            Guidelines for Multihomed and IPv4/IPv6 Dual-Stack
              Interactive Connectivity Establishment (ICE)

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 backward compatible with the original ICE
   specification (see RFC 5245).

Status of This Memo

   This memo documents an Internet Best Current Practice.

   This document is a product of the Internet Engineering Task Force
   (IETF). It represents the consensus of the IETF community. It has
   received public review and has been approved for publication by the
   Internet Engineering Steering Group (IESG). Further information on
   BCPs is available in Section 2 of RFC 7841.

   Information about the current status of this document, any errata,
   and how to provide feedback on it may be obtained at
   https://www.rfc-editor.org/info/rfc8421.

Table of Contents

1. Introduction

   In multihomed and IPv4/IPv6 dual-stack environments, ICE [RFC8445]
   would benefit by a fair distribution of its connectivity checks
   across available interfaces or IP address types. With a fair
   distribution of the connectivity checks, excessive delays are avoided
   if a particular network path is broken or slow. Arguably, it would
   be better to put the interfaces or address types known to the
   application last in the checklist. However, the main motivation by
   ICE is to make no assumptions regarding network topology; hence, a
   fair distribution of the connectivity checks is more appropriate. If
   an application operates in a well-known environment, it can safely
   override the recommendation given in this document.

   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
   describe how to deprioritize interfaces known by the application to
   provide unreliable connectivity.

   There is also a need to introduce better handling of connectivity
   checks for different IP address families in dual-stack IPv4/IPv6 ICE
   scenarios. Following the recommendations from RFC 6724 [RFC6724]
   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 delays with
   setup if the path for the prioritized address family is broken.

To avoid user-noticeable delays when either the IPv6 or IPv4 path is broken or excessively slow, this specification encourages intermingling the different address families when connectivity checks are performed. This 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. Further, this document recommends keeping track of previous known connectivity problems and assigning a lower priority to those addresses. Specific mechanisms and rules for tracking connectivity issues are out of scope for this document.

This document describes what parameters an agent can safely alter to fairly order the checklist candidate pairs in multihomed and dual-stack environments, thus affecting the sending order of the connectivity checks. The actual values of those parameters are an implementation detail. Dependent on the nomination method in use, this might 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 the original ICE implementation. This specification only alters the values used to create the resulting checklists in such a way that the core mechanisms from the original ICE specification [RFC5245] and its replacement [RFC8445] are still in effect.

2. Notational Conventions

   This document uses terminology defined in [RFC8445].

3. ICE Multihomed Recommendations

   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.

   The application knowledge regarding the reliability of an interface can also be based on simple metrics like previous connection success/ failure rates, or it can be a more static model based on interface types like wired, wireless, cellular, virtual, and tunneled in conjunction with other operational metrics. This would require the application to have the right permissions to obtain such operational metrics.

Candidates from an interface known to the application to provide
unreliable connectivity should get a low candidate priority. When to
consider connectivity as unreliable is implementation specific.
Usage of ICE is not limited to Voice over IP (VoIP) applications.
What an application sees as unreliability might be determined by a
mix of how long lived the connection is, how often setup is required,
and other, for now unknown, requirements. This is purely an
optimization to speed up the ICE connectivity check phase.

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

4. ICE Dual-Stack Recommendations

Candidates should be prioritized such that a sequence of candidates
belonging to the same address family will be intermingled with
candidates from an alternate IP family, for example, promote 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., reorder 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 select an algorithm or a technique of its choice to
ensure that the resulting checklists have a fair intermingled mix of
IPv4 and IPv6 address families. However, modifying the checklist
directly can lead to uncoordinated local and remote checklists that
result in ICE taking longer to complete or, in the worst case
scenario, fail. The best approach is to set the appropriate value
for local preference in the formula for calculating the candidate
priority value as described in the "Recommended Formula" section
(Section 5.1.2.1) of [RFC8445].

