

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 29, 2007

E. Rescorla
Network Resonance
February 25, 2007

Implementing Interactive Connectivity Establishment (ICE) in Lite Mode
draft-rescorla-mmusic-ice-lite-00.txt

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with [Section 6 of BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on August 29, 2007.

Copyright Notice

Copyright (C) The IETF Trust (2007).

Abstract

Interactive Connectivity Establishment (ICE) is a technique for discovering a set of addresses which two peers can use to communicate, even in the face of topological obstacles such as NATs. Because the topologies in which ICE may be used are complex, a full ICE implementation is also fairly complex. Implementation which will only be deployed in settings where they have public addresses (e.g., SIP-PSTN gateways) can, however, be substantially simpler. This document describes a subset of ICE suitable for such implementations.

Internet-Draft

Lite ICE

February 2007

Table of Contents

1.	Introduction	3
2.	Conventions Used In This Document	3
3.	Overview of Operation	3
4.	How to Read This Document	5
5.	Lite ICE Specification	5
5.1.	Gathering Candidates	5
5.2.	Setting Priorities	5
5.3.	Encoding Candidates in SDP	6
5.4.	Receiving SDP Offers/Answers	6
5.5.	Processing Periodic Checks	6
5.6.	Keepalives	7
6.	Security Considerations	7
7.	IANA Considerations	7
8.	References	7
8.1.	Normative References	7
8.2.	Informational References	8
	Author's Address	8
	Intellectual Property and Copyright Statements	9

[1.](#) Introduction

Network Address Translation (NAT) devices are a major obstacle to protocols in which a pair of network elements need to form a direct connection. In many cases, such elements are able to talk to each other directly using a signalling protocol such as SIP [\[4\]](#) but for efficiency reasons want to send data (e.g., media over RTP [\[1\]](#)) directly.

A number of techniques are available for traversing NATs, but entities need a mechanism for discovering which technique will work in its specific environment (and its peer's environment). Internet Connectivity Establishment (ICE) [\[3\]](#) is such a technique.

The basic principle behind ICE is that each entity collects all the addresses on which it might potentially be able to send and receive data. These may include its local address, addresses discovered via STUN [\[5\]](#) or addresses provided by media relays [\[6\]](#). The peers then exchange these candidate addresses and try each potential pairing in priority order until they find one that is satisfactory.

During the design of ICE, many implementors expressed concern about the complexity of the protocol and the difficulty of implementing it. This draft specifies a compatible simplified subset of ICE called "ICE Lite" which is suitable for implementations which will always be operated with public IP addresses. One particular environment where ICE Lite is intended to be useful is in SIP-PSTN gateways which are generally directly connected to the Internet.

[2.](#) Conventions Used In This Document

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 [\[2\]](#).

3. Overview of Operation

A Lite Ice Implementation (LII) behaves much like a normal ICE implementation, with three major differences:

- o It only gathers candidate addresses from its own interfaces.
- o It cannot be a controlling endpoint.
- o It does not generate checks but only responds to periodic checks from other endpoints.

A LII can interoperate with a Full ICE Implementation (FII). there is a subtle point here; ICE is intended to establish bi-directional

connectivity and the LII must have a way to know that its messages are getting through to the other endpoint. Ordinarily this would be done by having both sides perform checks but in order to optimize for simplicity of the LII, the LII does not do so. Rather, we use the fact that the FII receives the LII's STUN response as an implicit check. This requires that the LII receive some message from the peer that its message was received. This is accomplished by having the FII issue two checks. The first check omits the USE-CANDIDATE flag and so the LII will not attempt to use the candidate pair. The second check includes the USE-CANDIDATE flag and so tells the LII that the pair is safe to use.

The interaction of an LII and an FII is shown below.

FII	LII
---	---
SDP Offer ->	\
	<- SDP Answer > First offer/answer
	+ lite flag /
STUN request ->	\ Periodic
	<- STUN response / Check
STUN request ->	\ Candidate
+ USE-CANDIDATE flag	> Selection
	<- STUN response /

As with any ICE implementation, the first thing that happens is that the peers exchange SDP offer and answer. The LII attaches the a=ice-

lite attribute to indicate that it does ICE Lite. This has two implications for the peer:

1. It must be the controlling endpoint.
2. It must not send the USE-CANDIDATE flag the first time it performs any check.

As indicated above, the LII does not perform any checks. Thus, they must all be driven by the FII. The FII follows its usual behavior of creating the check list and starts performing checks. It sends all of its checks without the USE-CANDIDATE flag and only once it has a successful check on a candidate that it was willing to use does it send a second check on that candidate pair with the USE-CANDIDATE flag.

Two LIIs will interoperate but will not do ICE.

[4.](#) How to Read This Document

This document is intended to mostly relieve the implementor of a Lite ICE implementation from the burden of having to read and understand all of RFC XXXX [\[\[Insert ICE RFC # here\]](#) [\[3\]](#). However, it is not intended to be a standalone document. Rather it is intended to be read in conjunction with RFC XXXX. We assume that the reader is roughly familiar with how ICE works and has read at least Sections 1-3 of RFC XXXX.

