   Interactive Connectivity Establishment (ICE): A Protocol for Network
       Address Translator (NAT) Traversal for Offer/Answer Protocols
                         draft-ietf-mmusic-ice-17

Status of this Memo

Copyright Notice

Abstract

   This document describes a protocol for Network Address Translator
   (NAT) traversal for multimedia sessions established with the offer/
   answer model. This protocol is called Interactive Connectivity
   Establishment (ICE). ICE makes use of the Session Traversal
   Utilities for NAT (STUN) protocol and its extension, Traversal Using
   Relay NAT (TURN). ICE can be used by any protocol utilizing the

      offer/answer model, such as the Session Initiation Protocol (SIP).

Table of Contents

1. Introduction

   RFC 3264 [RFC3264] defines a two-phase exchange of Session
   Description Protocol (SDP) messages [RFC4566] for the purposes of
   establishment of multimedia sessions. This offer/answer mechanism is
   used by protocols such as the Session Initiation Protocol (SIP)
   [RFC3261].

   Protocols using offer/answer are difficult to operate through Network
   Address Translators (NAT). Because their purpose is to establish a
   flow of media packets, they tend to carry the IP addresses and ports
   of media sources and sinks within their messages, which is known to
   be problematic through NAT [RFC3235]. The protocols also seek to
   create a media flow directly between participants, so that there is
   no application layer intermediary between them. This is done to
   reduce media latency, decrease packet loss, and reduce the
   operational costs of deploying the application. However, this is
   difficult to accomplish through NAT. A full treatment of the reasons
   for this is beyond the scope of this specification.

   Numerous solutions have been proposed for allowing these protocols to
   operate through NAT. These include Application Layer Gateways
   (ALGs), the Middlebox Control Protocol [RFC3303], Simple Traversal
   Underneath NAT (STUN) [RFC3489] and its revision, retitled Session
   Traversal Utilities for NAT [I-D.ietf-behave-rfc3489bis], Traversal
   Using Relay NAT (TURN) [I-D.ietf-behave-turn], and Realm Specific IP
   [RFC3102] [RFC3103] along with session description extensions needed
   to make them work, such as the Session Description Protocol (SDP)
   [RFC4566] attribute for the Real Time Control Protocol (RTCP)
   [RFC3605]. Unfortunately, these techniques all have pros and cons
   which make each one optimal in some network topologies, but a poor
   choice in others. The result is that administrators and implementors
   are making assumptions about the topologies of the networks in which
   their solutions will be deployed. This introduces complexity and
   brittleness into the system. What is needed is a single solution
   which is flexible enough to work well in all situations.

   This specification defines Interactive Connectivity Establishment
   (ICE) as a technique for NAT traversal for media streams established
   by the offer/answer model. ICE is an extension to the offer/answer
   model, and works by including a multiplicity of IP addresses and
   ports in SDP offers and answers, which are then tested for
   connectivity by peer-to-peer STUN exchanges. The IP addresses and
   ports included in the SDP are gathered using STUN
   [I-D.ietf-behave-rfc3489bis] and Traversal Using Relay NAT (TURN)
   [I-D.ietf-behave-turn]. Because ICE exchanges a multiplicity of IP
   addresses and ports for each media stream, it also allows for address
   selection for multi-homed and dual-stack hosts, and for this reason

it deprecates RFC 4091 [RFC4091].

2. Overview of ICE

In a typical ICE deployment, we have two endpoints (known as AGENTS
in RFC 3264 terminology) which want to communicate. They are able to
communicate indirectly via some signaling protocol (such as SIP), by
which they can perform an offer/answer exchange of SDP [RFC3264]
messages. Note that ICE is not intended for NAT traversal for SIP,
which is assumed to be provided via another mechanism
[I-D.ietf-sip-outbound]. At the beginning of the ICE process, the
agents are ignorant of their own topologies. In particular, they
might or might not be behind a NAT (or multiple tiers of NATs). ICE
allows the agents to discover enough information about their
topologies to potentially find one or more paths by which they can
communicate.

Figure 1 shows a typical environment for ICE deployment. The two
endpoints are labelled L and R (for left and right, which helps
visualize call flows). Both L and R are behind their own respective
NATs though they may not be aware of it. The type of NAT and its
properties are also unknown. Agents L and R are capable of engaging
in an offer/answer exchange by which they can exchange SDP messages,
whose purpose is to set up a media session between L and R.
Typically, this exchange will occur through a SIP server.

In addition to the agents, a SIP server and NATs, ICE is typically
used in concert with STUN or TURN servers in the network. Each agent
can have its own STUN or TURN server, or they can be the same.

```
                            +-------+
                            | SIP   |
             +-------+      | Srvr  |            +-------+
             | STUN  |      |       |            | STUN  |
             | Srvr  |      +-------+            | Srvr  |
             |       |       /     \            |       |
             +-------+      /       \           +-------+
                           /         \
                          /           \
                         /             \
                        /               \
                       /  <-  Signalling ->  \
                      /                 \
                     /                   \
                +--------+           +--------+
                |  NAT   |           |  NAT   |
                +--------+           +--------+
                   /                     \
                  /                       \
                 /                         \
            +-------+                   +-------+
            | Agent |                   | Agent |
            |   L   |                   |   R   |
            |       |                   |       |
            +-------+                   +-------+
```
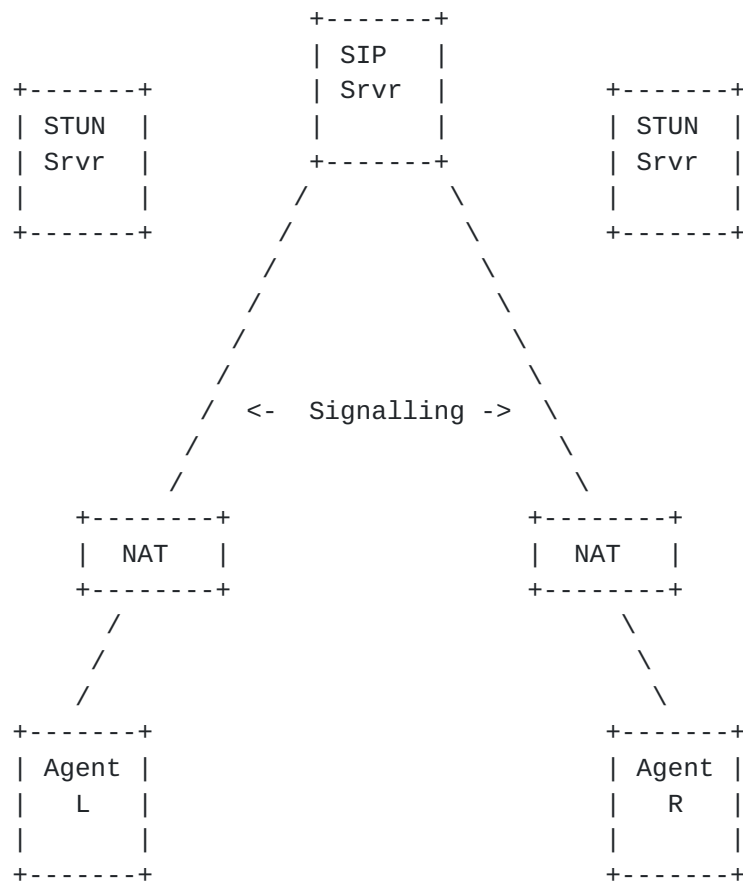
                  Figure 1: ICE Deployment Scenario

