Network Working Group E. Ivov
Internet-Draft Atlassian
Intended status: Standards Track E. Rescorla
Expires: September 29, 2018 RTFM, Inc.
J. Uberti
Google
P. Saint-Andre
Mozilla
March 28, 2018

Trickle ICE: Incremental Provisioning of Candidates for the Interactive Connectivity Establishment (ICE) Protocol

draft-ietf-ice-trickle-18

Abstract

This document describes "Trickle ICE", an extension to the Interactive Connectivity Establishment (ICE) protocol that enables ICE agents to begin connectivity checks while they are still gathering candidates, by incrementally "trickling" candidates over time. This method can considerably accelerate the process of establishing a communication session.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Interactive Connectivity Establishment (ICE) protocol [rfc5245bis] describes how an ICE agent gathers candidates, exchanges candidates with a peer ICE agent, and creates candidate pairs. Once the pairs have been created, the ICE agent will perform connectivity checks, and eventually nominate and select pairs that will be used for sending and receiving data within a communication session.

Following the procedures in [rfc5245bis] can lead to somewhat lengthy session establishment times, because candidate gathering often involves querying STUN servers [RFC5389] and allocating relayed candidates using TURN servers [RFC5766]. Although many ICE procedures can be completed in parallel, the pacing requirements from [rfc5245bis] still need to be followed.

This document defines a supplementary mode of ICE operation, "Trickle ICE", in which candidates can be exchanged incrementally as soon as they become available (and simultaneously with the gathering of other candidates). Connectivity checks can also start as soon as candidate pairs have been created. Because Trickle ICE enables candidate gathering and connectivity checks to be done in parallel, the method can considerably accelerate the process of establishing a communication session.

This document also defines how to discover support for Trickle ICE, how the procedures in [rfc5245bis] are modified or supplemented when

using Trickle ICE, and how a Trickle ICE agent can interoperate with an ICE agent compliant to [rfc5245bis].

This document does not define any protocol-specific usage of Trickle ICE. Instead, protocol-specific details for Trickle ICE are defined in separate usage documents. Examples of such documents are [I-D.ietf-mmusic-trickle-ice-sip] (which defines usage with the Session Initiation Protocol (SIP) [RFC3261] and the Session Description Protocol [RFC3261]) and [XEP-0176] (which defines usage with XMPP [RFC6120]). However, some of the examples in the document use SDP and the offer/answer model [RFC3264] to explain the underlying concepts.

The following diagram illustrates a successful Trickle ICE exchange with a signaling protocol that follows the offer/answer model:

```
             Alice                                            Bob
               |                     Offer                     |
               |---------------------------------------------->|
               |            Additional Candidates              |
               |---------------------------------------------->|
               |                     Answer                    |
               |<----------------------------------------------|
               |            Additional Candidates              |
               |<----------------------------------------------|
               | Additional Candidates and Connectivity Checks |
               |<--------------------------------------------->|
               |<========== CONNECTION ESTABLISHED ===========>|
```

Figure 1: Flow

There is quite a bit of operational experience with the technique behind Trickle ICE, going back as far as 2005 (when the XMPP Jingle extension defined a "dribble mode" as specified in [XEP-0176]); this document incorporates feedback from those who have implemented and deployed the technique over the years.

2. Terminology

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 specification makes use of all terminology defined for Interactive Connectivity Establishment in [rfc5245bis]. In addition, it defines the following terms:

Full Trickle: The typical mode of operation for Trickle ICE agents, in which the initial ICE description can include any number of candidates (even zero candidates) and does not need to include a full generation of candidates as in half trickle.

Generation: All of the candidates conveyed within an ICE session (as defined in [rfc5245bis]).

Half Trickle: A Trickle ICE mode of operation in which the initiator gathers a full generation of candidates strictly before creating and conveying the initial ICE description. Once conveyed, this candidate information can be processed by regular ICE agents, which do not require support for Trickle ICE. It also allows Trickle ICE capable responders to still gather candidates and perform connectivity checks in a non-blocking way, thus providing roughly "half" the advantages of Trickle ICE. The half trickle mechanism is mostly meant for use when the responder's support for Trickle ICE cannot be confirmed prior to conveying the initial ICE description.

ICE Description: Any session-related (as opposed to candidate-related) attributes required to configure an ICE agent. These include but are not limited to the username fragment, password, and other attributes.

Trickled Candidates: Candidates that a Trickle ICE agent conveys after conveying the initial ICE description or responding to the initial ICE description, but within the same ICE session. Trickled candidates can be conveyed in parallel with candidate gathering and connectivity checks.

Trickling: The act of incrementally conveying trickled candidates.

Empty Check List: A check list that initially does not contain any candidate pairs because they will be incrementally added as they are trickled. (This scenario does not arise with a regular ICE agent, because all candidate pairs are known when the agent creates the check list set).

3. Determining Support for Trickle ICE

To fully support Trickle ICE, applications SHOULD incorporate one of the following mechanisms so that implementations can determine whether Trickle ICE is supported:

1. Provide a capabilities discovery method so that agents can verify support of Trickle ICE prior to initiating a session (XMPP's Service Discovery [XEP-0030] is one such mechanism).

2. Make support for Trickle ICE mandatory so that user agents can assume support.

If an application protocol does not provide a method of determining ahead of time whether Trickle ICE is supported, agents can make use of the half trickle procedure described in Section 12.

Prior to conveying the initial ICE description, agents using signaling protocols that support capabilities discovery can attempt to verify whether or not the remote party supports Trickle ICE. If an agent determines that the remote party does not support Trickle ICE, it MUST fall back to using regular ICE or abandon the entire session.