Section [Section 5](#), contains a description of the responsibilities of a Lite Ice Implementation (LII). Each section follows the same pattern: expository text followed by a pointer to the relevant section of RFC XXXX.

[5.](#) Lite ICE Specification

A LII performs the following tasks:

1. Gathering candidates.
2. Sending an SDP offer/answer

3. Processing the peer's offer/answer
4. Responding to checks from peer endpoints.
5. Determining that candidate pairs are valid and/or favored

[5.1.](#) Gathering Candidates

Like any ICE implementation, a LII gathers candidates. However, unlike full ICE implementations, a LII gathers them only from its locally attached interfaces (host candidates). Other kinds of candidates are not necessary because a LII by definition has a public IP address. A LII may offer only one candidate per component. Note that this process is the same as what a non-ICE implementation does, namely allocating ports from the local interface.

See: Sections [2.1](#) (paragraphs 1,2), 4.1 (paragraphs 1-4), 4.2

[5.2.](#) Setting Priorities

As with full-mode ICE, the candidates must be prioritized, using the algorithm defined in RFC XXX S 4.1.2. However, a LII will only have one candidate type: host. The type preference SHOULD be set to 126.

The endpoint SHOULD set the local preference to 65535.

The component IDs are set as in RFC XXXX. For RTP this means component ID 1 and RTCP component ID 2.

Using these settings, an endpoint which wished to do RTP only would have a single candidate with priority 2130706431 (0x7effffff).

An endpoint which to do both RTP and RTCP would have priorities 2130706431 (0x7effffff) for RTP and 2130706430 (0x7efffffe) for RTCP.

See: [Section 4.1.2](#)

[5.3.](#) Encoding Candidates in SDP

Once the candidates are gathered, a LII must encode them in an SDP offer or answer. Each candidate contains the IP address and port of the candidate and the priority computed in the previous section. There will be one candidate for each component. All candidates MUST be marked as host candidates.

In addition, a LII must set the "a=ice-lite" session-level attribute in order to indicate that it is not a full ICE implementation.

See: Sections [4.3](#)

[5.4.](#) Receiving SDP Offers/Answers

When an LII receives an SDP offer or answer from a peer, it MUST first verify that the peer did not offer the "a=ice-lite" attribute. If it did, ICE processing MUST be terminated.

Although an LII does not maintain a check list, when it receives an offer or answer it needs to extract all the ufrag/upass values from the SDP in order to use them to verify the STUN integrity checks. It also must identify the set of all media streams and components for which ICE must be established.

See: [Section 5.1.](#)

[5.5.](#) Processing Periodic Checks

During ICE discovery, a LII will receive Binding Requests on the bases of some or all of the candidates it included in its most recent offer or answer. When such a Binding Request is received, the LII MUST:

- o Generate a STUN binding response.
- o If the request contains the USE-CANDIDATE flag, create a new candidate pair corresponding to the addresses in the STUN request, add it to the Valid list and mark it "favored".

As with ordinary ICE, the LII must combine its ufrag with the peers

ufrag and use the correct password (from the SDP) to integrity check the STUN request. Once the request has been integrity checked, the LII generates a STUN response containing the transport-level source address in the XOR-MAPPED-ADDRESS field.

Media may be sent on a candidate pair as soon as it is added to the Valid list. Once there is at least one entry on the Valid list for each component of each media stream, ICE processing is finished.

See: Sections [7.2](#), [7.2.2](#), [Section 8](#).

[5.6](#). Keepalives

Like all ICE implementations, all LIIs must send keepalives on active candidate pairs and be prepared to receive keepalives.

See: [Section 11](#).

[6](#). Security Considerations

The security considerations for this document are the same as those for full ICE.

[7](#). IANA Considerations

This document has no actions for IANA.

[8](#). References

[8.1](#). Normative References

- [1] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", [RFC 1889](#), January 1996.
- [2] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [3] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Methodology for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", [draft-ietf-mmusic-ice-13](#) (work in progress), January 2007.

[8.2](#). Informational References

- [4] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", [RFC 3261](#), June 2002.
- [5] Rosenberg, J., Weinberger, J., Huitema, C., and R. Mahy, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)", [RFC 3489](#), March 2003.
- [6] Rosenberg, J., "Obtaining Relay Addresses from Simple Traversal Underneath NAT (STUN)", [draft-ietf-behave-turn-02](#) (work in progress), October 2006.

Author's Address

Eric Rescorla
Network Resonance
2483 E. Bayshore #212
Palo Alto, CA 94303
USA

Email: ekr@networkresonance.com

Full Copyright Statement

Copyright (C) The IETF Trust (2007).

This document is subject to the rights, licenses and restrictions contained in [BCP 78](#), and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in [BCP 78](#) and [BCP 79](#).

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).

