Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 31, 2017

E. Ivov
Atlassian
E. Rescorla
RTFM, Inc.
J. Uberti
Google
P. Saint-Andre
Filament

February 27, 2017

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

draft-ietf-ice-trickle-07

Abstract

This document describes "Trickle ICE", an extension to the Interactive Connectivity Establishment (ICE) protocol that enables ICE agents to send and receive candidates incrementally rather than exchanging complete lists. With such incremental provisioning, ICE agents can begin connectivity checks while they are still gathering candidates and considerably shorten the time necessary for ICE processing to complete.

Table of Contents

1. Introduction

The Interactive Connectivity Establishment (ICE) protocol [rfc5245bis] describes mechanisms for gathering candidates, prioritizing them, choosing default ones, exchanging them with a remote party, pairing them, and ordering them into check lists. Once all of these actions have been completed (and only then), the parties can begin a phase of connectivity checks and eventually select the pair of candidates that will be used in a media session or for a given media stream.

Although the sequence described above has the advantage of being relatively straightforward to implement and debug once deployed, it can also be rather lengthy. Candidate gathering often involves things like querying STUN [RFC5389] servers and allocating relayed candidates at TURN [RFC5766] servers. All of these actions can be delayed for a noticeable amount of time; although they can be run in parallel, they still need to respect the pacing requirements from [rfc5245bis], which is likely to delay them even further. Some or all of these actions also need be completed by the remote agent. Both agents would next perform connectivity checks and only then would they be ready to begin streaming media.

These factors can lead to relatively lengthy session establishment times and thus to a degraded user experience.

This document defines an alternative or supplementary mode of operation for ICE implementations, known as "Trickle ICE", in which candidates can be exchanged incrementally. This enables ICE agents to exchange candidates as soon as an ICE negotiation session has been initiated. Connectivity checks for a media stream can also start as soon as the first candidates for that stream become available.

Trickle ICE can reduce session establishment times in cases where connectivity is confirmed for the first exchanged candidates (e.g., where candidates for one of the agents are directly reachable from

the second agent, such as candidates at a media relay). Even when this is not the case, performing candidate gathering for both agents and connectivity checks in parallel can considerably shorten ICE processing times.

It is worth noting that there is quite a bit of operational experience with the Trickle ICE technique, 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.

In addition to the basics of Trickle ICE, this document also describes how to discover support for Trickle ICE, how regular ICE processing needs to be modified when building and updating check lists, and how Trickle ICE implementations interoperate with agents that only implement regular ICE processing as defined in [rfc5245bis].

This specification does not define the usage of Trickle ICE with any specific signaling protocol (however, see [I-D.ietf-mmusic-trickle-ice-sip] for usage with SIP [RFC3261] and [XEP-0176] for usage with XMPP [RFC6120]). Similarly, it does not define Trickle ICE in terms of the Session Description Protocol (SDP) [RFC4566] or the offer/answer model [RFC3264] because the technique can be and already is used in application protocols that are not tied to SDP or to offer/answer semantics. However, because SDP and the offer/answer model are familiar to most readers of this specification, some examples in this document use those particulars in order to explain the underlying concepts.

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:

Candidate Gatherer: A module used by an ICE agent to obtain local candidates. Candidate gatherers use different mechanisms for discovering local candidates, such as STUN and TURN.

Generation: All of the candidates sent within an ICE negotiation session; these are the candidates that are associated with a local /remote ufrag pair (which will change on ICE restart, if any).

ICE Description: Any session-related (as opposed to candidate-related) attributes required to configure an ICE agent. These include but are not limited to "ice-ufrag", "ice-pwd", and "ice-options".

ICE Negotiation Session: A virtual session involving all of the interactions between ICE agents up until an ICE restart (if any).

Initiator: The ICE agent that starts an ICE negotiation session.

Responder: The ICE agent with which an initiator starts an ICE negotiation session.

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

Trickling: The act of sending trickled candidates.

Half Trickle: A Trickle ICE mode of operation where the initiator gathers a full generation of candidates strictly before creating and sending the initial ICE description. Once sent, that ICE description can be processed by regular ICE agents and does not require support for this specification. It also allows Trickle ICE capable responders to still gather candidates and perform connectivity checks in a non-blocking way, thus roughly providing "half" the advantages of Trickle ICE. The mechanism is mostly meant for use in cases where the remote agent's support for Trickle ICE cannot be confirmed prior to sending an initial ICE description.

Full Trickle: The typical mode of operation for Trickle ICE agents, in which an 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.

