Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2016

E. Ivov
Jitsi
E. Rescorla
RTFM, Inc.
J. Uberti
Google
P. Saint-Andre
&yet

October 19, 2015

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

draft-ietf-ice-trickle-00

Abstract

This document describes an extension to the Interactive Connectivity Establishment (ICE) protocol that allows 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. This mechanism is called "trickle ICE".

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Interactive Connectivity Establishment (ICE) protocol [RFC5245] describes mechanisms for gathering candidates, prioritizing them, choosing default ones, exchanging them with the remote party, pairing them and ordering them into check lists. Once all of the above have been completed, and only then, the participating agents can begin a phase of connectivity checks and eventually select the pair of candidates that will be used in the following session.

While the above sequence has the advantage of being relatively straightforward to implement and debug once deployed, it may also prove to be rather lengthy. Gathering candidates or candidate gathering often involves things like querying STUN [RFC5389] servers, discovering UPnP devices, and allocating relayed candidates at TURN [RFC5766] servers. All of these can be delayed for a noticeable amount of time and while they can be run in parallel, they still need to respect the pacing requirements from [RFC5245], which is likely to delay them even further. Some or all of the above would also have to be completed by the remote agent. Both agents would next perform connectivity checks and only then would they be ready to begin streaming media.

All of the above can lead to relatively lengthy session establishment times and degraded user experience.

The purpose of this document is to define an alternative mode of operation for ICE implementations, also known as "trickle ICE", where candidates can be exchanged incrementally. This would allow ICE agents to exchange candidates as soon as a session has been initiated. Connectivity checks for a media stream would also start as soon as the first candidates for that stream have become available.

Trickle ICE allows reducing session establishment times in cases where connectivity is confirmed for the first exchanged candidates (e.g. where the host candidates for one of the agents are directly reachable from the second agent). Even when this is not the case, running candidate gathering for both agents and connectivity checks all in parallel allows to considerably reduce ICE processing times.

It is worth pointing out that before being introduced to the IETF, trickle ICE had already been included in specifications such as XMPP Jingle [XEP-0176] and it has been in use in various implementations and deployments.

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 [RFC5245] processing.

This specification does not define usage of trickle ICE with any specific signalling protocol, different from [RFC5245] which contains a usage for ICE with SIP [RFC3261]. Such usages would have to be specified in separate documents such as for example [I-D.ietf-mmusic-trickle-ice-sip]. However, trickle ICE does however reuse and build upon the SDP syntax defined by [RFC5245].

Although this document mostly describes trickle ICE in terms of the offer/answer model [RFC3264], trickle ICE (and ICE itself) can be used by application protocols that do not follow the offer/answer model.

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 by the protocol for Interactive Connectivity Establishment in [RFC5245].

Vanilla ICE: The Interactive Connectivity Establishment protocol as defined in [RFC5245]. Through the rest of the text, the terms vanilla ICE and "RFC5245" are used interchangeably.

Candidate Harvester: A module used by an ICE agent to obtain local candidates. Candidate gatherers use different mechanisms for discovering local candidates. Some of them would typically make use of protocols such as STUN or TURN. Others may also employ techniques that are not referenced within [RFC5245]. UPnP based

port allocation and XMPP Jingle Relay Nodes [XEP-0278] are among the possible examples.

Trickled Candidates: Candidates that a trickle ICE agent is sending subsequently to but within the context defined by an offer or an answer. Trickled candidates can be sent in parallel with candidate gathering and connectivity checks.

Trickling/Trickle (v.): The act of sending trickled candidates.

Half Trickle: A trickle ICE mode of operation where the offerer gathers its first generation of candidates strictly before creating and sending the offer. Once sent, that offer can be processed by vanilla ICE agents and does not require support for this specification. It also allows trickle ICE capable answerers to still gather candidates and perform connectivity checks in a non-blocking way, thus roughly offering "half" the advantages of trickle ICE. The mechanism is mostly meant for use in cases where support for trickle ICE cannot be confirmed prior to sending a initial offer.

Full Trickle: Regular mode of operation for trickle ICE agents, used in opposition to the half trickle mode of operation.