Even if a signaling protocol does not include a capabilities discovery method, a user agent can provide an indication within the ICE description that it supports Trickle ICE by communicating an ICE option of 'trickle'. This token MUST be provided either at the session level or, if at the data stream level, for every data stream (an agent MUST NOT specify Trickle ICE support for some data streams but not others). Note: The encoding of the 'trickle' ICE option, and the message(s) used to carry it to the peer, are protocol specific; for instance, the encoding for the Session Description Protocol (SDP) [RFC4566] is defined in [I-D.ietf-mmusic-trickle-ice-sip].

Dedicated discovery semantics and half trickle are needed only prior to session initiation. After a session is established and Trickle ICE support is confirmed for both parties, either agent can use full trickle for subsequent exchanges.

4. Conveying the Initial ICE Description

An initiator can start gathering candidates as soon as it has an indication that communication is imminent (e.g., a user interface cue or an explicit request to initiate a session). Unlike in regular ICE, in Trickle ICE implementations do not need to gather candidates in a blocking manner. Therefore, unless half trickle is being used, the user experience is improved if the initiator generates and transmits their initial ICE description as early as possible (thus enabling the remote party to start gathering and trickling candidates).

An initiator MAY include any mix of candidates when conveying the initial ICE description. This includes the possibility of conveying

all the candidates the initiator plans to use (as in half trickle), conveying only a publicly-reachable IP address (e.g., a candidate at a data relay that is known to not be behind a firewall), or conveying no candidates at all (in which case the initiator can obtain the responder's initial candidate list sooner and the responder can begin candidate gathering more quickly).

Methods for calculating priorities and foundations, as well as determining redundancy of candidates, work just as in regular ICE [rfc5245bis] (with the exception of pruning of duplicate peer reflexive candidates as described under Section 7.1).

5. Responder Procedures

When a responder receives the initial ICE description, it will first check if the ICE description or initiator indicates support for Trickle ICE as explained in Section 3. If this is not the case, the responder MUST process the initial ICE description according to regular ICE procedures [rfc5245bis] (or, if no ICE support is detected at all, according to relevant processing rules for the underlying signaling protocol, such as offer/answer processing rules [RFC3264]).

If support for Trickle ICE is confirmed, a responder will automatically assume support for regular ICE as well. Specifically, the rules from [rfc5245bis] would imply that ICE itself is not supported if the initial ICE description includes no candidates; however, such a conclusion is not warranted if the responder can confirm that the initiator supports Trickle ICE, in which case fallback to non-ICE processing rules is not necessary.

If the initial ICE description indicates support for Trickle ICE, the responder will determine its role and start gathering and prioritizing candidates; while doing so, it will also respond by conveying its own ICE description, so that both the initiator and the responder can start forming check lists and begin connectivity checks.

5.1. Conveying the Initial Response

A responder can respond to the initial ICE description at any point while gathering candidates. The ICE description in the response MAY contain any set of candidates, including all candidates or no candidates. (The benefit of including no candidates is to convey the ICE description as quickly as possible, so that both parties can consider the overall session to be under active negotiation as soon as possible.)

As noted in Section 3, in application protocols that use SDP the responder's ICE description can indicate support for Trickle ICE by including a token of "trickle" in the ice-options attribute.

6. Initiator Procedures

When processing the initial ICE description from a responder, the initiator follows regular ICE procedures to determine its role, after which it forms check lists (as described in Section 7.1) and begins connectivity checks.

7. Performing Connectivity Checks

For the most part, Trickle ICE agents perform connectivity checks following regular ICE procedures. However, the fact that gathering and communicating candidates is asynchronous in Trickle ICE results in several differences.

7.1. Forming Check Lists and Beginning Connectivity Checks

According to regular ICE procedures [rfc5245bis], in order for candidate pairing to be possible and for duplicate candidates to be pruned, the candidates would need to be provided in the relevant ICE descriptions. By contrast, under Trickle ICE check lists can be empty until candidates are conveyed or received. Therefore Trickle ICE agents handle check list formation and candidate pairing in a slightly different way than regular ICE agents: the agents still form the check lists, but they populate a given check list only after they actually have candidate pairs for that check list. Every check list is initially placed in the Running state, even if the check list is empty. An agent then begins connectivity checks (which includes changing the state of some candidate pairs from Frozen to Waiting) as defined in Section 6.1.2.6 of [rfc5245bis].

With regard to pruning of duplicate candidate pairs, a Trickle ICE agent SHOULD follow a policy of keeping the higher priority candidate unless it is peer reflexive.

7.2. Scheduling Checks

As specified in [rfc5245bis], whenever timer Ta fires, only check lists in the Running state will be picked in order to check the scheduling of connectivity checks for candidate pairs.

Therefore, a Trickle ICE agent MUST keep each check list in the Running state as long as it expects candidate pairs to be incrementally added to the check list. After that, the check list state is set according to the procedures in [rfc5245bis].

7.3. Empty Check Lists

The state of an empty check list is initially set to Running, in accordance with Section 6.1.2.1 of [rfc5245bis].

Whenever timer Ta fires, and an empty check list is picked, no action is performed for the list. Without waiting for timer Ta to expire again, the agent selects the next check list in the Running state, in accordance with Section 6.1.4.2 of [rfc5245bis].

In accordance with the rules defined in Section 8.1.1, when inserting a new candidate pair into an empty check list, the agent sets the pair to a state of Waiting or Frozen as appropriate.

7.4. Setting Check List State to Failed

Section 7.2.5.3.3 of [rfc5245bis] requires that agents update check lists and timer states upon completing a connectivity check transaction. During such an update, regular ICE agents would set the state of a check list to Failed if both of the following two conditions are satisfied:

- o all of the pairs in the check list are either in the Failed state or Succeeded state; and

- o there is not a pair in the valid list for each component of the data stream.