   The basic idea behind ICE is as follows: each agent has a variety of
   candidate TRANSPORT ADDRESSES (combination of IP address and port) it
   could use to communicate with the other agent. These might include:

   o A transport address on a directly attached network interface

   o A translated transport address on the public side of a NAT (a
      "server reflexive" address)

   o The transport address allocated from a TURN server(a "relayed
      address".

   Potentially, any of L's candidate transport addresses can be used to
   communicate with any of R's candidate transport addresses. In
   practice, however, many combinations will not work. For instance, if
   L and R are both behind NATs, their directly attached interface
   addresses are unlikely to be able to communicate directly (this is
   why ICE is needed, after all!). The purpose of ICE is to discover
   which pairs of addresses will work. The way that ICE does this is to
   systematically try all possible pairs (in a carefully sorted order)

until it finds one or more that works.

2.1. Gathering Candidate Addresses

In order to execute ICE, an agent has to identify all of its address
candidates. A CANDIDATE is a transport address - a combination of IP
address and port for a particular transport protocol. This document
defines three types of candidates, some derived from physical or
logical network interfaces, others discoverable via STUN and TURN.
Naturally, one viable candidate is a transport address obtained
directly from a local interface. Such a candidate is called a HOST
CANDIDATE. The local interface could be ethernet or WiFi, or it
could be one that is obtained through a tunnel mechanism, such as a
Virtual Private Network (VPN) or Mobile IP (MIP). In all cases, such
a network interface appears to the agent as a local interface from
which ports (and thus a candidate) can be allocated.

If an agent is multihomed, it obtains a candidate from each
interface. Depending on the location of the PEER (the other agent in
the session) on the IP network relative to the agent, the agent may
be reachable by the peer through one or more of those interfaces.
Consider, for example, an agent which has a local interface to a
private net 10 network (I1), and a second connected to the public
Internet (I2). A candidate from I1 will be directly reachable when
communicating with a peer on the same private net 10 network, while a
candidate from I2 will be directly reachable when communicating with
a peer on the public Internet. Rather than trying to guess which
interface will work prior to sending an offer, the offering agent
includes both candidates in its offer.

Next, the agent uses STUN or TURN to obtain additional candidates.
These come in two flavors: translated addresses on the public side of
a NAT (SERVER REFLEXIVE CANDIDATES) and addresses on TURN servers
(RELAYED CANDIDATES). When TURN servers are utilized, both types of
candidates are obtained from the TURN server. If only STUN servers
are utilized, only server reflexive canddiates are obtained from
them. The relationship of these candidates to the host candidate is
shown in Figure 2. In this figure, both types of candidates are
discovered using TURN. In the figure, the notation X:x means IP
address X and port x.

```
                To Internet


                    |
                    |
                    |  /------------  Relayed
              Y:y | /               Address
              +--------+
              |        |
              |  TURN  |
              | Server |
              |        |
              +--------+

                   |
                   |
                   | /------------  Server
            X1':x1'|/               Reflexive
             +------------+         Address
             |    NAT     |
             +------------+

                  |
                  | /------------  Local
            X:x |/                 Address
             +--------+
             |        |
             | Agent  |
             |        |
             +--------+
```
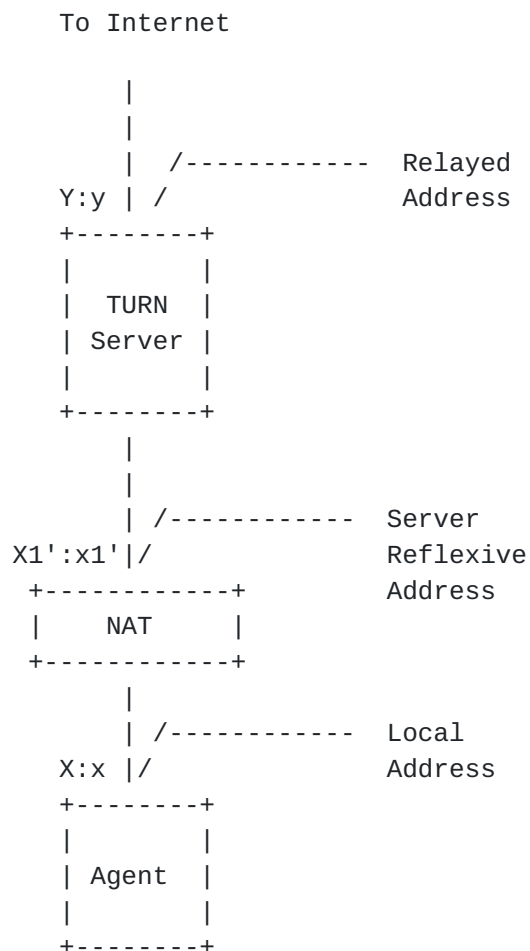
Figure 2: Candidate Relationships

When the agent sends the TURN Allocate Request from IP address and port X:x, the NAT (assuming there is one) will create a binding X1':x1', mapping this server reflexive candidate to the host candidate X:x. Outgoing packets sent from the host candidate will be translated by the NAT to the server reflexive candidate. Incoming packets sent to the server relexive candidate will be translated by the NAT to the host candidate and forwarded to the agent. We call the host candidate associated with a given server reflexive candidate the BASE.