3. Determining Support for Trickle ICE

To fully support Trickle ICE, applications SHOULD incorporate one of the following mechanisms to enable implementations to 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 14.

Prior to sending an 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 (e.g., in SDP this would be a token of "trickle" in the ice-options attribute).

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. Sending the Initial ICE Description

An agent 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, agents SHOULD generate and transmit their initial ICE description as early as possible, so that the remote party can start gathering and trickling candidates.

Trickle ICE agents MAY include any mix of candidates in an ICE description. This includes the possibility of sending an ICE description that contains all the candidates that the agent plans to use (as in half trickle mode), sending an ICE description that contains only a publicly-reachable IP address (e.g., a candidate at a media relay that is known to not be behind a firewall), or sending an ICE description with 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 with regular ICE (with the exception of pruning of duplicate peer reflexive candidates as described under Section 5.2).

5. Receiving the Initial ICE Description

When a responder receives an 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 agent MUST process the 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, an agent will automatically assume support for regular ICE as well even if the support verification procedure in [rfc5245bis] indicates otherwise. 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 this case, fallback to [RFC3264] is not necessary.

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

5.1. Sending the Initial Response

An agent can respond to an initial ICE description at any point while gathering candidates. Here again the ICE description MAY contain any set of candidates, including all candidates or no candidates. (The benefit of including no candidates is to send 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.

5.2. Forming Check Lists and Beginning Connectivity Checks

After the initiator and responder exchange ICE descriptions, and as soon as they have obtained local and remote candidates, agents begin forming candidate pairs, computing candidate pair priorities, ordering candidate pairs, pruning duplicate pairs, and creating check lists according to regular ICE procedures [rfc5245bis].

According to those procedures, 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. Under Trickle ICE, check lists can be empty until candidate pairs are sent or received. Therefore Trickle ICE agents handle check lists and candidate pairing in a slightly different way than regular ICE agents: the agents still create the check lists, but they populate the check lists only after they actually have the candidate pairs.

A Trickle ICE agent initially considers all check lists to be frozen. It then inspects the first check list and attempts to unfreeze all candidate pairs it has received so far that belong to the first component on the first media stream (i.e., the first media stream that was reported to the ICE implementation from the using application). If that first component of the first media stream does not contain candidates for one or more of the currently known pair foundations, and if candidate pairs already exist for that foundation in one of the following components or media streams, then the agent unfreezes the first of those candidate pairs.

With regard to pruning of duplicate candidate pairs, a Trickle ICE agent SHOULD follow a policy of "highest priority wins, except for peer reflexive candidates".

6. Receiving the Initial Answer

When processing an 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 5.2) 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 imposes a number of changes as described in the following sections.

7.1. Scheduling Checks

The ICE specification [rfc5245bis], Section 5.8, requires that agents terminate the timer for a triggered check in relation to an active check list once the agent has exhausted all frozen pairs in the check list. This will not work with Trickle ICE, because more pairs will be added to the check list incrementally.

Therefore, a Trickle ICE agent SHOULD NOT terminate the timer until the state of the check list is Completed or Failed as specified herein (see Section 8.2).

7.2. Check List and Timer State Updates

The ICE specification [rfc5245bis], Section 7.1.3.3, 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 media 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 gatherers have completed and the agent is not expecting to discover any new local candidates;

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

Regular ICE requires that agents then update all other check lists, placing one pair from each of them into the Waiting state, effectively unfreezing all remaining check lists. However, under Trickle ICE other check lists might still be empty at that point. Therefore a Trickle ICE agent MUST monitor whether a check list is active or frozen independently of the state of the candidate pairs that the check list contains, and MUST consider a check list to be active when unfreezing the first candidate pair in the check list. When there is no candidate pair in a check list (i.e., when the check list is empty), a Trickle ICE agent MAY consider it to be either active or frozen. An empty frozen check list SHOULD be changed to active if another check list is completely finished (i.e., every pair is either Successful or Failed), or if another checklist has a valid candidate pair for all components.

8. Discovering and Sending Additional Local Candidates

After ICE descriptions have been sent, 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 sends (i.e., 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 send trickle updates for server reflexive candidates and instead rely on the discovery of peer reflexive ones).

When trickle updates are sent, each candidate MUST be delivered to the receiving Trickle ICE implementation not more than once. If there are any candidate retransmissions, they need to be hidden from the ICE implementation.

Also, candidate trickling needs to be correlated to a specific ICE negotiation 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 ufrag 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 ufrag 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 negotiation session in force (as identified by the ufrag) 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 13).

Once the candidate has been sent to the remote party, the agent checks if any remote candidates are currently known for this same

stream. 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.