3. Determining Support for Trickle ICE

According to [RFC5245], supported features are to be advertised in the ice-options attribute. Therefore an agent supporting trickle ICE MUST include a token of "trickle" in the ice-options attribute every time it generates an offer or an answer. Syntax for this token is defined in Section 4.1.

Agents that receive offers or answers can verify support by examining them for the "trickle" ice-options token. However, agents that are about to send an initial offer have no way of doing this. Thus usages of trickle for specific protocols need to either:

- o Provide a way for agents to verify support of trickle ICE prior to initiating a session (XMPP's Service Discovery [XEP-0030] is an example of one such mechanism); or

- o Make support for trickle ICE mandatory so that support could be assumed the agents.

Alternately, for cases where a protocol provides neither of the above, agents may either rely on provisioning/configuration, or use the half trickle procedure described in Section 15.

Prior to sending an initial offer, agents using signaling protocols that support capabilities discovery MAY 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 vanilla ICE or abandon the entire session.

All trickle ICE offers and answers MUST indicate support of this specification, as explained in Section 4.1.

Note that out-of-band discovery semantics and half trickle are only necessary prior to session initiation, or in other words, when sending the initial offer. Once a session is established and trickle ICE support is confirmed for both parties, either agent can use full trickle for subsequent offers.

4. Sending the Initial Offer

An agent starts 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). Contrary to vanilla ICE, implementations of trickle ICE do not need to gather candidates in a blocking manner. Therefore, unless half trickle is being used, agents SHOULD generate and transmit their initial offer as early as possible, in order to allow the remote party to start gathering and trickling candidates.

Trickle ICE agents MAY include any set of candidates in an offer. This includes the possibility of generating one with no candidates, or one that contains all the candidates that the agent is planning on using in the following session.

For optimal performance, it is RECOMMENDED that the candidates in an initial offer (if any) be host candidates only. This would allow both agents to start gathering server reflexive, relayed and other non-host candidates simultaneously, and it would also enable them to begin connectivity checks.

If the privacy implications of revealing host addresses are a concern, agents MAY generate an offer that contains no candidates and then only trickle candidates that do not reveal host addresses (e.g. relayed candidates).

Methods for calculating priorities and foundations, as well as determining redundancy of candidates, work just as with vanilla ICE.

4.1. Encoding the SDP

The process of encoding the SDP [RFC4566] is mostly the same as the one used by vanilla ICE. Still, trickle ICE does require a few differences described here.

Agents MUST indicate support for Trickle ICE by including the "trickle" token for the "a=ice-options" attribute:

```
    a=ice-options:trickle
```

As mentioned earlier in this section, offers and answers can contain any set of candidates, which means that a trickle ICE session description MAY contain no candidates at all. Doing so enables the offerer to receive the answerer's initial candidate list sooner, and also enables the answerer to begin candidate gathering more quickly. In such cases the agent would still need to place an address in the "c=" line(s). If the use of a host address there is undesirable (e.g., for privacy reasons), the agent MAY set the connection address to 0.0.0.0. In this case it MUST also set the port number to 9 (Discard). There is no need to include a fictitious candidate for the 0.0.0.0 address when doing so.

It is worth noting that the use of IP6 :: has been selected over IP4 0.0.0.0, even though [RFC3264] already gives the latter semantics appropriate for such use. The reason for this choice is the historic use of 0.0.0.0 as a means of putting a stream on hold [RFC2543] and the ambiguity that this may cause with legacy libraries and applications.

It is also worth mentioning that use of IP6 :: here does not constitute any kind of indication as to the actual use of IPv6 candidates in a session and it can very well appear in a negotiation that only involves IPv4 candidates.

5. Receiving the Initial Offer

When an agent receives an initial offer, it will first check if it indicates support for trickle ICE as explained in Section 3. If this is not the case, the agent MUST process the offer according to the [RFC5245] procedures or standard [RFC3264] processing in case no ICE support is detected at all.

It is worth pointing out that in case support for trickle ICE is confirmed, an agent will automatically assume support for vanilla ICE as well even if the support verification procedure in [RFC5245]

indicates otherwise. Specifically, the rules from RFC 5245 would imply that ICE itself is not supported if the initial offer includes no candidates in the offer; however, such a conclusion is not warranted if the answerer can confirm that the offerer supports trickle ICE. In this case, the IP6 :: address present in the "c=" line would not "appear in a candidate attribute". Fallback to [RFC3264] is not necessary in this scenario.