NOTE: "Base" refers to the address an agent sends from for a particular candidate. Thus, as a degenerate case host candidates also have a base, but it's the same as the host candidate.

When there are multiple NATs between the agent and the TURN server, the TURN request will create a binding on each NAT, but only the outermost server reflexive candidate (the one nearest the TURN

server) will be discovered by the agent. If the agent is not behind
a NAT, then the base candidate will be the same as the server
reflexive candidate and the server reflexive candidate is redundant
and will be eliminated.

The Allocate request then arrives at the TURN server. The TURN
server allocates a port y from its local interface Y, and generates
an Allocate response, informing the agent of this relayed candidate.
The TURN server also informs the agent of the server reflexive
candidate, X1':x1' by copying the source transport address of the
Allocate request into the Allocate response. The TURN server acts as
a packet relay, forwarding traffic between L and R. In order to send
traffic to L, R sends traffic to the TURN server at Y:y, and the TURN
server forwards that to X1':x1', which passes through the NAT where
it is mapped to X:x and delivered to L.

When only STUN servers are utilized, the agent sends a STUN Binding
Request [I-D.ietf-behave-rfc3489bis] to its STUN server. The STUN
server will inform the agent of the server reflexive candidate
X1':x1' by copying the source transport address of the Binding
request into the Binding response.

2.2. Connectivity Checks

Once L has gathered all of its candidates, it orders them in highest
to lowest priority and sends them to R over the signalling channel.
The candidates are carried in attributes in the SDP offer. When R
receives the offer, it performs the same gathering process and
responds with its own list of candidates. At the end of this
process, each agent has a complete list of both its candidates and
its peer's candidates. It pairs them up, resulting in CANDIDATE
PAIRS. To see which pairs work, the agent schedules a series of
CHECKS. Each check is a STUN request/response transaction that the
client will perform on a particular candidate pair by sending a STUN
request from the local candidate to the remote candidate.

The basic principle of the connectivity checks is simple:

1. Sort the candidate pairs in priority order.

2. Send checks on each candidate pair in priority order.

3. Acknowledge checks received from the other agent.

With both agents performing a check on a candidate pair, the result
is a 4-way handshake:

```
   L                          R
   -                          -
   STUN request ->              \  L's
            <- STUN response   /  check

             <- STUN request  \  R's
   STUN response ->            /  check
```

Figure 3: Basic Connectivity Check

It is important to note that the STUN requests are sent to and from
the exact same IP addresses and ports that will be used for media
(e.g., RTP and RTCP). Consequently, agents demultiplex STUN and RTP/
RTCP using contents of the packets, rather than the port on which
they are received. Fortunately, this demultiplexing is easy to do,
especially for RTP and RTCP.

Because a STUN Binding Request is used for the connectivity check,
the STUN Binding response will contain the agent's translated
transport address on the public side any NATs between the agent and
its peer. If this transport address is different from other
candidates the agent already learned, it represents a new candidate,
called a PEER REFLEXIVE CANDIDATE, which then gets tested by ICE just
the same as any other candidate.

As an optimization, as soon as R gets L's check message, R schedules
a connectivity check message to be sent to L on the same candidate
pair. This accelerates the process of finding a valid candidate, and
is called a TRIGGERED CHECK.

At the end of this handshake, both L and R know that they can send
(and receive) messages end-to-end in both directions.

2.3. Sorting Candidates

Because the algorithm above searches all candidate pairs, if a
working pair exists it will eventually find it no matter what order
the candidates are tried in. In order to produce faster (and better)
results, the candidates are sorted in a specified order. The
resulting list of sorted candidate pairs is called the CHECK LIST.
The algorithm is described in Section 4.1.2 but follows two general
principles:

o Each agent gives its candidates a numeric priority which is sent
   along with the candidate to the peer

o The local and remote priorities are combined so that each agent
   has the same ordering for the candidate pairs.

The second property is important for getting ICE to work when there
are NATs in front of L and R. Frequently, NATs will not allow packets
in from a host until the agent behind the NAT has sent a packet
towards that host. Consequently, ICE checks in each direction will
not succeed until both sides have sent a check through their
respective NATs.

The agent works through this check list by sending a STUN request for
the next candidate pair on the list periodically. These are called
ORDINARY CHECKS.

In general the priority algorithm is designed so that candidates of
similar type get similar priorities and so that more direct routes
(that is, through fewer media relays and through fewer NATs) are
preferred over indirect ones (ones with more media relays and more
NATs). Within those guidelines, however, agents have a fair amount
of discretion about how to tune their algorithms.

2.4. Frozen Candidates

The previous description only addresses the case where the agents
wish to establish a media session with one COMPONENT (a piece of a
media stream requiring a single transport address; a media stream may
require multiple components, each of which has to work for the media
stream as a whole to be work). Typically, (e.g., with RTP and RTCP)
the agents actually need to establish connectivity for more than one
flow.

The network properties are likely to be very similar for each
component (especially because RTP and RTCP are sent and received from
the same IP address). It is usually possible to leverage information
from one media component in order to determine the best candidates
for another. ICE does this with a mechanism called "frozen
candidates."