With Trickle ICE, the above situation would often occur when candidate gathering and trickling are still in progress, even though it is quite possible that future checks will succeed. For this reason, Trickle ICE agents add the following conditions to the above list:

- o all candidate gathering has completed and the agent is not expecting to discover any new local candidates; and

- o the remote agent has conveyed an end-of-candidates indication for that check list as described in Section 8.2.

8. Discovering and Conveying Additional Local Candidates

After candidate information has been conveyed, agents will most likely continue discovering new local candidates as STUN, TURN, and other non-host candidate gathering mechanisms begin to yield results. Whenever an agent discovers such a new candidate it will compute its priority, type, foundation, and component ID according to regular ICE procedures.

The new candidate is then checked for redundancy against the existing list of local candidates. If its transport address and base match those of an existing candidate, it will be considered redundant and will be ignored. This would often happen for server reflexive candidates that match the host addresses they were obtained from (e.g., when the latter are public IPv4 addresses). Contrary to regular ICE, Trickle ICE agents will consider the new candidate redundant regardless of its priority.

Next the agent "trickles" the newly discovered candidate(s) to the remote agent. The actual delivery of the new candidates is handled by a signaling protocol such as SIP or XMPP. Trickle ICE imposes no restrictions on the way this is done (e.g., some applications may choose not to trickle updates for server reflexive candidates and instead rely on the discovery of peer reflexive ones).

When candidates are trickled, the signaling protocol MUST deliver each candidate (and any end-of-candidates indication as described in Section 8.2) to the receiving Trickle ICE implementation not more than once and in the same order it was conveyed. If the signaling protocol provides any candidate retransmissions, they need to be hidden from the ICE implementation.

Also, candidate trickling needs to be correlated to a specific ICE session, so that if there is an ICE restart, any delayed updates for a previous session can be recognized as such and ignored by the receiving party. For example, applications that choose to signal candidates via SDP may include a Username Fragment value in the corresponding a=candidate line, such as:

```
a=candidate:1 1 UDP 2130706431 2001:db8::1 5000 typ host ufrag 8hhY
```

Or as another example, WebRTC implementations may include a Username Fragment in the JavaScript objects that represent candidates.

Note: The signaling protocol needs to provide a mechanism for both parties to indicate and agree on the ICE session in force (as identified by the Username Fragment and Password combination) so that they have a consistent view of which candidates are to be paired. This is especially important in the case of ICE restarts (see Section 11).

Once the candidate has been conveyed to the remote party, the agent checks if any remote candidates are currently known for this same stream and component. If not, the new candidate will simply be added to the list of local candidates.

Otherwise, if the agent has already learned of one or more remote candidates for this stream and component, it will begin pairing the new local candidates with them and adding the pairs to the existing check lists according to their priority.

Note: A Trickle ICE agent MUST NOT pair a local candidate until it has been trickled to the remote agent.

8.1. Pairing Newly Learned Candidates and Updating Check Lists

Forming candidate pairs works as described in the ICE specification [rfc5245bis]. However, actually adding the new pair to a check list happens according to the rules described below.

If the check list where the pair is to be added already contains the maximum number of candidate pairs (100 by default as per [rfc5245bis]), the new pair is discarded.

If the new pair's local candidate is server reflexive, the server reflexive candidate MUST be replaced by its base before adding the pair to the list.

Once this is done, the agent examines the check list looking for another pair that would be redundant with the new one. If such a pair exists and the type of its remote candidate is not peer reflexive, the pair with the higher priority is kept and the one with the lower priority is discarded. If, on the other hand, the type of the remote candidate in the pre-existing pair is peer reflexive, the agent MUST replace it with the newly formed pair (regardless of their respective priorities); this is done by setting the priority of the new candidate to the priority of the pre-existing candidate and then re-sorting the check list.

For all other pairs, including those with a server reflexive local candidate that were not found to be redundant, the rules specified in the following section apply.

8.1.1. Inserting a New Pair in a Check List

Consider the following tabular representation of all check lists in an agent (note that initially for one of the foundations, i.e., f5, there are no candidate pairs):

	f1	f2	f3	f4	f5
s1 (Audio.RTP)	F	F	F		
s2 (Audio.RTCP)	F	F	F	F	
s3 (Video.RTP)	F				
s4 (Video.RTCP)	F				

Figure 2: Example of Check List State

Each row in the table represents a component for a given data stream (e.g., s1 and s2 might be the RTP and RTCP components for audio) and thus a single check list in the check list set. Each column represents one foundation. Each cell represents one candidate pair. In the tables shown in this section, "F" stands for "frozen", "W" stands for "waiting", and "S" stands for "succeeded"; in addition, "^^" is used to notate newly-added candidate pairs.

When an agent commences ICE processing, in accordance with Section 6.1.2.6 of [rfc5245bis], for each foundation it will unfreeze the pair with the lowest component ID and, if the component IDs are equal, with the highest priority (this is the topmost candidate pair in every column). This initial state is shown in the following table.

	f1	f2	f3	f4	f5
s1 (Audio.RTP)	W	W	W		
s2 (Audio.RTCP)	F	F	F	W	
s3 (Video.RTP)	F				
s4 (Video.RTCP)	F				

Figure 3: Initial Check List State

Then, as the checks proceed (see Section 7.2.5.4 of [rfc5245bis]), for each pair that enters the Succeeded state (denoted here by "S"),

the agent will unfreeze all pairs for all data streams with the same foundation (e.g., if the pair in column 1, row 1 succeeds then the agent will unfreeze the pair in column 1, rows 2, 3, and 4).

	f1	f2	f3	f4	f5
s1 (Audio.RTP)	S	W	W		
s2 (Audio.RTCP)	W	F	F	W	
s3 (Video.RTP)	W				
s4 (Video.RTCP)	W				

Figure 4: Check List State with Succeeded Candidate Pair

Trickle ICE preserves all of these rules as they apply to "static" check list sets. This implies that if a Trickle ICE agent were to begin connectivity checks with all of its pairs already present, the way that pair states change is indistinguishable from that of a regular ICE agent.