Implementations should prioritize IPv6 candidates by putting some of
them first in the intermingled checklist. This increases the chance
of IPv6 connectivity checks to complete first and be ready for
nomination or usage. This enables implementations to follow the
intent of "Happy Eyeballs: Success with Dual-Stack Hosts" [RFC8305].
It is worth noting that the timing recommendations in [RFC8305] will
be overruled by how ICE paces out its connectivity checks.

A simple formula to calculate how many IPv6 addresses to put before any IPv4 addresses could look like:

$$Hi = (N\_4 + N\_6) / N\_4$$

Where Hi = Head start before intermingling starts
      N_4 = Number of IPv4 addresses
      N_6 = Number of IPv6 addresses

If a host has two IPv4 addresses and six IPv6 addresses, it will insert an IPv4 address after four IPv6 addresses by choosing the appropriate local preference values when calculating the pair priorities.

5. Compatibility

The formula in Section 5.1.2 of [RFC8445] should be used to calculate the candidate priority. The formula is as follows:

$$priority = (2^{24})*(type\ preference) + (2^8)*(local\ preference) + (2^0)*(256 - component\ ID)$$

"Guidelines for Choosing Type and Local Preferences" (Section 5.1.2.2 of [RFC8445]) 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 end 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 Section 5.1.2.1 of [RFC8445] states that "if there are multiple candidates for a particular component for a particular data 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 an unreliable interface should be ordered at the end of the checklist, i.e., 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
candidate priorities can be calculated to provide better fairness
between IPv4 and IPv6 candidates without breaking any of the ICE
connectivity checks.

```
              Candidate   Address Component
                Type        Type      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 the Real-time
Transport Control Protocol (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 the ICE connectivity check
is successful, then other candidate pairs with the same foundation
will be unfrozen (see "Computing Candidate Pair States" in
Section 6.1.2.6 of [RFC8445]).

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 checklists.

The checklist is made up of candidate pairs. A candidate pair is two
candidates paired up and given a candidate pair priority as described
in Section 6.1.2.3 of [RFC8445]. 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 is the candidate priority provided by the controlled agent.
This ensures that the local and remote checklists 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 checklist the most due to (2^32*MIN(G,D)) in the formula.

These recommendations are backward compatible with the original ICE
implementation. The resulting local and remote checklist will still
be synchronized.

Dependent of the nomination method in use, the procedures described
in this document might change what candidate pair ends up as the
active one.

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

6. IANA Considerations

   This document has no IANA actions.

7. Security Considerations

   The security considerations described in [RFC8445] are valid. It
   changes recommended values and describes how an agent could choose
   those values in a safe way. In Section 3, the agent can prioritize
   the network interface based on previous network knowledge. This can
   potentially be unwanted information leakage towards the remote agent.

8. References

8.1. Normative References

   [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,
              <https://www.rfc-editor.org/info/rfc5245>.

   [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,
              <https://www.rfc-editor.org/info/rfc6724>.

   [RFC8305] Schinazi, D. and T. Pauly, "Happy Eyeballs Version 2:
              Better Connectivity Using Concurrency", RFC 8305,
              DOI 10.17487/RFC8305, December 2017,
              <https://www.rfc-editor.org/info/rfc8305>.

   [RFC8445] Keranen, A., Holmberg, C., and J. Rosenberg, "Interactive
              Connectivity Establishment (ICE): A Protocol for Network
              Address Translator (NAT) Traversal", RFC 8445,
              DOI 10.17487/RFC8445, July 2018,
              <https://www.rfc-editor.org/info/rfc8445>.

8.2. Informative References

   [ICE_dualstack_imp]
              "ICE Happy Eyeball Test Algorithms", commit 45083fb,
              January 2014,
              <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
   McAfee, Inc.
   Embassy Golf Link Business Park
   Bangalore, Karnataka 560071
   India

   Email: TirumaleswarReddy_Konda@McAfee.com

   Prashanth Patil
   Cisco Systems, Inc.
   Bangalore
   India

   Email: praspati@cisco.com