Each candidate is associated with a property called its FOUNDATION.
Two candidates have the same foundation when they are "similar" - of
the same type and obtained from the same interface and STUN server
using the same protocol. Otherwise, their foundation is different.
A candidate pair has a foundation too, which is just the
concatenation of the foundations of its two candidates. Initially,
only the candidate pairs with unique foundations are tested. The
other candidate pairs are marked "frozen". When the connectivity
checks for a candidate pair succeed, the other candidate pairs with
the same foundation are unfrozen. This avoids repeated checking of
components which are superficially more attractive but in fact are
likely to fail.

While we've described "frozen" here as a separate mechanism for
expository purposes, in fact it is an integral part of ICE and the
the ICE prioritization algorithm automatically ensures that the right
candidates are unfrozen and checked in the right order.

2.5. Security for Checks

Because ICE is used to discover which addresses can be used to send
media between two agents, it is important to ensure that the process
cannot be hijacked to send media to the wrong location. Each STUN
connectivity check is covered by a message authentication code (MAC)
computed using a key exchanged in the signalling channel. This MAC
provides message integrity and data origin authentication, thus
stopping an attacker from forging or modifying connectivity check
messages. The MAC also aids in disambiguating ICE exchanges from
forked calls when ICE is used with SIP [RFC3261].

2.6. Concluding ICE

ICE checks are performed in a specific sequence, so that high
priority candidate pairs are checked first, followed by lower
priority ones. One way to conclude ICE is to declare victory as soon
as a check for each component of each media stream completes
successfully. Indeed, this is a reasonable algorithm, and details
for it are provided below. However, it is possible that packet
losses will cause a higher priority check to take longer to complete.
In that case, allowing ICE to run a little longer might produce
better results. More fundamentally, however, the prioritization
defined by this specification may not yield "optimal" results. As an
example, if the aim is to select low latency media paths, usage of a
relay is a hint that latencies may be higher, but it is nothing more
than a hint. An actual RTT measurement could be made, and it might
demonstrate that a pair with lower priority is actually better than
one with higher priority.

Consequently, ICE assigns one of the agents in the role of the
CONTROLLING AGENT, and the other of the CONTROLLED AGENT. The
controlling agent gets to nominate which candidate pairs will get
used for media amongst the ones that are valid. It can do this in
one of two ways - using REGULAR NOMINATION or AGGRESSIVE NOMINATION.

With regular nomination, the controlling agent lets the checks
continue until at least one valid candidate pair for each media
stream is found. Then, it picks amongst those that are valid, and
sends a second STUN request on its NOMINATED candidate pair, but this
time with a flag set to tell the peer that this pair has been
nominated for use. This is shown in Figure 4.

```
   L                           R
   -                           -
   STUN request ->             \  L's
            <- STUN response   /  check

             <- STUN request   \  R's
   STUN response ->            /  check

   STUN request + flag ->      \  L's
            <- STUN response   /  check
```

                      Figure 4: Regular Nomination

   Once the STUN transaction with the flag completes, both sides cancel
   any future checks for that media stream. ICE will now send media
   using this pair. The pair an ICE agent is using for media is called
   the SELECTED PAIR.

   In aggressive nomination, the controlling agent puts the flag in
   every STUN request it sends. This way, once the first check
   succeeds, ICE processing is complete for that media stream and the
   controlling agent doesn't have to send a second STUN request. The
   selected pair will be the highest priority valid pair whose check
   succeeeded. Aggressive nomination is faster than regular nomination,
   but gives less flexibility. Aggressive nomination is shown in
   Figure 5.

```
   L                           R
   -                           -
   STUN request + flag ->      \  L's
            <- STUN response   /  check

             <- STUN request   \  R's
   STUN response ->            /  check
```

                     Figure 5: Aggressive Nomination

   Once all of the media streams are completed, the controlling endpoint
   sends an updated offer if the candidates in the m and c lines for the
   media stream (called the DEFAULT CANDIDATES) don't match ICE's
   SELECTED CANDIDATES.

   Once ICE is concluded, it can be restarted at any time for one or all
   of the media streams by either agent. This is done by sending an
   updated offer indicating a restart.

2.7. Lite Implementations

   In order for ICE to be used in a call, both agents need to support
   it. However, certain agents will always be connected to the public
   Internet and have a public IP address at which it can receive packets
   from any correspondent. To make it easier for these devices to
   support ICE, ICE defines a special type of implementation called LITE
   (in contrast to the normal FULL implementation). A lite
   implementation doesn't gather candidates; it includes only host
   candidates for any media stream. Lite agents do not generate
   connectivity checks or run the state machines, though they need to be
   able to respond to connectivity checks. When a lite implementation
   connects with a full implementation, the full agent takes the role of
   the controlling agent, and the lite agent takes on the controlled
   role. When two lite implementations connect, no checks are sent.

   For guidance on when a lite implementation is appropriate, see the
   discussion in Appendix A.

   It is important to note that the lite implementation was added to
   this specification to provide a stepping stone to full
   implementation. Even for devices that are always connected to the
   public Internet, a full implementation is preferable if achievable.

3. 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 RFC 2119 [RFC2119].

   Readers should be familiar with the terminology defined in the offer/
   answer model [RFC3264], STUN [I-D.ietf-behave-rfc3489bis] and NAT
   Behavioral requirements for UDP [RFC4787]

   This specification makes use of the following additional terminology:

   Agent: As defined in RFC 3264, an agent is the protocol
      implementation involved in the offer/answer exchange. There are
      two agents involved in an offer/answer exchange.

   Peer: From the perspective of one of the agents in a session, its
      peer is the other agent. Specifically, from the perspective of
      the offerer, the peer is the answerer. From the perspective of
      the answerer, the peer is the offerer.

Transport Address: The combination of an IP address and transport
   protocol (such as UDP or TCP) port.

Candidate: A transport address that is a potential point of contact
   for receipt of media. Candidates also have properties - their
   type (server reflexive, relayed or host), priority, foundation,
   and base.

Component: A component is a piece of a media stream requiring a
   single transport address; a media stream may require multiple
   components, each of which has to work for the media stream as a
   whole to work. For media streams based on RTP, there are two
   components per media stream - one for RTP, and one for RTCP.