Of course, the major difference with Trickle ICE is that check list sets can be dynamically updated because candidates can arrive after connectivity checks have started. When this happens, an agent sets the state of the newly formed pair as described below.

Rule 1: If the newly formed pair has the lowest component ID and, if the component IDs are equal, the highest priority of any candidate pair for this foundation (i.e., if it is the topmost pair in the column), set the state to Waiting. For example, this would be the case if the newly formed pair were placed in column 5, row 1. This rule is consistent with Section 6.1.2.6 of [rfc5245bis].

	f1	f2	f3	f4	f5
s1 (Audio.RTP)	S	W	W		^W^
s2 (Audio.RTCP)	W	F	F	W	
s3 (Video.RTP)	W				
s4 (Video.RTCP)	W				

Figure 5: Check List State with Newly Formed Pair, Rule 1

Rule 2: If there is at least one pair in the Succeeded state for this foundation, set the state to Waiting. For example, this would be the case if the pair in column 5, row 1 succeeded and the newly formed pair were placed in column 5, row 2. This rule is consistent with Section 7.2.5.3.3 of [rfc5245bis].

	f1	f2	f3	f4	f5
s1 (Audio.RTP)	S	W	W		S
s2 (Audio.RTCP)	W	F	F	W	^W^
s3 (Video.RTP)	W				
s4 (Video.RTCP)	W				

Figure 6: Check List State with Newly Formed Pair, Rule 2

Rule 3: In all other cases, set the state to Frozen. For example, this would be the case if the newly formed pair were placed in column 3, row 3.

	f1	f2	f3	f4	f5
s1 (Audio.RTP)	S	W	W		S
s2 (Audio.RTCP)	W	F	F	W	W
s3 (Video.RTP)	W		^F^		
s4 (Video.RTCP)	W				

Figure 7: Check List State with Newly Formed Pair, Rule 3

8.2. Announcing End of Candidates

Once all candidate gathering is completed or expires for an ICE session associated with a specific data stream, the agent will generate an "end-of-candidates" indication for that session and convey it to the remote agent via the signaling channel. Although the exact form of the indication depends on the application protocol, the indication MUST specify the generation (Username Fragment and Password combination) so that an agent can correlate the end-of-candidates indication with a particular ICE session. The indication can be conveyed in the following ways:

- As part of an initiation request (which would typically be the case with the initial ICE description for half trickle)

- Along with the last candidate an agent can send for a stream

- As a standalone notification (e.g., after STUN Binding requests or TURN Allocate requests to a server time out and the agent is no longer actively gathering candidates)

Conveying an end-of-candidates indication in a timely manner is important in order to avoid ambiguities and speed up the conclusion of ICE processing. In particular:

- A controlled Trickle ICE agent SHOULD convey an end-of-candidates indication after it has completed gathering for a data stream, unless ICE processing terminates before the agent has had a chance to complete gathering.

- A controlling agent MAY conclude ICE processing prior to conveying end-of-candidates indications for all streams. However, it is RECOMMENDED for a controlling agent to convey end-of-candidates

indications whenever possible for the sake of consistency and to keep middleboxes and controlled agents up-to-date on the state of ICE processing.

When conveying an end-of-candidates indication during trickling (rather than as a part of the initial ICE description or a response thereto), it is the responsibility of the using protocol to define methods for associating the indication with one or more specific data streams.

Receiving an end-of-candidates indication enables an agent to update check list states and, in case valid pairs do not exist for every component in every data stream, determine that ICE processing has failed. It also enables an agent to speed up the conclusion of ICE processing when a candidate pair has been validated but it involves the use of lower-preference transports such as TURN. In such situations, an implementation MAY choose to wait and see if higher-priority candidates are received; in this case the end-of-candidates indication provides a notification that such candidates are not forthcoming.

An agent MAY also choose to generate an end-of-candidates indication before candidate gathering has actually completed, if the agent determines that gathering has continued for more than an acceptable period of time. However, an agent MUST NOT convey any more candidates after it has conveyed an end-of-candidates indication.

When performing half trickle, an agent SHOULD convey an end-of-candidates indication together with its initial ICE description unless it is planning to potentially trickle additional candidates (e.g., in case the remote party turns out to support Trickle ICE).

After an agent conveys the end-of-candidates indication, it will update the state of the corresponding check list as explained in Section 7. Past that point, an agent MUST NOT trickle any new candidates within this ICE session. After an agent has received an end-of-candidates indication, it MUST also ignore any newly received candidates for that data stream or data session. Therefore, adding new candidates to the negotiation is possible only through an ICE restart (see Section 11).

This specification does not override regular ICE semantics for concluding ICE processing. Therefore, even if end-of-candidates indications are conveyed, an agent will still need to go through pair nomination. Also, if pairs have been nominated for components and data streams, ICE processing MAY still conclude even if end-of-candidates indications have not been received for all streams. In all cases, an agent MUST NOT trickle any new candidates within an ICE

session after nomination of a candidate pair as described in Section 8.1.1 of [rfc5245bis].

