Snowflake - A Lightweight, Asymmetric, Flexible, Receiver Driven Connectivity Establishment

draft-jennings-dispatch-snowflake-01

Abstract

Interactive Connectivity Establishment (ICE) (RFC5245) defines protocol machinery for two peers to discover each other and establish connectivity in order to send and receive Media Streams.

This draft raises some issues inherent in the assumptions with ICE and proposes a lightweight receiver driven protocol for asymmetric connectivity establishment.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 3, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

Jennings & Nandakumar   Expires September 3, 2018
1. Introduction

ICE was designed over a decade ago and certain assumptions about the network topology, timing considerations, application complexity have drastically changed since then. Newer additions/clarifications to ICE in [I-D.ietf-ice-rfc5245bis] and Trickle ICE [I-D.ietf-ice-trickle] have indeed help improve its performance and the way the connectivity checks are performed.

However, enforcing stringent global pacing requirements coupled with brute force connectivity checks, tightly coupled timing dependencies between the ICE agents, the need for symmetric connection setup, for example, has rendered the protocol inflexible for innovation and increasingly difficult to apply and debug in a dynamic network and evolving application contexts.

This specification defines Snowflake, where, like ICE, both sides gather a set of address candidates that may work for communication. However, instead of both sides trying to synchronize connectivity checks in time-coupled fashion, the sending side acts as a slave and sends STUN packets wherever the receiving side tells it to and when it is told to do so. The receiving side is free to choose whatever algorithm and timing it wants to find a path that works. The sender
and receiver roles are reversed for media flow in the opposite direction.

The current version of this draft builds on its original instantiation submitted in year 2015 as <https://datatracker.ietf.org/doc/draft-jennings-mmusic-ice-fix/>

2. Terminology

In this document, the key words "MUST", "MUST NOT", "SHOULD", "SHOULD NOT", "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119 [RFC2119] and indicate requirement levels for compliant implementations.

3. Problem Statement

ICE was developed roughly ten years ago and several things have been learned that could be improved:

1. It is spectacularly difficult to debug and analyze failures or successes in ICE or develop good automated tests. Many implementations have had significant bugs for long periods of time. This is further complicated by the timing dependency as explained next.

2. It is timing dependent. It relies on both sides to do something (candidate pairing, validation) at roughly the same time and that ability to do this goes down with the number of interfaces and candidates being handled. Mobile interfaces, dual stack agents make this situation worse.

3. Differences in interpretation and implementation of the protocol with respect to aggressive vs normal nomination may hinder rapid convergence or may end up in agents choosing suboptimal routes.

4. It does not discover asymmetric routes. For example UDP leaving a device may work just fine even though UDP coming into that device may not work at all.

5. Many deployments consider using a TURN/Media Router in their topology today in order to support fast session start or ensuring reliable connection (although with small latency overhead). At the time ICE was designed it was not understood if this would be too expensive or not so. ICE works without TURN but better with it.

6. The asymmetric nature of the controlling / controlled roles has caused many interoperability problems and bugs. Also Role
conflicts might lead to degrade connection setup depending on which side gets the the controlling role.

7. Priorities are complicated in dual stack world and ICE is brittle to changes in this part of the algorithm. Although there are advises in [I-D.ietf-ice-dualstack-fairness] specification that might help here.

4. Snowflake for connectivity establishment

Snowflake is a light weight, asymmetric, flexible and receiver controlled protocol for end points to establish connectivity between them.

The following subsections go into further details of its working.

4.1. System Components

A typical Snowflake operating model has the following components

- Sender Agent: A Software agent interested in sending data stream(s) to a remote receiver.

- Receiver Agent: A Software agent capable of receiving data stream(s).

- Snowflake Agent: A software agent that is expected to have a STUN Client implementation at the minimum for gathering candidates and performing connectivity checks. Sender/Receiver agents are Snowflake agents as well.

- CallAgent/Backchannel: Publicly reachable Server in the cloud accessible by both the Sender and the receiver agents, acts as backchannel/message bus for carrying signals between the Snowflake agents.

- STUN Server: Optional component for determining the public facing transport address of an agent behind NAT.

- TURN Server/Media Router: Recommended component acting as media relay between the agents. A TURN Server can also act as backchannel in certain instantiations.

4.2. Protocol Workings

The basic principle here is, each side (Receiver Agent) is responsible for discovering a viable path for it’s incoming media. It does so by indicating the addresses for the Sender to verify the
connectivity. Once a viable path is established, the Sender Agent continues to transmit the media. This process deviates from ICE by negating the need for agent's role (controlled vs controlling), nomination procedures (aggressive vs passive) and tightly coupled symmetric checklists validation.