Host Candidate: A candidate obtained by binding to a specific port
   from an interface on the host. This includes both physical
   interfaces and logical ones, such as ones obtained through Virtual
   Private Networks (VPNs) and Realm Specific IP (RSIP) [RFC3102]
   (which lives at the operating system level).

Server Reflexive Candidate: A candidate whose IP address and port
   are a binding allocated by a NAT for an agent when it sent a
   packet through the NAT to a server. Server reflexive candidates
   can be learned by STUN servers using the Binding Request, or TURN
   servers, which provides both a Relayed and Server Reflexive
   candidate.

Peer Reflexive Candidate: A candidate whose IP address and port are
   a binding allocated by a NAT for an agent when it sent a STUN
   Binding Request through the NAT to its peer.

Relayed Candidate: A candidate obtained by sending a TURN Allocate
   request from a host candidate to a TURN server. The relayed
   candidate is resident on the TURN server, and the TURN server
   relays packets back towards the agent.

Base: The base of a server reflexive candidate is the host candidate
   from which it was derived. A host candidate is also said to have
   a base, equal to that candidate itself. Similarly, the base of a
   relayed candidate is that candidate itself.

Foundation: An arbitrary string that is the same for two candidates
   that have the same type, base IP address, protocol (UDP, TCP,
   etc.) and STUN or TURN server. If any of these are different then
   the foundation will be different. Two candidate pairs with the
   same foundation pairs are likely to have similar network
   characteristics. Foundations are used in the frozen algorithm.

Local Candidate: A candidate that an agent has obtained and included
   in an offer or answer it sent.

Remote Candidate: A candidate that an agent received in an offer or
   answer from its peer.

Default Destination/Candidate: The default destination for a
   component of a media stream is the transport address that would be
   used by an agent that is not ICE aware. For the RTP component,
   the default IP address is in the c line of the SDP, and the port
   in the m line. For the RTCP component it is in the rtcp attribute
   when present, and when not present, the IP address in the c line
   and 1 plus the port in the m line. A default candidate for a
   component is one whose transport address matches the default
   destination for that component.

Candidate Pair: A pairing containing a local candidate and a remote
   candidate.

Check, Connectivity Check, STUN Check: A STUN Binding Request
   transaction for the purposes of verifying connectivity. A check
   is sent from the local candidate to the remote candidate of a
   candidate pair.

Check List: An ordered set of candidate pairs that an agent will use
   to generate checks.

Ordinary Check: A connectivity check generated by an agent as a
   consequence of a timer that fires periodically, instructing it to
   send a check.

Triggered Check: A connectivity check generated as a consequence of
   the receipt of a connectivity check from the peer.

Valid List: An ordered set of candidate pairs for a media stream
   that have been validated by a successful STUN transaction.

Full: An ICE implementation that performs the complete set of
   functionality defined by this specification.

Lite: An ICE implementation that omits certain functions,
   implementing only as much as is necessary for a peer
   implementation that is full to gain the benefits of ICE. Lite
   implementations do not maintain any of the state machines and do
   not generate connectivity checks.

Controlling Agent: The ICE agent which is responsible for selecting
   the final choice of candidate pairs and signaling them through
   STUN and an updated offer, if needed. In any session, one agent
   is always controlling. The other is the controlled agent.

Controlled Agent: An ICE agent which waits for the controlling agent
   to select the final choice of candidate pairs.

Regular Nomination: The process of picking a valid candidate pair
   for media traffic by validating the pair with one STUN request,
   and then picking it by sending a second STUN request with a flag
   indicating its nomination.

Aggressive Nomination: The process of picking a valid candidate pair
   for media traffic by including a flag in every STUN request, such
   that the first one to produce a valid candidate pair is used for
   media.

Nominated: If a valid candidate pair has its nominated flag set, it
   means that it may be selected by ICE for sending and receiving
   media.

Selected Pair, Selected Candidate: The candidate pair selected by
   ICE for sending and receiving media is called the selected pair,
   and each of its candidates is called the selected candidate.

4. Sending the Initial Offer

In order to send the initial offer in an offer/answer exchange, an
agent must (1) gather candidates, (2) prioritize them, (3) choose
default candidates, and then (4) formulate and send the SDP. The
first of these four steps differ for full and lite implementations.

4.1. Full Implementation Requirements

4.1.1. Gathering Candidates

An agent gathers candidates when it believes that communications is
imminent. An offerer can do this based on a user interface cue, or
based on an explicit request to initiate a session. Every candidate
is a transport address. It also has a type and a base. Three types
are defined and gathered by this specification - host candidates,
server reflexive candidates, and relayed candidates. The server
reflexive and relayed candidates are gathered using STUN or TURN, and
relayed candidates are obtained through TURN. The base of a
candidate is the candidate that an agent must send from when using
that candidate.

4.1.1.1. Host Candidates

The first step is to gather host candidates. Host candidates are obtained by binding to ports (typically ephemeral) on an interface (physical or virtual, including VPN interfaces) on the host. The process for gathering host candidates depends on the transport protocol. Procedures are specified here for UDP.

For each UDP media stream the agent wishes to use, the agent SHOULD obtain a candidate for each component of the media stream on each interface that the host has. It obtains each candidate by binding to a UDP port on the specific interface. A host candidate (and indeed every candidate) is always associated with a specific component for which it is a candidate. Each component has an ID assigned to it, called the component ID. For RTP-based media streams, the RTP itself has a component ID of 1, and RTCP a component ID of 2. If an agent is using RTCP it MUST obtain a candidate for it. If an agent is using both RTP and RTCP, it would end up with 2*K host candidates if an agent has K interfaces.

The base for each host candidate is set to the candidate itself.

4.1.1.2. Server Reflexive and Relayed Candidates

Agents SHOULD obtain relayed candidates and SHOULD obtain server reflexive candidates. These requirements are at SHOULD strength to allow for provider variation. Use of STUN and TURN servers may be unnecessary in closed networks where agents are never connected to the public Internet or to endpoints outside of the closed network. In such cases, a full implementation would be used for agents that are dual-stack or multi-homed, to select a host candidate. Use of TURN servers is expensive, and when ICE is being used, they will only be utilized when both endpoints are behind NATs that perform address and port dependent mapping. Consequently, some deployments might consider this use case to be marginal, and elect not to use TURN servers. If an agent does not gather server reflexive or relayed candidates, it is RECOMMENDED that the functionality be implemented and just disabled through configuration, so that it can re-enabled through configuration if conditions change in the future.