9. Receiving Additional Remote Candidates

At any time during ICE processing, a Trickle ICE agent might receive new candidates from the remote agent. When this happens and no local candidates are currently known for this same stream, the new remote candidates are added to the list of remote candidates.

Otherwise, the new candidates are used for forming candidate pairs with the pool of local candidates and they are added to the local check lists as described in Section 8.1.

Once the remote agent has completed candidate gathering, it will convey an end-of-candidates indication. Upon receiving such an indication, the local agent MUST update check list states as per Section 7. This might lead to some check lists being marked as Failed.

10. Receiving an End-Of-Candidates Indication

When an agent receives an end-of-candidates indication for a specific data stream, it will update the state of the relevant check list as per Section 7. If the check list is still in the Running state after the update, the agent will persist the fact that an end-of-candidates indication has been received and take it into account in future updates to the check list.

11. Subsequent Exchanges

Before conveying an end-of-candidates indication, either agent MAY convey subsequent candidate information at any time allowed by the signaling protocol in use. When this happens, agents will use [rfc5245bis] semantics to determine whether or not the new candidate information require an ICE restart. If an ICE restart occurs, the agents can assume that Trickle ICE is still supported if support was determined previously, and thus can engage in Trickle ICE behavior as they would in an initial exchange of ICE descriptions where support was determined through a capabilities discovery method.

12. Unilateral Use of Trickle ICE (Half Trickle)

In half trickle, the initiator conveys the initial ICE description with a full generation of candidates. This ensures that the ICE description can be processed by a regular ICE responder and is mostly meant for use in cases where support for Trickle ICE cannot be confirmed prior to conveying the initial ICE description. The

initial ICE description indicates support for Trickle ICE, which means the responder can respond with something less than a full generation of candidates and then trickle the rest. The initial ICE description for half trickle would typically contain an end-of-candidates indication, although this is not mandatory because if trickle support is confirmed then the initiator can choose to trickle additional candidates before it conveys an end-of-candidates indication.

The half trickle mechanism can be used in cases where there is no way for an agent to verify in advance whether a remote party supports Trickle ICE. Because the initial ICE description contain a full generation of candidates, it can thus be handled by a regular ICE agent, while still allowing a Trickle ICE agent to use the optimization defined in this specification. This prevents negotiation from failing in the former case while still giving roughly half the Trickle ICE benefits in the latter.

Use of half trickle is only necessary during an initial exchange of ICE descriptions. After both parties have received an ICE description from their peer, they can each reliably determine Trickle ICE support and use it for all subsequent exchanges.

In some instances, using half trickle might bring more than just half the improvement in terms of user experience. This can happen when an agent starts gathering candidates upon user interface cues that the user will soon be initiating an interaction, such as activity on a keypad or the phone going off hook. This would mean that some or all of the candidate gathering could be completed before the agent actually needs to convey the candidate information. Because the responder will be able to trickle candidates, both agents will be able to start connectivity checks and complete ICE processing earlier than with regular ICE and potentially even as early as with full trickle.

However, such anticipation is not always possible. For example, a multipurpose user agent or a WebRTC web page where communication is a non-central feature (e.g., calling a support line in case of a problem with the main features) would not necessarily have a way of distinguishing between call intentions and other user activity. In such cases, using full trickle is most likely to result in an ideal user experience. Even so, using half trickle would be an improvement over regular ICE because it would result in a better experience for responders.

13. Trickle ICE and Peer Reflexive Candidates

Even though Trickle ICE does not explicitly modify the procedures for handling peer-reflexive candidates, use of Trickle ICE can have an impact on how they are processed. With Trickle ICE, it is possible that server reflexive candidates can be discovered as peer reflexive in cases where incoming connectivity checks are received from these candidates before the trickle updates that carry them.

While this would certainly increase the number of cases where ICE processing nominates and selects candidates discovered as peer-reflexive, it does not require any change in processing.

It is also likely that some applications would prefer not to trickle server reflexive candidates to entities that are known to be publicly accessible and where sending a direct STUN binding request is likely to reach the destination faster than the trickle update that travels through the signaling path.

14. Requirements for Signaling Protocols

In order to fully enable the use of Trickle ICE, this specification defines the following requirements for signaling protocols.

- A signaling protocol SHOULD provide a way for parties to advertise and discover support for Trickle ICE before an ICE session begins (see Section 3).

- A signaling protocol MUST provide methods for incrementally conveying (i.e., "trickling") additional candidates after conveying the initial ICE description (see Section 8).

- A signaling protocol MUST deliver each trickled candidate or end-of-candidates indication not more than once and in the same order it was conveyed (see Section 8).

- A signaling protocol MUST provide a mechanism for both parties to indicate and agree on the ICE session in force (see Section 8).

- A signaling protocol MUST provide a way for parties to communicate the end-of-candidates indication, which MUST specify the particular ICE session to which the indication applies (see Section 8.2).

15. Preserving Candidate Order while Trickling

One important aspect of regular ICE is that connectivity checks for a specific foundation and component are attempted simultaneously by both agents, so that any firewalls or NATs fronting the agents would whitelist both endpoints and allow all except for the first ("suicide") packets to go through. This is also important to unfreezing candidates at the right time. While not crucial, preserving this behavior in Trickle ICE is likely to improve ICE performance.

To achieve this, when trickling candidates, agents MUST respect the order of components as reflected by their component IDs; that is, candidates for a given component MUST NOT be conveyed prior to candidates for a component with a lower ID number within the same foundation. In addition, candidates MUST be paired, following the procedures in Section 8.1.1, in the same order they are conveyed.

