MMUSIC
Internet-Draft
Intended status: Informational
Expires: March 10, 2016

P. Martinsen
T. Reddy
P. Patil
Cisco

September 7, 2015

ICE Multihomed and IPv4/IPv6 Dual Stack Fairness
draft-ietf-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 March 10, 2016.

Table of Contents

1. Introduction 2
2. Notational Conventions 3
3. Improving ICE Multihomed Fairness 3
4. Improving ICE Dual Stack Fairness 4
5. Compatibility 4
6. IANA Considerations 7
7. Implementation Status 7
7.1. ICE-Dual Starck Fairness Test code 8
7.2. ICE-Dual Starck Fairness Test code 8
8. Security Considerations 8
9. Acknowledgements 9
10. References 9
10.1. Normative References 9
10.2. Informative References 9
Authors' Addresses 10

1. Introduction

Applications should take special care to deprioritize network interfaces known to provide unreliable connectivity when operating in a multihomed environment. For example certain tunnel services might provide unreliable connectivity. Doing so will ensure a more fair distribution of the connectivity checks across available network interfaces on the device. The simple guidelines presented here describes how to deprioritize interfaces known by the application to provide unreliable connectivity.

There is a also a need to introduce more fairness when handling 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.

This document describes how to fairly order the candidates in multihomed and dual-stack environments, thus affecting the sending order of the connectivity checks. If aggressive nomination is in use, this will have an effect on what candidate pair ends up as the active one. Ultimately it should be up to the agent to decide what candidate pair is best suited for transporting media.

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 and IP addresses. It is possible for an interface to have several IP addresses associated with it. To avoid unnecessary delay when performing connectivity checks it would be beneficial to prioritize interfaces and IP addresses known by the agent to provide stable connectivity. If the agent have access to information about the physical network it is connected to (Like SSID in a WiFi Network) this can be used as information regarding how that network interface should be prioritized at this point in time.

The application knowledge regarding the reliability of an interface can also 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, tunneled and so on.

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". It is worth noting that the timing recommendations in [RFC6555] are to excessive for ICE usage.

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 assign lower priorities to IP addresses derived from unreliable interfaces using the local preference value.

It is worth mentioning that [RFC5245] section 4.1.2 say that; "if there are multiple candidates for a particular component for a particular media stream that have the same type, the local preference MUST be unique for each one".

The local type preference can be dynamically changed 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.

Candidates with IP addresses from a unreliable interface SHOULD be ordered at the end of the checklist. Not intermingled as the dual-stack 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 (No multihomed candidates). 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

If aggressive nomination is in use the procedures described in this document might change what candidate pair ends up as the active one.

A test implementation with an example algorithm is available [ICE_dualstack_imp].

6. IANA Considerations

None.

7. Implementation Status

[Note to RFC Editor: Please remove this section and reference to [RFC6982] prior to publication.]

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC6982]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC6982], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of

running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

7.1. ICE-Dual Starck Fairness Test code

Organization: Cisco

Description: Open-Source ICE, TURN and STUN implementation.

Implementation: https://github.com/palerikm/ICE-DualStackFairness

Level of maturity: Code is stable. Tests

Coverage: Follows the recommendations in this document

Licensing: BSD

Implementation experience: Straightforward as there are no compatibility issues.

Contact: Paal-Erik Martinsen palmarti@cisco.com

7.2. ICE-Dual Starck Fairness Test code

Organization: Others

Description: Major ICE implementations, browser based and stand-alone ICE, TURN and STUN implementations.

Implementation: Product specific.

Level of maturity: Code is stable and available in the wild.

Coverage: Implements the recommendations in this document.

Licensing: Some open source, some close source

Implementation experience: Already implemented in some of the implementations. This documents describes what needs to be done to achieve the desired fairness.

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, Ole Troan and Simon Perreault for their comments and review.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3484] Draves, R., "Default Address Selection for Internet Protocol version 6 (IPv6)", RFC 3484, DOI 10.17487/RFC3484, February 2003, <http://www.rfc-editor.org/info/rfc3484>.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, DOI 10.17487/RFC5245, April 2010, <http://www.rfc-editor.org/info/rfc5245>.

[RFC6555] Wing, D. and A. Yourtchenko, "Happy Eyeballs: Success with Dual-Stack Hosts", RFC 6555, DOI 10.17487/RFC6555, April 2012, <http://www.rfc-editor.org/info/rfc6555>.

[RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012, <http://www.rfc-editor.org/info/rfc6724>.

[RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 6982, DOI 10.17487/RFC6982, July 2013, <http://www.rfc-editor.org/info/rfc6982>.

10.2. Informative References

[ICE_dualstack_imp]
Martinsen, P., "ICE DualStack Test Implementation github repo", <https://github.com/palerikm/ICE-DualStackFairness>.

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