If an agent is gathering both relayed and server reflexive candidates, it uses a TURN server. If it is gathering just server reflexive candidates, it uses a STUN server.

The agent next pairs each host candidate with the STUN or TURN server with which it is configured or has discovered by some means. If a STUN or TURN server is configured, it is RECOMMENDED that a domain name be configured, and the DNS procedures in

[I-D.ietf-behave-rfc3489bis] (using SRV records with the "stun"
service) be used to discover the STUN server, and the DNS procedures
in [I-D.ietf-behave-turn] (using SRV records with the "turn" service)
be used to discover the TURN server.

This specification only considers usage of a single STUN or TURN
server. When there are multiple choices for that single STUN or TURN
server (when, for example, they are learned through DNS records and
multiple results are returned), an agent SHOULD use a single STUN or
TURN server (based on its IP address) for all candidates for a
particular session. This improves the performance of ICE. The
result is a set of pairs of host candidates with STUN or TURN
servers. The agent then chooses one pair, and sends a Binding or
Allocate request to the server from that host candidate. Binding
Requests to a STUN server are not authenticated, and any ALTERNATE-
SERVER attribute in a response is ignored. Agents MUST support the
backwards compatibility mode for the Binding Request defined in
[I-D.ietf-behave-rfc3489bis]. Allocate requests SHOULD be
authenticated using a long-term credential provisioned into the
client.

Every Ta milliseconds thereafter, the agent can generate another new
STUN or TURN transaction. This transaction can either be a retry of
a previous transaction which failed with a recoverable error (such as
authentication failure), or a transaction for a new host candidate
and STUN or TURN server pair. The agent SHOULD NOT generate
transactions more frequently than one every Ta milliseconds. See
Section 16 for guidance on how to set Ta and the STUN retransmit
timer, RTO.

The agent will receive a Binding or Allocate response. A successful
Allocate Response will provide the agent with a server reflexive
candidate (obtained from the mapped address) and a relayed candidate
in the RELAY-ADDRESS attribute. If the Allocate request is rejected
because the server lacks resources to fulfill it, the agent SHOULD
instead send a Binding Request to obtain a server reflexive
candidate. A Binding Response will provide the agent with only a
server reflexive candidate (also obtained from the mapped address).
The base of the server reflexive candidate is the host candidate from
which the Allocate or Binding request was sent. The base of a
relayed candidate is that candidate itself. If a relayed candidate
is identical to a host candidate (which can happen in rare cases),
the relayed candidate MUST be discarded.

4.1.1.3. Eliminating Redundant Candidates

Next, the agent eliminates redundant candidates. A candidate is
redundant if its transport address equals another candidate, and its

base equals the base of that other candidate. Note that two
candidates can have the same transport address yet have different
bases, and these would not be considered redundant. Frequently, a
server reflexive candidate and a host candidate will be redundant
when the agent is not behind a NAT.

4.1.1.4. Computing Foundations

Finally, the agent assigns each candidate a foundation. The
foundation is an identifier, scoped within a session. Two candidates
MUST have the same foundation ID when all of the following are true:

o they are of the same type (host, relayed, server reflexive, or
   peer reflexive)

o their bases have the same IP address (the ports can be different)

o for reflexive and relayed candidates, the STUN or TURN servers
   used to obtain them have the same IP address.

o they were obtained using the same transport protocol (TCP, UDP,
   etc.)

Similarly, two candidates MUST have different foundations if their
types are different, their bases have different IP addresses, the
STUN or TURN servers used to obtain them have different IP addresses,
or their transport protocols are different.

4.1.1.5. Keeping Candidates Alive

Once server reflexive and relayed candidates are allocated, they MUST
be kept alive until ICE processing has completed, as described in
Section 8.3. For server reflexive candidates learned through a
Binding request, the bindings MUST be kept alive by another Binding
Request to the server. For relayed candidates learned through an
Allocate request, the keepalive MUST be a new Allocate request. The
Allocate request will also refresh the server reflexive candidate.

4.1.2. Prioritizing Candidates

The prioritization process results in the assignment of a priority to
each candidate. Each candidate for a media stream MUST have a unique
priority that MUST be a positive integer between 1 and (2**32 - 1).
This priority will be used by ICE to determine the order of the
connectivity checks and the relative preference for candidates.

An agent SHOULD compute this priority using the formula in
Section 4.1.2.1 and choose its parameters using the guidelines in

Section 4.1.2.2. If an agent elects to use a different formula, ICE
will take longer to converge since both agents will not be
coordinated in their checks.

4.1.2.1. Recommended Formula

When using the formula, an agent computes the priority by determining
a preference for each type of candidate (server reflexive, peer
reflexive, relayed and host), and, when the agent is multihomed,
choosing a preference for its interfaces. These two preferences are
then combined to compute the priority for a candidate. That priority
is computed using the following formula:

priority = (2^24)*(type preference) +
           (2^8)*(local preference) +
           (2^0)*(256 - component ID)

The type preference MUST be an integer from 0 to 126 inclusive, and
represents the preference for the type of the candidate (where the
types are local, server reflexive, peer reflexive and relayed). A
126 is the highest preference, and a 0 is the lowest. Setting the
value to a 0 means that candidates of this type will only be used as
a last resort. The type preference MUST be identical for all
candidates of the same type and MUST be different for candidates of
different types. The type preference for peer reflexive candidates
MUST be higher than that of server reflexive candidates. Note that
candidates gathered based on the procedures of Section 4.1.1 will
never be peer reflexive candidates; candidates of these type are
learned from the connectivity checks performed by ICE.