For example, the following SDP description contains two components (RTP and RTCP) and two foundations (host and server reflexive):

```
v=0
o=jdoe 2890844526 2890842807 IN IP6 2001:db8:a0b:12f0::1
s=
c=IN IP6 2001:db8:a0b:12f0::1
t=0 0
a=ice-pwd:asd88fgpdd777uzjYhagZg
a=ice-ufrag:8hhY
m=audio 5000 RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=candidate:1 1 UDP 2130706431 2001:db8:a0b:12f0::1 5000 typ host
a=candidate:1 2 UDP 2130706431 2001:db8:a0b:12f0::1 5001 typ host
a=candidate:2 1 UDP 1694498815 2001:db8:a0b:12f0::3 5000 typ srflx
    raddr 2001:db8:a0b:12f0::1 rport 8998
a=candidate:2 2 UDP 1694498815 2001:db8:a0b:12f0::3 5001 typ srflx
    raddr 2001:db8:a0b:12f0::1 rport 8998
```

For this candidate information the RTCP host candidate would not be conveyed prior to the RTP host candidate. Similarly the RTP server reflexive candidate would be conveyed together with or prior to the RTCP server reflexive candidate.

16. IANA Considerations

IANA is requested to register the following ICE option in the "ICE Options" sub-registry of the "Interactive Connectivity Establishment (ICE) registry", following the procedures defined in [RFC6336].

ICE Option: trickle

Contact: IESG, iesg@ietf.org

Change control: IESG

Description: An ICE option of "trickle" indicates support for incremental communication of ICE candidates.

Reference: RFC XXXX

17. Security Considerations

This specification inherits most of its semantics from [rfc5245bis] and as a result all security considerations described there apply to Trickle ICE.

If the privacy implications of revealing host addresses on an endpoint device are a concern (see for example the discussion in [I-D.ietf-rtcweb-ip-handling] and in Section 19 of [rfc5245bis]), agents can generate ICE descriptions that contain no candidates and then only trickle candidates that do not reveal host addresses (e.g., relayed candidates).

18. Acknowledgements

The authors would like to thank Bernard Aboba, Flemming Andreasen, Rajmohan Banavi, Taylor Brandstetter, Philipp Hancke, Christer Holmberg, Ari Keranen, Paul Kyzivat, Jonathan Lennox, Enrico Marocco, Pal Martinsen, Nils Ohlmeier, Thomas Stach, Peter Thatcher, Martin Thomson, Dale R. Worley, and Brandon Williams for their reviews and suggestions on improving this document. Thanks also to Ari Keranen and Peter Thatcher in their role as chairs, and Ben Campbell in his role as responsible Area Director.

19. References

19.1. 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, <https://www.rfc-editor.org/info/rfc2119>.

[rfc5245bis] Keranen, A., Holmberg, C., and J. Rosenberg, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal", draft-ietf-ice-rfc5245bis-17 (work in progress), February 2018.

19.2. Informative References

[I-D.ietf-mmusic-trickle-ice-sip] Ivov, E., Stach, T., Marocco, E., and C. Holmberg, "A Session Initiation Protocol (SIP) usage for Trickle ICE", draft-ietf-mmusic-trickle-ice-sip-14 (work in progress), February 2018.

[I-D.ietf-rtcweb-ip-handling] Uberti, J. and G. Shieh, "WebRTC IP Address Handling Requirements", draft-ietf-rtcweb-ip-handling-06 (work in progress), March 2018.

[RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996, <https://www.rfc-editor.org/info/rfc1918>.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <https://www.rfc-editor.org/info/rfc3261>.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, DOI 10.17487/RFC3264, June 2002, <https://www.rfc-editor.org/info/rfc3264>.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, DOI 10.17487/RFC4566, July 2006, <https://www.rfc-editor.org/info/rfc4566>.

[RFC4787] Audet, F., Ed. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, DOI 10.17487/RFC4787, January 2007, <https://www.rfc-editor.org/info/rfc4787>.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, DOI 10.17487/RFC5389, October 2008, <https://www.rfc-editor.org/info/rfc5389>.

[RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, DOI 10.17487/RFC5766, April 2010, <https://www.rfc-editor.org/info/rfc5766>.

[RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, DOI 10.17487/RFC6120, March 2011, <https://www.rfc-editor.org/info/rfc6120>.

[RFC6336] Westerlund, M. and C. Perkins, "IANA Registry for Interactive Connectivity Establishment (ICE) Options", RFC 6336, DOI 10.17487/RFC6336, July 2011, <https://www.rfc-editor.org/info/rfc6336>.

[XEP-0030] Hildebrand, J., Millard, P., Eatmon, R., and P. Saint-Andre, "XEP-0030: Service Discovery", XEP XEP-0030, June 2008.

[XEP-0176] Beda, J., Ludwig, S., Saint-Andre, P., Hildebrand, J., Egan, S., and R. McQueen, "XEP-0176: Jingle ICE-UDP Transport Method", XEP XEP-0176, June 2009.

Appendix A. Interaction with Regular ICE

The ICE protocol was designed to be flexible enough to work in and adapt to as many network environments as possible. Despite that flexibility, ICE as specified in [rfc5245bis] does not by itself support trickle ICE. This section describes how trickling of candidates interacts with ICE.

[rfc5245bis] describes the conditions required to update check lists and timer states while an ICE agent is in the Running state. These conditions are verified upon transaction completion and one of them stipulates that:

> If there is not a pair in the valid list for each component of the data stream, the state of the check list is set to Failed.

This could be a problem and cause ICE processing to fail prematurely in a number of scenarios. Consider the following case:

1. Alice and Bob are both located in different networks with Network Address Translation (NAT). Alice and Bob themselves have different address but both networks use the same private internet block (e.g., the "20-bit block" 172.16/12 specified in [RFC1918]).