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

5.1. Sending the Initial Answer

An agent can respond to an initial offer at any point while gathering candidates. The answer can again contain any set of candidates, including all candidates or no candidates. (The benefit of including no candidates is to send the answer as quickly as possible, so that both parties can consider the overall session to be under active negotiation as soon as possible.) Unless the answering agent is protecting host addresses for privacy reasons, it would typically construct this initial answer including only them, thus allowing the remote party to also start forming checklists and performing connectivity checks.

The answer MUST indicate support for trickle ICE as described by Section 3.

5.2. Forming Check Lists and Beginning Connectivity Checks

After exchanging offer and answer, and as soon as they have obtained local and remote candidates, agents will begin forming candidate pairs, computing their priorities and creating check lists according to the vanilla ICE procedures described in [RFC5245]. Obviously in order for candidate pairing to be possible, candidates would need to be provided in both the offer and the answer. If not, then the agents will still create the check lists (so that their Active/Frozen state could be monitored and updated) but they will only populate the check lists once they actually have the candidate pairs.

Initially, all check lists will have their Active/Frozen state set to Frozen.

Trickle ICE agents will then inspect the first check list and attempt to unfreeze all candidates belonging 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 this checklist is still empty however, agents will hold off further processing until this is no longer the case.

Respecting the order in which lists have been reported to an ICE implementation, or in other words, the order in which they appear in SDP, is crucial to the frozen candidates algorithm and important when making sure that connectivity checks are performed simultaneously by both agents.

5.3. Encoding the SDP

The process for encoding the SDP at the answerer is identical to the process followed by the offerer for both full and lite implementations, as described in Section 4.1.

6. Receiving the Initial Answer

When receiving an answer, agents will follow vanilla ICE procedures to determine their role and they would then form check lists (as described in Section 5.2) and begin connectivity checks .

7. Performing Connectivity Checks

For the most part, trickle ICE agents perform connectivity checks following vanilla ICE procedures. Of course, the asynchronous nature of gathering and communicating candidates in trickle ICE would impose a number of changes described here.

7.1. Scheduling Checks

The ICE specification [RFC5245], 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 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 8.3).

7.2. Check List and Timer State Updates

The ICE specification [RFC5245], Section 7.1.3.3, requires that agents update check lists and timer states upon completing a connectivity check transaction. During such an update vanilla 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 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 8.3.

Vanilla ICE requires that agents then update all other check lists, placing one pair in each of them into the Waiting state, effectively unfreezing all remaining check lists. Given that with trickle ICE, other check lists may still be empty at that point, a trickle ICE agent SHOULD also maintain an explicit Active/Frozen state for every check list, rather than deducing it from the state of the pairs it contains. This state should be set to Active when unfreezing the first pair in a list or when that couldn't happen because a list was empty.

8. Discovering and Sending Additional Local Candidates

After an offer or an answer 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 normal vanilla 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 vanilla ICE, trickle ICE agents will consider the new candidate redundant regardless of its priority.

Next the client sends (i.e. trickles) the newly learnt candidate(s) to the remote agent. The actual delivery of the new candidates will

be specified by using protocols such as SIP. Trickle ICE imposes no restrictions on the way this is done or whether it is done at all. For example, some applications may choose not to send trickle updates for server reflexive candidates and rely on the discovery of peer reflexive ones instead.

When trickle updates are sent however, each candidate MUST be delivered to the receiving Trickle ICE implementation not more than once and in the same order that they were sent. In other words, if there are any candidate retransmissions, they must 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.

One important aspect of Vanilla ICE is that connectivity checks for a specific foundation and component be 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 crucial to unfreezing candidates in the right time.