The local preference MUST be an integer from 0 to 65535 inclusive.
It represents a preference for the particular interface from which
the candidate was obtained, in cases where an agent is multihomed.
65535 represents the highest preference, and a zero, the lowest.
When there is only a single interface, this value SHOULD be set to
65535. More generally, if there are multiple candidates for a
particular component for a particular media stream which have the
same type, the local preference MUST be unique for each one. In this
specification, this only happens for multi-homed hosts.

The component ID is the component ID for the candidate, and MUST be
between 1 and 256 inclusive.

4.1.2.2. Guidelines for Choosing Type and Local Preferences

   One criteria for selection of the type and local preference values is
   the use of a media intermediary, such as a TURN server, VPN server or
   NAT. With a media intermediary, if media is sent to that candidate,
   it will first transit the media intermediary before being received.
   Relayed candidates are one type of candidate that involves a media
   intermediary. Another are host candidates obtained from a VPN
   interface. When media is transited through a media intermediary, it
   can increase the latency between transmission and reception. It can
   increase the packet losses, because of the additional router hops
   that may be taken. It may increase the cost of providing service,
   since media will be routed in and right back out of a media
   intermediary run by a provider. If these concerns are important, the
   type preference for relayed candidates SHOULD be lower than host
   candidates. The RECOMMENDED values are 126 for host candidates, 100
   for server reflexive candidates, 110 for peer reflexive candidates,
   and 0 for relayed candidates. Furthermore, if an agent is multi-
   homed and has multiple interfaces, the local preference for host
   candidates from a VPN interface SHOULD have a priority of 0.

   Another criteria for selection of preferences is IP address family.
   ICE works with both IPv4 and IPv6. It therefore provides a
   transition mechanism that allows dual-stack hosts to prefer
   connectivity over IPv6, but to fall back to IPv4 in case the v6
   networks are disconnected (due, for example, to a failure in a 6to4
   relay) [RFC3056]. It can also help with hosts that have both a
   native IPv6 address and a 6to4 address. In such a case, higher local
   preferences could be assigned to the v6 interface, followed by the
   6to4 interfaces, followed by the v4 interfaces. This allows a site
   to obtain and begin using native v6 addresses immediately, yet still
   fallback to 6to4 addresses when communicating with agents in other
   sites that do not yet have native v6 connectivity.

   Another criteria for selecting preferences is security. If a user is
   a telecommuter, and therefore connected to their corporate network
   and a local home network, they may prefer their voice traffic to be
   routed over the VPN in order to keep it on the corporate network when
   communicating within the enterprise, but use the local network when
   communicating with users outside of the enterprise. In such a case,
   a VPN interface would have a higher local preference than any other
   interface.

   Another criteria for selecting preferences is topological awareness.
   This is most useful for candidates that make use of intermediaries.
   In those cases, if an agent has preconfigured or dynamically
   discovered knowledge of the topological proximity of the
   intermediaries to itself, it can use that to assign higher local

preferences to candidates obtained from closer intermediaries.

4.1.3. Choosing Default Candidates

A candidate is said to be default if it would be the target of media
from a non-ICE peer; that target being called the DEFAULT
DESTINATION. If the default candidates are not selected by the ICE
algorithm when communicating with an ICE-aware peer, an updated
offer/answer will be required after ICE processing completes in order
to "fix-up" the SDP so that the default destination for media matches
the candidates selected by ICE. If ICE happens to select the default
candidates, no updated offer/answer is required.

An agent MUST choose a set of candidates, one for each component of
each in-use media stream, to be default. A media stream is in-use if
it does not have a port of zero (which is used in RFC 3264 to reject
a media stream). Consequently, a media stream is in-use even if it
is marked as a=inactive [RFC4566] or has a bandwidth value of zero.

It is RECOMMENDED that default candidates be chosen based on the
likelihood of those candidates to work with the peer that is being
contacted. It is RECOMMENDED that the default candidates are the
relayed candidates (if relayed candidates are available), server
reflexive candidates (if server reflexive candidates are available),
and finally host candidates.

4.2. Lite Implementation

Lite implementations only utilize host candidates. A lite
implementation MUST, for each component of each media stream,
allocate zero or one IPv4 candidates. It MAY allocate zero or more
IPv6 candidates, but no more than one per each IPv6 address utilized
by the host. Since there can be no more than one IPv4 candidate per
component of each media stream, if an agent has multiple IPv4
interfaces, it MUST choose one for allocating the candidate. If a
host is dual-stack, it is RECOMMENDED that it allocate one IPv4
candidate and one global IPv6 address. With the lite implementation,
ICE cannot be used to dynamically choose amongst candidates.
Therefore, including more than one candidate from a particular scope
is NOT RECOMMENDED, since only a connectivity check can truly
determine whether to use one address or the other.

Each component has an ID assigned to it, called the component ID.
For RTP-based media streams the RTP itself has a component ID of 1,
and RTCP a component ID of 2. If an agent is using RTCP it MUST
obtain candidates for it.

Each candidate is assigned a foundation. The foundation MUST be

different for two candidates allocated from different IP addresses,
and MUST be the same otherwise. A simple integer that increments for
each IP address will suffice. In addition, each candidate MUST be
assigned a unique priority amongst all candidates for the same media
stream. This priority SHOULD be equal to:


```
priority = (2^24)*(126) +
           (2^8)*(IP precedence) +
           (2^0)*(256 - component ID)
```


If a host is v4-only, it SHOULD set the IP precedence to 65535. If a
host is v6 or dual-stack, the IP precedence is the precedence value
for IP addresses described in RFC 3484 [RFC3484].

Next, an agent chooses a default candidate for each component of each
media stream. If a host is IPv4 only, there would only be one
candidate for each component of each media stream, and therefore that
candidate is the default. If a host is IPv6 or dual stack, the
selection of default is a matter of local policy. This default
SHOULD be chosen, such that, it is the candidate most likely to be
used with a peer. For IPv6-only hosts, this would typically by a
globally scoped IPv6 address. For dual-stack hosts, the IPv4 address
is RECOMMENDED.

4.3. Encoding the SDP

The process of encoding the SDP is identical between full and lite
implementations.

