Preventing Premature Interactive Connectivity Establishment (ICE) Failures

draft-holmberg-ice-premature-00

Abstract
...

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 31, 2019.

Copyright Notice

Copyright (c) 2019 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.
1. Introduction

[RFC8445] describes a protocol, Interactive Connectivity Establishment (ICE), for Network Address Translator (NAT) traversal for UDP-based communication.

Typically, when using ICE, endpoints will exchange candidate addresses, form a list of candidate pairs, and then test each candidate pair to see if connectivity can be established. If the test for a given pair fails, it is marked accordingly, and if all pairs have failed, the overall ICE process is considered to have failed.

During the process of connectivity checks, additional candidates may be created as a result of successful inbound checks from the remote peer. Such candidates are referred to as peer-reflexive candidates, and once discovered, will be used to form new candidate pairs which will be tested like any other. However, there is an inherent race condition here; if, before learning about any peer-reflexive candidates, an endpoint runs out of candidate pairs to check, either because it has none, or it considers them all to have failed, it will prematurely declare failure and terminate ICE processing. This race condition can occur in many common situations.

This specification updates [RFC8445], by simply requiring that an endpoint wait a minimum amount of time before declaring ICE failure, even if there are no candidate pairs to check, or if all candidate pairs have failed. This delay provides enough time for the discovery of peer-reflexive candidates, which may eventually lead to ICE processing completing successfully.
2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Relevant Scenarios

As noted above, the core problem this specification attempts to address is the situation where even after local gathering and remote candidate signaling has completed, the ICE agent immediately ends up with no valid pairs and no candidate pairs left to check, resulting in a premature ICE failure. This failure is premature because not enough time has elapsed to allow for discovery of peer-reflexive candidates from inbound connectivity checks; if discovered, these candidates are very likely to result in a valid pair.

In most ICE scenarios, the lengthy timeouts for connectivity checks, typically tens of seconds, will prevent this problem from occurring. However, there are certain specific cases where this problem will frequently occur.

3.1. No Candidates From Peer

It is entirely legal for an ICE agent to provide zero candidates of its own. If the agent somehow knows that the remote endpoint is directly reachable, gathering local candidates is unnecessary and will only cause delays; the remote endpoint can discover the appropriate local candidate via connectivity checks.

However, following the procedures from [RFC8445] strictly will result in immediate ICE failure, since the checklist at the remote endpoint will be empty.

3.2. All Candidates Discarded

Even if the ICE agent provides candidates, they may be discarded by the remote endpoint if it does not know what to do with them. For example, candidates may use an address family that the remote endpoint does not support, (e.g., a host candidate with an IPv6 address in a NAT64 scenario), or may not be usable for some other reason (e.g., a candidate that contains a FQDN that fails to resolve).
In these scenarios, when the candidates are discarded, the checklist at the remote endpoint will once again be empty, leading to immediate ICE failure.

### 3.3. Immediate Candidate Pair Failure

Section 7.2.5.2 of [RFC8445] describes several situations in which a candidate pair will be considered to have failed, well before the connectivity check timeout.

As a result, even if the ICE agent provides usable candidates, the pairs created by the remote endpoint may fail immediately when checked, e.g., a check to a nonroutable address that receives an immediate ICMP error.

In this situation, the checklist at the remote endpoint may contain only failed pairs, resulting in immediate ICE failure.

### 4. Update to RFC 8445

In order to avoid the problem raised by this document, the ICE agent needs to wait enough time to allow peer-reflexive candidates to be discovered. Accordingly, when full ICE implementations begin their ICE processing, as described in [RFC8445], Section 6.1, the ICE agent MUST set a timer, and its duration SHOULD be equal to the ICE agent’s connectivity check timeout.

This timeout value is chosen to roughly coincide with the maximum possible duration of ICE connectivity checks from the remote peer, which, if successful, could create peer-reflexive candidates. Because the ICE agent doesn’t know the exact number of candidate pairs and pacing interval in use by the remote side, this timeout value is simply a guess, albeit an educated one. Regardless, for this particular problem, the desired benefits will be realized as long as the ICE agent waits some reasonable amount of time.

While the timer is running, if a checklist has no pairs left to check, i.e., there are no pairs that are not in the failed state, the ICE agent MUST not conclude that ICE processing has failed, and MUST wait for the timer to elapse before doing so.

One consequence of this behavior is that in cases where ICE should fail, e.g., where both sides provide candidates with unresolvable FQDNs ICE will no longer fail immediately, and only fail when the aforementioned timer expires. However, because most ICE scenarios require an extended period of time to determine failure, the fact that some specific scenarios no longer fail fast should have minimal application impact, if any.
5. Security Considerations

The security considerations for ICE are defined in [RFC8445]. This specification only recommends ICE endpoints to wait for a certain time of period before they declare ICE failure, and does not introduce new security considerations.

6. IANA considerations

This specification makes no requests to IANA.

7. Acknowledgements

8. Normative References


Authors’ Addresses

Christer Holmberg
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: christer.holmberg@ericsson.com

Justin Uberti
Google
747 6th St W
Kirkland 98033
USA

Email: justin@uberti.name