any relevant metrics, it SHOULD treat all interfaces as if they have reliable connectivity. This ensures all interfaces gets their fair chance to perform their connectivity checks.

4. Improving ICE Dual Stack Fairness

Candidates SHOULD be prioritized such that a long sequence of candidates belonging to the same address family will be intermingled with candidates from an alternate IP family. For example, promoting IPv4 candidates in the presence of many IPv6 candidates such that an IPv4 address candidate is always present after a small sequence of IPv6 candidates, i.e., reordering candidates such that both IPv6 and IPv4 candidates get a fair chance during the connectivity check phase. This makes ICE connectivity checks more responsive to broken path failures of an address family.

An ICE agent can choose an algorithm or a technique of its choice to ensure that the resulting check lists have a fair intermingled mix of IPv4 and IPv6 address families. However, modifying the check list directly can lead to uncoordinated local and remote check lists that result in ICE taking longer to complete or in the worst case scenario fail. The best approach is to modify the formula for calculating the candidate priority value described in ICE [RFC5245] section 4.1.2.1.

Implementations SHOULD prioritize IPv6 candidates by putting some of them first in the the intermingled checklist. This increases the chance of a IPv6 connectivity checks to complete first and be ready for nomination or usage. This enables implementations to follow the intent of [RFC6555]Happy Eyeballs: Success with Dual-Stack Hosts.

5. Compatibility

ICE [RFC5245] section 4.1.2 states that the formula in section 4.1.2.1 SHOULD be used to calculate the candidate priority. The formula is as follows:

```
priority = (2^24)*(type preference) +
           (2^8)*(local preference) +
           (2^0)*(256 - component ID)
```

ICE [RFC5245] section 4.1.2.2 has guidelines for how the type preference and local preference value should be chosen. Instead of having a static local preference value for IPv4 and IPv6 addresses, it is possible to choose this value dynamically in such a way that IPv4 and IPv6 address candidate priorities ends up intermingled within the same candidate type.

It is also possible to dynamically change the type preference in such a way that IPv4 and IPv6 address candidates end up intermingled regardless of candidate type. This is useful if there are a lot of IPv6 host candidates effectively blocking connectivity checks for IPv4 server reflexive candidates.

The list below shows a sorted local candidate list where the priority is calculated in such a way that the IPv4 and IPv6 candidates are intermingled. To allow for earlier connectivity checks for the IPv4 server reflexive candidates, some of the IPv6 host candidates are demoted. This is just an example of how a candidate priorities can be calculated to provide better fairness between IPv4 and IPv6 candidates without breaking any of the ICE connectivity checks.

	Candidate Type	Address Type	Component ID	Priority
(1)	HOST	IPv6	(1)	2129289471
(2)	HOST	IPv6	(2)	2129289470
(3)	HOST	IPv4	(1)	2129033471
(4)	HOST	IPv4	(2)	2129033470
(5)	HOST	IPv6	(1)	2128777471
(6)	HOST	IPv6	(2)	2128777470
(7)	HOST	IPv4	(1)	2128521471
(8)	HOST	IPv4	(2)	2128521470
(9)	HOST	IPv6	(1)	2127753471
(10)	HOST	IPv6	(2)	2127753470
(11)	SRFLX	IPv6	(1)	1693081855
(12)	SRFLX	IPv6	(2)	1693081854
(13)	SRFLX	IPv4	(1)	1692825855
(14)	SRFLX	IPv4	(2)	1692825854
(15)	HOST	IPv6	(1)	1692057855
(16)	HOST	IPv6	(2)	1692057854
(17)	RELAY	IPv6	(1)	15360255
(18)	RELAY	IPv6	(2)	15360254
(19)	RELAY	IPv4	(1)	15104255
(20)	RELAY	IPv4	(2)	15104254

SRFLX = server reflexive

Note that the list does not alter the component ID part of the formula. This keeps the different components (RTP and RTCP) close in the list. What matters is the ordering of the candidates with component ID 1. Once the checklist is formed for a media stream the candidate pair with component ID 1 will be tested first. If ICE connectivity check is successful then other candidate pairs with the same foundation will be unfrozen ([RFC5245] section 5.7.4. Computing States).

The local and remote agent can have different algorithms for choosing the local preference and type preference values without impacting the synchronization between the local and remote check lists.

The check list is made up by candidate pairs. A candidate pair is two candidates paired up and given a candidate pair priority as described in [RFC5245] section 5.7.2. Using the pair priority formula:

```
pair priority = 2^32*MIN(G,D) + 2*MAX(G,D) + (G>D?1:0)
```

Where G is the candidate priority provided by the controlling agent and D the candidate priority provided by the controlled agent. This ensures that the local and remote check lists are coordinated.

Even if the two agents have different algorithms for choosing the candidate priority value to get an intermingled set of IPv4 and IPv6 candidates, the resulting checklist, that is a list sorted by the pair priority value, will be identical on the two agents.