> Note: So that both agents will have the same view of candidate priorities, it is important to replacing existing pairs with seemingly equivalent higher-priority ones and to always update peer-reflexive candidates if equivalent alternatives are received through signaling.

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 checklists in an agent:

	f1	f2	f3	f4	f5
Audio.RTP	cp	cp	cp		
Audio.RTCP	cp	cp	cp	cp	
Video.RTP	cp				cp
Video.RTCP	cp				cp

Figure 1: Trickle State Updates

Each row in the table represents a component for a given media stream. Each column represents one foundation. Each cell represents one candidate pair.

When an agent commences ICE processing as per [rfc5245bis], it will unfreeze (i.e., place in the Waiting state) the topmost candidate pair in every column. Then, as the checks proceed, for each pair that enters the Succeeded state the agent will unfreeze the pair that is immediately underneath the pair that succeeded (e.g., if the pair in column 1, row 1 succeeds then the agent will unfreeze the pair in column 1, row 2). ICE also specifies that, if all the pairs in a media stream for one foundation are unfrozen (e.g., column 1, rows 1 and 2 representing both components for the audio stream), then all of the candidate pairs in the entire column are unfrozen (e.g., column 1, rows 3 and 4).

Trickle ICE preserves all of these rules. This implies that if, for some reason, a Trickle 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 candidates can arrive after connectivity checks have started. When this happens, an agent sets the state of the newly formed pair as follows:

Waiting: if the newly formed pair is the topmost pair in this column;

Waiting: if the pair immediately above the newly formed pair in this column is in the Succeeded state;

Waiting: if there is at least one pair in this column below the row of the newly formed pair whose state is either Succeeded or Failed.

Frozen: in all other cases.

8.2. Announcing End of Candidates

Once all candidate gathering is completed or expires for a specific media stream, the agent will generate an "end-of-candidates" indication for that stream and send it to the remote agent via the signaling channel. The exact form of the indication depends on the application protocol. The indication can be sent in the following ways:

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

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

- o As a standalone notification (e.g., after STUN Binding requests or TURN Allocate requests to a server time out and the agent has no other active gatherers)

Sending 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:

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

- o A controlling agent MAY conclude ICE processing prior to sending end-of-candidates indications for all streams. However, it is RECOMMENDED for a controlling agent to send 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 sending an end-of-candidates indication during trickling (rather than as a part of an initial ICE description or response), it is the responsibility of the using protocol to define methods for relating the indication to one or more specific media 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 media stream, determine that ICE processing has failed. It also enables agents 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 send any more candidates after it has sent an end-of-candidates indication.

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

After an agent sends the end-of-candidates indication, it will update the state of the corresponding check list as explained in Section 7.2. Past that point, an agent MUST NOT send any new candidates within this ICE negotiation session. After an agent has received an end-of-candidates indication, it MUST also ignore any newly received candidates for that media stream or media session. Therefore, adding new candidates to the negotiation is possible only through an ICE restart (see Section 13).

This specification does not override regular ICE semantics for concluding ICE processing. Therefore, even if end-of-candidates indications are sent, agents will still have to go through pair nomination. Also, if pairs have been nominated for components and media streams, ICE processing MAY still conclude even if end-of-candidates indications have not been received for all streams.

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 send an end-of-candidates indication. Upon receiving such an indication, the local agent MUST update check list states as per Section 7.2. This might lead to some check lists being marked as Failed.

10. Receiving an End-Of-Candidates Notification

When an agent receives an end-of-candidates indication for a specific media stream, it will update the state of the relevant check list as per Section 7.2. If the check list is still in the Active 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. 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.

12. Concluding ICE Processing

This specification does not directly modify the procedures for ending ICE processing described in Section 8 of [rfc5245bis], and Trickle ICE implementations follow the same rules.

13. Subsequent Exchanges

Either agent MAY generate a subsequent ICE description at any time allowed by [RFC3264]. When this happens agents will use [rfc5245bis] semantics to determine whether or not the new ICE description requires an ICE restart. If an ICE restart occurs, the user 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.

14. Unilateral Use of Trickle ICE (Half Trickle)

In half trickle mode, the initiator sends a regular 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 sending an 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. A half trickle ICE description 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 sends 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 contains 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 (hence the name of the mechanism).

Use of half trickle is only necessary during an initial exchange of ICE descriptions. After both parties have received a session 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 send the ICE description. 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.

15. 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 negotiation session begins (see Section 3).

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

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

- A signaling protocol MUST provide a way for parties to communicate the end-of-candidates indication (see Section 8.2).