2. Alice conveys to Bob the candidate 172.16.0.1 which also happens to correspond to an existing host on Bob's network.

3. Bob creates a check list consisting solely of 172.16.0.1 and starts checks.

4. These checks reach the host at 172.16.0.1 in Bob's network, which responds with an ICMP "port unreachable" error; per [rfc5245bis] Bob marks the transaction as Failed.

At this point the check list only contains Failed candidates and the valid list is empty. This causes the data stream and potentially all ICE processing to fail, even though if Trickle ICE agents could subsequently convey candidates that would cause previously empty check lists to become non-empty.

A similar race condition would occur if the initial ICE description from Alice contain only candidates that can be determined as unreachable from any of the candidates that Bob has gathered (e.g., this would be the case if Bob's candidates only contain IPv4 addresses and the first candidate that he receives from Alice is an IPv6 one).

Another potential problem could arise when a non-trickle ICE implementation initiates an interaction with a Trickle ICE implementation. Consider the following case:

1. Alice's client has a non-Trickle ICE implementation.

2. Bob's client has support for Trickle ICE.

3. Alice and Bob are behind NATs with address-dependent filtering [RFC4787].

4. Bob has two STUN servers but one of them is currently unreachable.

After Bob's agent receives Alice's initial ICE description it would immediately start connectivity checks. It would also start gathering candidates, which would take a long time because of the unreachable STUN server. By the time Bob's answer is ready and conveyed to Alice, Bob's connectivity checks may well have failed: until Alice

gets Bob's answer, she won't be able to start connectivity checks and punch holes in her NAT. The NAT would hence be filtering Bob's checks as originating from an unknown endpoint.

Appendix B. Interaction with ICE Lite

The behavior of ICE lite agents that are capable of Trickle ICE does not require any particular rules other than those already defined in this specification and [rfc5245bis]. This section is hence provided only for informational purposes.

An ICE lite agent would generate candidate information as per [rfc5245bis] and would indicate support for Trickle ICE. Given that the candidate information will contain a full generation of candidates, it would also be accompanied by an end-of-candidates indication.

When performing full trickle, a full ICE implementation could convey the initial ICE description or response thereto with no candidates. After receiving a response that identifies the remote agent as an ICE lite implementation, the initiator can choose to not trickle any additional candidates. The same is also true in the case when the ICE lite agent initiates the interaction and the full ICE agent is the responder. In these cases the connectivity checks would be enough for the ICE lite implementation to discover all potentially useful candidates as peer reflexive. The following example illustrates one such ICE session using SDP syntax:

```
        ICE Lite                                          Bob
         Agent
           |   Offer (a=ice-lite a=ice-options:trickle)    |
           |---------------------------------------------->|
           |                                               |no cand
           |         Answer (a=ice-options:trickle)        |trickling
           |<----------------------------------------------|
           |              Connectivity Checks              |
           |<--------------------------------------------->|
  peer rflx|                                               |
 cand disco|                                               |
           |<========== CONNECTION ESTABLISHED ===========>|
```

Figure 8: Example

In addition to reducing signaling traffic this approach also removes the need to discover STUN bindings or make TURN allocations, which may considerably lighten ICE processing.

Appendix C. Changes from Earlier Versions

Note to the RFC Editor: please remove this section prior to publication as an RFC.

C.1. Changes from draft-ietf-ice-trickle-17

- Simplified the rules for inserting a new pair in a check list.
- Clarified it is not allowed to nominate a candidate pair after a pair has already been nominated (a.k.a. renomination or continuous nomination).
- Removed some text that referenced older versions of rfc5245bis.
- Removed some text that duplicated concepts and procedures specified in rfc5245bis.
- Removed the ill-defined concept of stream order.
- Shortened the introduction.

C.2. Changes from draft-ietf-ice-trickle-16

- Made "ufrag" terminology consistent with 5245bis.
- Applied in-order delivery rule to end-of-candidates indication.

C.3. Changes from draft-ietf-ice-trickle-15

- Adjustments to address AD review feedback.

C.4. Changes from draft-ietf-ice-trickle-14

- Minor modifications to track changes to ICE core.

C.5. Changes from draft-ietf-ice-trickle-13

- Removed independent monitoring of check list "states" of frozen or active, since this is handled by placing a check list in the Running state defined in ICE core.

C.6. Changes from draft-ietf-ice-trickle-12

- Specified that the end-of-candidates indication must include the generation (ufrag/pwd) to enable association with a particular ICE session.

- Further editorial fixes to address WGLC feedback.

C.7. Changes from draft-ietf-ice-trickle-11

- Editorial and terminological fixes to address WGLC feedback.

C.8. Changes from draft-ietf-ice-trickle-10

- Minor editorial fixes.

C.9. Changes from draft-ietf-ice-trickle-09

- Removed immediate unfreeze upon Fail.

- Specified MUST NOT regarding ice-options.

- Changed terminology regarding initial ICE parameters to avoid implementer confusion.

C.10. Changes from draft-ietf-ice-trickle-08

- Reinstated text about in-order processing of messages as a requirement for signaling protocols.

- Added IANA registration template for ICE option.

- Corrected Case 3 rule in Section 8.1.1 to ensure consistency with regular ICE rules.

- Added tabular representations to Section 8.1.1 in order to illustrate the new pair rules.

C.11. Changes from draft-ietf-ice-trickle-07

- Changed "ICE description" to "candidate information" for consistency with 5245bis.

C.12. Changes from draft-ietf-ice-trickle-06

- Addressed editorial feedback from chairs' review.