In order to preserve this feature here, when trickling candidates agents MUST respect the order of the components as they appear (implicitly or explicitly) in the Offer/Answer 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 session description contains two components (RTP and RTCP), and two foundations (host and the server reflexive):

```
v=0
o=jdoe 2890844526 2890842807 IN IP4 10.0.1.1
s=
c=IN IP4 10.0.1.1
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 10.0.1.1 5000 typ host
a=candidate:1 2 UDP 2130706431 10.0.1.1 5001 typ host
a=candidate:2 1 UDP 1694498815 192.0.2.3 5000 typ srflx
    raddr 10.0.1.1 rport 8998
a=candidate:2 2 UDP 1694498815 192.0.2.3 5001 typ srflx
    raddr 10.0.1.1 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.

Note that the order restriction only applies among candidates that belong to the same foundation.

It is also equally important to preserve this order across media streams and this is covered by the requirement to always start unfreezing candidates starting from the first media stream Section 5.2.

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 this is not the case 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.

8.1. Pairing Newly Learned Candidates and Updating Check Lists

Forming candidate pairs will work the way it is described by the ICE specification [RFC5245]. Actually adding the new pair to a check list however, will happen 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 [RFC5245]), 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, the newly formed pair is ignored.

For all other pairs, including those with a server reflexive local candidate that were not found to be redundant:

- if this check list is Frozen then the new pair will also be assigned a Frozen state.

- else if the check list is Active and it is either empty or contains only candidates in the Succeeded and Failed states, then the new pair's state is set to Waiting.

- else if the check list is non-empty and Active, then the new pair state will be set to

  Frozen: if there is at least one pair in the list whose foundation matches the one in the new pair and whose state is neither Succeeded nor Failed (eventually the new pair will get unfrozen after the the on-going check for the existing pair concludes);

  Waiting: if the list contains no pairs with the same foundation as the new one, or, in case such pairs exist but they are all in either the Succeeded or Failed states.

8.2. Encoding the SDP for Additional Candidates

To facilitate interoperability an ICE agent will encode additional candidates using the vanilla ICE SDP syntax. For example:

```
a=candidate:2 1 UDP 1658497328 198.51.100.33 5000 typ host
```

Given that such lines do not provide a relationship between the candidate and the m line that it relates to, signalling protocols using trickle ICE MUST establish that relation themselves using an MID [RFC3388]. Such MIDs use "media stream identification", as defined in [RFC3388], to identify a corresponding m-line. When creating candidate lines usages of trickle ICE MUST use the MID if

possible, or the m-line index if not. Obviously, agents MUST NOT send individual candidates prior to generating the corresponding SDP session description.

The exact means of transporting additional candidates to a remote agent is left to the protocols using trickle ICE. It is important to note, however, that these candidate exchanges are not part of the offer/answer model.

8.3. Announcing End of Candidates

Once all candidate gatherers for a specific media stream complete, or expire, the agents will generate an "end-of-candidates" indication for that stream and send it to the remote agent via the signalling channel. Such indications are sent in the form of a media-level attribute that has the following form: end-of-candidates.