As a precursor to connectivity establishment, the protocol assumes that there exists a dedicated backchannel that the agents can use to exchange protocol control messages.

The protocol starts with the Sender Agent conveying its intention to send media via the backchannel to the Receiver agent. The sender does so by sending a "PlaceCall" control message and populates the same with the ICE candidates gathered so far.

On receiving the sender’s intention to send media (via the backchannel), the Receiver Agent proceeds with gathering the candidates defined by its local policies or previous knowledge of connectivity checks. The Receiver Agent then directs the Sender Agent to carryout STUN connectivity checks towards the receiver by sending the "DoPing" control message via the backchannel. This message is populated with the candidate pair that the receiver wants the sender to verify the reachability.

The Receiver Agent may sends multiple "DoPing" messages to the Sender Agent, sending "DoPing" message per candidate pair to be tested for connectivity, as deemed necessary. The order, the timing and the number of candidate pairs to be tested are fully controlled by the Receiver Agent’s implementation.

On receiving the "DoPing" message with the candidate pair to be tested, the Sender Agent carries out STUN ping checks on that candidate pair. It does so by sending the STUN Binding Request message towards the receiver over the media path (as its done with ICE today). This opens up the required local pinholes and are further maintained by the Sender for the duration of the session. The Sender Agent also ensures that the frequency and the timing of these checks respect the congestion control requirements for the underlying transport.

On receiving the STUN Ping from the Sender Agent, the Receiver Agent does the following two things:

1. It responds to the connectivity check on the media path by sending a STUN Binding Response.
2. It also sends a "GotPing" control message with the details from the STUN Binding Response over the backchannel to the Sender
Agent. This is done so that the Sender Agent can verify the connectivity status results over the backchannel as well. This mechanism is especially beneficial for one-sided media scenarios where the Receiver Agent can’t send the STUN response to the sender or if the response to STUN connectivity response was lost in transmission.

If a successful STUN Ping response was received (either via the media path or the backchannel), there is a viable path for the Sender to transmit the media.

The above set of procedures can be continuously performed during the lifetime of the session as and when the Receiver Agent determines better candidates for receiving the media. Such a decision is totally defined by the local policies and can be performed independently of the other side.

Below picture captures one instance of protocol exchange where the Receiver Agent indicates the Sender Agent to carry out the connectivity checks. One can envision multiple executions of the protocol as and when receiver has updated its knowledge of addresses or priorities or bandwidth availability.

Snowflake Information Flow (One-way Media)

```
Sender Agent          CallAgent(backchannel)          Receiver Agent
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>(1) connect to backchannel</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Gather Sender Candidates</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>(2) PlaceCall [Sender Candidates]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>(3) PlaceCall [Sender Candidates]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Gather Receiver Candidates</td>
</tr>
</tbody>
</table>
```
4.3. Advantages of Snowflake

4.3.1. Diagnostics

This makes it very easy to see which outbound connection were sent from the Sender Agent to open a pin hole. Then when the Sender asked the Receiver Agent to send a test STUN Ping, the connectivity can be easily verified. It becomes easier to set up a client with an automated test jig that tests all the combinations and makes sure they work as you only need to test receiving capability and sender capability independently.

4.3.2. Timing

This more or less removes the timing complexity by allowing both sides to be responsible for their own timing. If it turns out that
we can pace things much faster than 50ms then this allows us to take advantage of that without both sides upgrading at the same time. If we end up with a lot more candidates due to v6, mobile etc, this removes the issue we have today where a path might have worked but the two sides did not find it due to timing issues.

4.3.3. Asymmetric Media

This allows media to be sent in one direction over a path that does not work in the reverse direction.

4.3.4. Fast Start

Given there exists a dedicated backchannel, this protocol can speed up the media flow by using TURN server for the backchannel, for example. Once either agents learns more about the candidates, each can update the other side to ensure a better low latency path is used for media.

5. IANA Consideration

TODO

6. Security Considerations

TODO

7. Acknowledgements

TODO

8. References

8.1. Normative References


8.2. Informative References

[I-D.ietf-ice-rfc5245bis]

[I-D.ietf-ice-trickle]

Authors’ Addresses

Cullen Jennings
Cisco
Email: fluffy@iii.ca

Suhas Nandakumar
Cisco
Email: snandaku@cisco.com