- Clarified terminology regarding generations.

C.13. Changes from draft-ietf-ice-trickle-05

- Rewrote the text on inserting a new pair into a check list.

C.14. Changes from draft-ietf-ice-trickle-04

- Removed dependency on SDP and offer/answer model.

- Removed mentions of aggressive nomination, since it is deprecated in 5245bis.

- Added section on requirements for signaling protocols.

- Clarified terminology.

- Addressed various WG feedback.

C.15. Changes from draft-ietf-ice-trickle-03

- Provided more detailed description of unfreezing behavior, specifically how to replace pre-existing peer-reflexive candidates with higher-priority ones received via trickling.

C.16. Changes from draft-ietf-ice-trickle-02

- Adjusted unfreezing behavior when there are disparate foundations.

C.17. Changes from draft-ietf-ice-trickle-01

- Changed examples to use IPv6.

C.18. Changes from draft-ietf-ice-trickle-00

- Removed dependency on SDP (which is to be provided in a separate specification).

- Clarified text about the fact that a check list can be empty if no candidates have been sent or received yet.

- Clarified wording about check list states so as not to define new states for "Active" and "Frozen" because those states are not defined for check lists (only for candidate pairs) in ICE core.

- Removed open issues list because it was out of date.

- Completed a thorough copy edit.

C.19. Changes from draft-mmusic-trickle-ice-02

- Addressed feedback from Rajmohan Banavi and Brandon Williams.

- Clarified text about determining support and about how to proceed if it can be determined that the answering agent does not support Trickle ICE.

- Clarified text about check list and timer updates.

- Clarified when it is appropriate to use half trickle or to send no candidates in an offer or answer.

- Updated the list of open issues.

C.20. Changes from draft-ivov-01 and draft-mmusic-00

- Added a requirement to trickle candidates by order of components to avoid deadlocks in the unfreezing algorithm.

- Added an informative note on peer-reflexive candidates explaining that nothing changes for them semantically but they do become a more likely occurrence for Trickle ICE.

- Limit the number of pairs to 100 to comply with 5245.

- Added clarifications on the non-importance of how newly discovered candidates are trickled/sent to the remote party or if this is done at all.

- Added transport expectations for trickled candidates as per Dale Worley's recommendation.

C.21. Changes from draft-ivov-00

- Specified that end-of-candidates is a media level attribute which can of course appear as session level, which is equivalent to having it appear in all m-lines. Also made end-of-candidates optional for cases such as aggressive nomination for controlled agents.

- Added an example for ICE lite and Trickle ICE to illustrate how, when talking to an ICE lite agent doesn't need to send or even discover any candidates.

- Added an example for ICE lite and Trickle ICE to illustrate how, when talking to an ICE lite agent doesn't need to send or even discover any candidates.

- o Added wording that explicitly states ICE lite agents have to be prepared to receive no candidates over signaling and that they should not freak out if this happens. (Closed the corresponding open issue).

- o It is now mandatory to use MID when trickling candidates and using m-line indexes is no longer allowed.

- o Replaced use of 0.0.0.0 to IP6 :: in order to avoid potential issues with RFC2543 SDP libraries that interpret 0.0.0.0 as an on-hold operation. Also changed the port number here from 1 to 9 since it already has a more appropriate meaning. (Port change suggested by Jonathan Lennox).

- o Closed the Open Issue about use about what to do with cands received after end-of-cands. Solution: ignore, do an ICE restart if you want to add something.

- o Added more terminology, including trickling, trickled candidates, half trickle, full trickle,

- o Added a reference to the SIP usage for Trickle ICE as requested at the Boston interim.

C.22. Changes from draft-rescorla-01

- o Brought back explicit use of Offer/Answer. There are no more attempts to try to do this in an O/A independent way. Also removed the use of ICE Descriptions.

- o Added SDP specification for trickled candidates, the trickle option and 0.0.0.0 addresses in m-lines, and end-of-candidates.

- o Support and Discovery. Changed that section to be less abstract. As discussed in IETF85, the draft now says implementations and usages need to either determine support in advance and directly use trickle, or do half trickle. Removed suggestion about use of discovery in SIP or about letting implementing protocols do what they want.

- o Defined Half Trickle. Added a section that says how it works. Mentioned that it only needs to happen in the first o/a (not necessary in updates), and added Jonathan's comment about how it could, in some cases, offer more than half the improvement if you can pre-gather part or all of your candidates before the user actually presses the call button.

- o Added a short section about subsequent offer/answer exchanges.

- Added a short section about interactions with ICE Lite implementations.

- Added two new entries to the open issues section.

C.23. Changes from draft-rescorla-00

- Relaxed requirements about verifying support following a discussion on MMUSIC.

- Introduced ICE descriptions in order to remove ambiguous use of 3264 language and inappropriate references to offers and answers.

- Removed inappropriate assumption of adoption by RTCWEB pointed out by Martin Thomson.

Authors' Addresses

Emil Ivov
Atlassian
303 Colorado Street, #1600
Austin, TX 78701
USA

Phone: +1-512-640-3000
Email: eivov@atlassian.com

Eric Rescorla
RTFM, Inc.
2064 Edgewood Drive
Palo Alto, CA 94303
USA

Phone: +1 650 678 2350
Email: ekr@rtfm.com

Justin Uberti
Google
747 6th St S
Kirkland, WA 98033
USA

Phone: +1 857 288 8888
Email: justin@uberti.name

Peter Saint-Andre
Mozilla
P.O. Box 787
Parker, CO 80134
USA

Phone: +1 720 256 6756
Email: stpeter@mozilla.com
URI: https://www.mozilla.com/