```
            a=end-of-candidates
```

The end-of-candidates indications can be sent in the following ways:

- As part of an offer (which would typically be the case with half trickle initial offers)
- 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 timeout and the agent has no other active gatherers)

Controlled trickle ICE agents SHOULD always send end-of-candidates indications once gathering for a media stream has completed unless ICE processing terminates before they've had a chance to do so. Sending the indication is necessary in order to avoid ambiguities and speed up ICE conclusion. Controlling agents on the other hand MAY sometimes conclude ICE processing prior to sending end-of-candidates notifications for all streams. This would typically be the case with aggressive nomination. Yet it is RECOMMENDED that controlling agents do send such indications whenever possible for the sake of consistency and keeping middle boxes and controlled agents up-to-date on the state of ICE processing.

When sending end-of-candidates during trickling, rather than as a part of an offer or an answer, it is the responsibility of the using protocol to define means that can be used to relate the indication to one or more specific m-lines.

Receiving an end-of-candidates notification allows 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 allows agents to speed ICE conclusion in cases where a candidate pair has been validates but it involves the use of lower-preference transports such as TURN. In such situations some implementations may choose to wait in case higher-priority candidates are received and end-of-candidates provides an indication that this is not going to happen.

An agent MAY also choose to generate an end-of-candidates event 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 send an end-of-candidates notification.

When performing half trickle agents SHOULD send end-of-candidates together with their initial offer unless they are planning on potentially sending additional candidates in case the remote party turns out to actually support trickle ICE.

When end-of-candidates is sent as part of an offer or an answer it can appear as a session-level attribute, which would be equivalent to having it appear in all m-lines.

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

This specification does not override vanilla 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 will still conclude even if end-of-candidate indications have not been received for all streams.

9. Receiving Additional Remote Candidates

At any point of ICE processing, a trickle ICE agent may 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 simply 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 event. Upon receiving such an event, the local agent MUST update check list states as per Section 7.2. This may lead to some check lists being marked as Failed.

10. Receiving an End-Of-Candidates Notification

When an agent receives an end-of-candidates notification for a specific check list, they will update its state as per Section 7.2. In case the list is still in the Active state after the update, the agent will persist the the fact that an end-of-candidates notification has been received for and take it into account in future list updates.

11. Trickle ICE and Peer Reflexive Candidates

Even though Trickle ICE does not explicitly modify the procedures for handling peer reflexive candidates, their processing could be impacted in implementations. With Trickle ICE, it is possible that server reflexive candidates 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 signalling path.

12. Concluding ICE Processing

This specification does not directly modify the procedures ending ICE processing described in Section 8 of [RFC5245], and trickle ICE implementations will follow the same rules.

13. Subsequent Offer/Answer Exchanges

Either agent MAY generate a subsequent offer at any time allowed by [RFC3264]. When this happens agents will use [RFC5245] semantics to determine whether or not the new offer requires an ICE restart. If

this is the case then agents would perform trickle ICE as they would in an initial offer/answer exchange.

The only differences between an ICE restart and a brand new media session are that:

- during the restart, media can continue to be sent to the previously validated pair.

- both agents are already aware whether or not their peer supports trickle ICE, and there is no longer need for performing half trickle or confirming support with other mechanisms.

14. Interaction with ICE Lite

Behaviour of Trickle ICE capable ICE lite agents does not require any particular rules other than those already defined in this specification and [RFC5245]. This section is hence added with an informational purpose only.

A Trickle ICE capable ICE Lite agent would generate offers or answers as per [RFC5245]. Both will indicate support for trickle ICE (Section 4.1) and given that they will contain a complete set of candidates (the agent's host candidates) these offers and answers would also be accompanied with an end-of-candidates notification.

When performing full trickle, a full ICE implementation could send an offer or an answer with no candidates and an IP6 :: connection line address. After receiving an answer that identifies the remote agent as an ICE lite implementation, the offerer may very well choose to not send any additional candidates. The same is also true in the case when the ICE lite agent is making the offer and the full ICE one is answering. 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:

```
           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 1: Example

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

15. Unilateral Use of Trickle ICE (Half Trickle)

In half trickle mode, the offerer sends a regular, vanilla ICE offer, with a complete set of candidates. This ensures that the offer can be processed by a vanilla ICE answerer and is mostly meant for use in cases where support for trickle ICE cannot be confirmed prior to sending a initial offer. The initial offer indicates support for trickle ICE, so that the answerer can respond with an incomplete set of candidates and continue trickling the rest. Half trickle offers typically contain an end-of-candidates indication, although this is not mandatory because if trickle support is confirmed then the offerer can choose to trickle additional candidates before it declares end of trickling.

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 it contains a full set of candidates, its initial offer can thus be handled by a regular vanilla ICE agent, while still allowing a trickle one to use the optimisation 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 offer/answer exchange. Once both parties have received a session description from

their peer, they can each reliably determine trickle ICE support and use it for all subsequent offer/answer 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 offer, 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 offer. Because that the answerer will be able to trickle candidates, both agents will be able to start connectivity checks and complete ICE processing earlier than with vanilla ICE and potentially even as early as with full trickle.

However, such anticipation is not 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 vanilla ICE because it would improve the experience for answerers.

16. Example Flow

A typical successful trickle ICE exchange with an Offer/Answer protocol would look this way:

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

Figure 2: Example

17. Security Considerations

This specification inherits most of its semantics from [RFC5245] and as a result all security considerations described there remain the same.

18. Acknowledgements

The authors would like to thank Bernard Aboba, Flemming Andreasen, Rajmohan Banavi, 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.

19. References

19.1. Normative References

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

[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.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

19.2. 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-03 (work in progress), October 2015.