16. Example Flow

As an example, a typical successful Trickle ICE exchange with a signaling protocol that follows the offer/answer model would look this way:

```
           Alice                                            Bob
             |                     Offer                     |
             |---------------------------------------------->|
             |            Additional Candidates              |
             |---------------------------------------------->|
             |                                               |
             |                     Answer                    |
             |<----------------------------------------------|
             |            Additional Candidates              |
             |<----------------------------------------------|
             |                                               |
             | Additional Candidates and Connectivity Checks |
             |<--------------------------------------------->|
             |                                               |
             |<=============== MEDIA FLOWS =================>|
```

Figure 2: Example

17. IANA Considerations

This specification requests no actions from IANA.

18. 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, agents can generate an ICE description that contains no candidates and then only trickle candidates that do not reveal host addresses (e.g., relayed candidates).

19. Acknowledgements

The authors would like to thank Bernard Aboba, Flemming Andreasen, Rajmohan Banavi, Taylor Brandstetter, Christer Holmberg, Jonathan Lennox, Enrico Marocco, Pal Martinsen, Martin Thomson, Dale R. Worley, and Brandon Williams for their reviews and suggestions on improving this document.

20. References

20.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[rfc5245bis] Keranen, A., Keranen, A., and J. Rosenberg, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal", draft-ietf-ice-rfc5245bis-08 (work in progress), December 2016.

20.2. Informative References

[I-D.ietf-mmusic-trickle-ice-sip] Ivov, E., Thomas, T., Marocco, E., and C. Holmberg, "A Session Initiation Protocol (SIP) usage for Trickle ICE", draft-ietf-mmusic-trickle-ice-sip-06 (work in progress), October 2016.

[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, <http://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, June 2002.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

[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, <http://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, <http://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, April 2010.

[RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, March 2011.

[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 media 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 [RFC1918] block.

2. Alice sends Bob the candidate 2001:db8:a0b:12f0::10 which also happens to correspond to an existing host on Bob's network.

3. Bob creates a check list consisting solely of 2001:db8:a0b:12f0::10 and starts checks.

4. These checks reach the host at 2001:db8:a0b:12f0::10 in Bob's network, which responds with an ICMP "port unreachable" error and 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 media stream and potentially all ICE processing to fail.

A similar race condition would occur if the initial ICE description from Alice only contains 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 sent 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 an ICE description as per [rfc5245bis] and would indicate support for Trickle ICE. Given that

the ICE description 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 send an initial ICE description or response with no candidates. After receiving a response that identifies the remote agent as an ICE lite implementation, the initiator can choose to not send 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|                                               |
           |                                               |
           |<=============== MEDIA FLOWS =================>|
```

Figure 3: 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. 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 in which the components and streams as they have been negotiated appear (implicitly or explicitly) in the relevant ICE descriptions. Therefore a candidate for a specific component MUST NOT be sent prior to candidates for other components within the same foundation.

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 IP4 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 description the RTCP host candidate MUST NOT be sent prior to the RTP host candidate. Similarly the RTP server reflexive candidate MUST be sent together with or prior to the RTCP server reflexive candidate.

Similar considerations apply at the level of media streams in addition to foundations; this is covered by the requirement to always start unfreezing candidates starting from the first media stream as described under Section 5.2.

Appendix D. Changes from Earlier Versions

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

D.1. Changes from draft-ietf-ice-trickle-04

o 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.

D.2. Changes from draft-ietf-ice-trickle-03

- Copy edit.

D.3. 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.

D.4. Changes from draft-ietf-ice-trickle-02

- Adjusted unfreezing behavior when there are disparate foundations.

D.5. Changes from draft-ietf-ice-trickle-01

- Changed examples to use IPv6.

D.6. 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.

D.7. Changes from draft-mmusic-trickle-ice-02

- Addressed feedback from Rajmohan Banavi and Brandon Williams.

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

o Clarified text about check list and timer updates.

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

o Updated the list of open issues.

D.8. Changes from draft-ivov-01 and draft-mmusic-00

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

o 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.

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

o 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.

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

D.9. Changes from draft-ivov-00

o 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.

o 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 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).

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

- 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).

- 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.

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

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

D.10. Changes from draft-rescorla-01

- 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.

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

- 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.

- 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.

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

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

- o Added two new entries to the open issues section.

D.11. Changes from draft-rescorla-00

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

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

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

Authors' Addresses

Emil Ivov
Atlassian
303 Colorado Street, #1600
Austin 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
Filament
P.O. Box 787
Parker, CO 80134
USA

Phone: +1 720 256 6756
Email: peter@filament.com
URI: https://filament.com/