The agent that has promoted IPv4 cautiously i.e. lower IPv4 candidate priority values compared to the other agent, will influence the check list the most due to (2^32*MIN(G,D)) in the formula.

These recommendations are backward compatible with a standard ICE implementation. The resulting local and remote checklist will still be synchronized. The introduced fairness might be better, but not worse than what exists today

6. Example Algorithm for Choosing the Local Preference

The algorithm described in this section can be used by an implementation to introduce IPv4/IPv6 dual stack and multihomed fairness. Implementations implementing their own algorithm must take care not to break any ICE compatibility. See Section Section 5 for details.

The value space for the local preference is from 0 to 65535 inclusive. This value space can be divided up in chunks for each IP address family.

An IPv6 and IPv4 start priority must be given. In this example IPv6 starts at 60000 and IPv4 at 59000. IPv6 should be given the highest start priority.

Interfaces known to the application to provide unreliable connectivity will be given a low local_preference value. This will place candidates from those interface near the end in a sorted candidate list.

```
       IPv6    IPv4
       Start   Start
65535  60k     59k    58k    57k    56k    55k                     0
+--------+------+------+------+------+------+---------------------+
|        | IPv6 | IPv4 | IPv6 | IPv4 | IPv6 |                     |
|        | (1)  |  (1) |  (2) |  (2) |  (3) |                     |
+--------+------+------+------+------+------+---------------------+
          <- N->
```

The local preference can be calculated by the given formula:

```
local_preference = ((S - N*2*(Cn/Cmax))* Ri) + I
```

S: Address type specific start value (IPv4 or IPv6 Start)

N: Absolute value of IPv6_start-IPv4_start. This ensures a positive number even if IPv4 is the highest priority.

Cn: Number of current candidates of a specific IP address type and candidate type (host, server reflexive or relay).

Cmax: Number of allowed consecutive candidates of the same IP address type.

Ri: Reliable interface. A reliable interface known by the application to provide reliable connectivity should set this value to 1. Interfaces known to provide unreliable connectivity should set this to 0. (Allowed values are 0 and 1)

I: Interface priority. Unreliable interfaces can set this value to get a priority among the unreliable interfaces. Max value is recommended to be N. Reliable interfaces should set this to 0.

Using the values N=abs(60000-59000) and Cmax = 2 yields the following sorted local candidate list with only reliable interfaces:

```
(1)  HOST  IPv6 (1) Priority: 2129289471
(2)  HOST  IPv6 (2) Priority: 2129289470
(3)  HOST  IPv4 (1) Priority: 2129033471
(4)  HOST  IPv4 (2) Priority: 2129033470
(5)  HOST  IPv6 (1) Priority: 2128777471
(6)  HOST  IPv6 (2) Priority: 2128777470
(7)  HOST  IPv4 (1) Priority: 2128521471
(8)  HOST  IPv4 (2) Priority: 2128521470
(9)  HOST  IPv6 (1) Priority: 2128265471
(10) HOST  IPv6 (2) Priority: 2128265470
(11) SRFLX IPv6 (1) Priority: 1693081855
(12) SRFLX IPv6 (2) Priority: 1693081854
(13) SRFLX IPv4 (1) Priority: 1692825855
(14) SRFLX IPv4 (2) Priority: 1692825854
(15) RELAY IPv6 (1) Priority: 15360255
(16) RELAY IPv6 (2) Priority: 15360254
(17) RELAY IPv4 (1) Priority: 15104255
(18) RELAY IPv4 (2) Priority: 15104254
```

The result is an even spread of IPv6 and IPv4 candidates among the different candidate types (host, server reflexive, relay). The local preference value is calculated separately for each candidate type.

7. IANA Considerations

None.

8. Security Considerations

STUN connectivity check using MAC computed during key exchanged in the signaling channel provides message integrity and data origin authentication as described in section 2.5 of [RFC5245] apply to this use.

9. Acknowledgements

Authors would like to thank Dan Wing, Ari Keranen, Bernard Aboba, Martin Thomson, Jonathan Lennox, Balint Menyhart and Simon Perreault for their comments and review.

10. Normative References

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

[RFC3484] Draves, R., "Default Address Selection for Internet Protocol version 6 (IPv6)", RFC 3484, February 2003.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

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

[RFC6724] Thaler, D., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, September 2012.

Authors' Addresses

Paal-Erik Martinsen
Cisco Systems, Inc.
Philip Pedersens Vei 22
Lysaker, Akershus 1325
Norway

Email: palmarti@cisco.com

Tirumaleswar Reddy
Cisco Systems, Inc.
Cessna Business Park, Varthur Hobli
Sarjapur Marathalli Outer Ring Road
Bangalore, Karnataka 560103
India

Email: tireddy@cisco.com

Prashanth Patil
Cisco Systems, Inc.
Bangalore
India

Email: praspati@cisco.com