[I-D.keranen-mmusic-ice-address-selection]
Keraenen, A. and J. Arkko, "Update on Candidate Address Selection for Interactive Connectivity Establishment (ICE)", draft-keranen-mmusic-ice-address-selection-01 (work in progress), July 2012.

[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>.

[RFC2543] Handley, M., Schulzrinne, H., Schooler, E., and J. Rosenberg, "SIP: Session Initiation Protocol", RFC 2543, DOI 10.17487/RFC2543, March 1999, <http://www.rfc-editor.org/info/rfc2543>.

[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.

[RFC3388] Camarillo, G., Eriksson, G., Holler, J., and H. Schulzrinne, "Grouping of Media Lines in the Session Description Protocol (SDP)", RFC 3388, DOI 10.17487/RFC3388, December 2002, <http://www.rfc-editor.org/info/rfc3388>.

[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.

[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.

[XEP-0278] Camargo, T., "XEP-0278: Jingle Relay Nodes", XEP XEP-0278, June 2011.

Appendix A. Open Issues

At the time of writing of this document the authors have no clear view on how and if the following list of issues should be addressed.

A.1. MID/Stream Indices in SDP

This specification does not currently define syntax for candidate-to-stream bindings although it says that they should be implemented with MID or a stream index. Yet, it is reasonable to assume that most usages would need to do this within the SDP and it may make sense to agree on the format. Here's one possible way to do this:

```
a=mid:1
a=candidate:1 1 UDP 1658497328 192.168.100.33 5000 typ host
a=candidate:2 1 UDP 1658497328 96.1.2.3 5000 typ srflx
a=mid:2
a=candidate:2 1 UDP 1658497328 96.1.2.3 5002 typ srflx
a=end-of-candidates
```

A.2. Starting Checks

Normally vanilla ICE implementations would first activate a check list, validate at least one pair in every component and only then unfreeze all other checklists. With trickle ICE this would be suboptimal since candidates can arrive randomly and we would be wasting time waiting for a checklist to fill (almost as if we were doing vanilla ICE). We need to decide if unfreezing everything solely based on foundation is good enough.

A.3. Checklist States

It's been proposed that we add a waiting-for-candidates state (e.g., if the checklist is empty and no candidate pairs have been sent or received yet).

A.4. Relationship to Continuous Nomination and Passive Nomination

Does it make sense to tie trickle ICE more explicitly the continuous nomination and passive nomination specs? In particular, is address mobility a goal for the trickle ICE specification?

A.5. ICE Restarts

We need to describe how trickle ICE interacts with ICE restarts. Specifically, is it sufficient to modify the ufrag and pwd without starting a full offer/answer exchange, if the signaling protocol being used does not require it or if the restarting entity does not include a media description?

A.6. Candidate Redundancy and Priority

We need to clarify the relationship between RFC 5245 and trickle ICE with respect to candidate redundancy and priority.

A.7. Make Trickle ICE SDP-Agnostic

Would it make sense to remove the tie to SDP in the spec? This is similar to what's being done with the ICEbis spec, so consistency might be desirable.

Appendix B. Interaction with ICE

The ICE protocol was designed to be flexible enough to would work in and adapt to as many network environments as possible. Despite that flexibility, ICE as specified in [RFC5245] does not by itself support trickle ICE. This section describes how trickling of candidates interacts with ICE.

[RFC5245] 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 10.0.0.10 which also happens to correspond to an existing host on Bob's network.

3. Bob creates a check list consisting solely of 10.0.0.10 and starts checks.

4. These checks reach the host at 10.0.0.10 in Bob's network, which responds with an ICMP "port unreachable" error and per [RFC5245] 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 offer from Alice only contains candidates that can be determined as unreachable (per [I-D.keranen-mmusic-ice-address-selection]) from any of the candidates that Bob has gathered. 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 sends an offer to a trickle one. 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 offer it would immediately start connectivity checks. It would also start gathering candidates, which would take long 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 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-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.2. 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.3. 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.

- Added wording that explicitly states ICE lite agents have to be prepared to receive no candidates over signalling 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.

C.4. 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.

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

- Added two new entries to the open issues section.

C.5. 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
Jitsi
Strasbourg 67000
France

Phone: +33 6 72 81 15 55
Email: emcho@jitsi.org

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
&yet

Email: peter@andyet.com
URI: https://andyet.com/