The agent will include an m-line for each media stream it wishes to
use. The ordering of media streams in the SDP is relevant for ICE.
ICE will perform its connectivity checks for the first m-line first,
and consequently media will be able to flow for that stream first.
Agents SHOULD place their most important media stream, if there is
one, first in the SDP.

There will be a candidate attribute for each candidate for a
particular media stream. Section 15 provides detailed rules for
constructing this attribute. The attribute carries the IP address,
port and transport protocol for the candidate, in addition to its
properties that need to be signaled to the peer for ICE to work: the
priority, foundation, and component ID. The candidate attribute also
carries information about the candidate that is useful for
diagnostics and other functions: its type and related transport
addresses.

STUN connectivity checks between agents are authenticated using the
short term credential mechanism defined for STUN
[I-D.ietf-behave-rfc3489bis]. This mechanism relies on a username
and password that are exchanged through protocol machinery between
the client and server. With ICE, the offer/answer exchange is used
to exchange them. The username part of this credential is formed by
concatenating a username fragment from each agent, separated by a
colon. Each agent also provides a password, used to compute the
message integrity for requests it receives. The username fragment
and password are exchanged in the ice-ufrag and ice-pwd attributes,
respectively. In addition to providing security, the username
provides disambiguation and correlation of checks to media streams.
See Appendix B.4 for motivation.

If an agent is a lite implementation, it MUST include an "a=ice-lite"
session level attribute in its SDP. If an agent is a full
implementation, it MUST NOT include this attribute.

The default candidates are added to the SDP as the default
destination for media. For streams based on RTP, this is done by
placing the IP address and port of the RTP candidate into the c and m
lines, respectively. If the agent is utilizing RTCP, it MUST encode
the RTCP candidate using the a=rtcp attribute as defined in RFC 3605
[RFC3605]. If RTCP is not in use, the agent MUST signal that using
b=RS:0 and b=RR:0 as defined in RFC 3556 [RFC3556].

The transport addresses that will be the default destination for
media when communicating with non-ICE peers MUST also be present as
candidates in one or more a=candidate lines.

ICE provides for extensibility by allowing an offer or answer to
contain a series of tokens which identify the ICE extensions used by
that agent. If an agent supports an ICE extension, it MUST include
the token defined for that extension in the ice-options attribute.

The following is an example SDP message that includes ICE attributes
(lines folded for readability):

```
      v=0
      o=jdoe 2890844526 2890842807 IN IP4 10.0.1.1
      s=
      c=IN IP4 192.0.2.3
      t=0 0
      a=ice-pwd:asd88fgpdd777uzjYhagZg
      a=ice-ufrag:8hhY
      m=audio 45664 RTP/AVP 0
      b=RS:0
      b=RR:0
      a=rtpmap:0 PCMU/8000
      a=candidate:1 1 UDP 2130706431 10.0.1.1 8998 typ host
      a=candidate:2 1 UDP 1694498815 192.0.2.3 45664 typ srflx raddr
   10.0.1.1 rport 8998
```

Once an agent has sent its offer or sent its answer, that agent MUST be prepared to receive both STUN and media packets on each candidate. As discussed in Section 11.1, media packets can be sent to a candidate prior to its appearance as the default destination for media in an offer or answer.

5. Receiving the Initial Offer

When an agent receives an initial offer, it will check if the offerer supports ICE, determine its own role, gather candidates, prioritize them, choose default candidates, encode and send an answer, and for full implementations, form the check lists and begin connectivity checks.

5.1. Verifying ICE Support

The agent will proceed with the ICE procedures defined in this specification if, for each media stream in the SDP it received, the default destination for each component of that media stream appears in a candidate attribute. For example, in the case of RTP, the IP address and port in the c and m line, respectively, appears in a candidate attribute and the value in the rtcp attribute appears in a candidate attribute.

If this condition is not met, the agent MUST process the SDP based on normal RFC 3264 procedures, without using any of the ICE mechanisms described in the remainder of this specification with the following exceptions:

1. The agent MUST follow the rules of Section 10, which describe keepalive procedures for all agents.

2. If the agent is not proceeding with ICE because there were
    a=candidate attributes, but none that matched the default
    destination of the media stream, the agent MUST include an a=ice-
    mismatch attribute in its answer.

5.2. Determining Role

For each session, each agent takes on a role. There are two roles -
controlling, and controlled. The controlling agent is responsible
for the choice of the final candidate pairs used for communications.
For a full agent, this means nominating the candidate pairs that can
be used by ICE for each media stream, and for generating the updated
offer based on ICE's selection, when needed. For a lite
implementation, being the controlling agent means selecting a
candidate pair based on the ones in the offer and answer (for IPv4,
there is only ever one pair), and then generating an updated offer
reflecting that selection, when needed (it is never needed for an
IPv4 only host). The controlled agent is told which candidate pairs
to use for each media stream, and does not generate an updated offer
to signal this information. The sections below describe in detail
the actual procedures following by controlling and controlled nodes.

The rules for determining the role and the impact on behavior are as
follows:

Both agents are full: The agent which generated the offer which
   started the ICE processing MUST take the controlling role, and the
   other MUST take the controlled role. Both agents will form check
   lists, run the ICE state machines, and generate connectivity
   checks. The controlling agent will execute the logic in
   Section 8.1 to nominate pairs that will be selected by ICE, and
   then both agents end ICE as described in Section 8.1.2. In
   unusual cases, described in Appendix B.11, it is possible for both
   agents to mistakenly believe they are controlled or controlling.
   To resolve this, each agent MUST select a random number, called
   the tie-breaker, uniformly distributed between 0 and (2**64) - 1
   (that is, a 64 bit positive integer). This number is used in
   connectivity checks to detect and repair this case, as described
   in Section 7.1.1.2.

One agent Full, one Lite: The full agent MUST take the controlling
   role, and the lite agent MUST take the controlled role. The full
   agent will form check lists, run the ICE state machines, and
   generate connectivity checks. That agent will execute the logic
   in Section 8.1 to nominate pairs that will be selected by ICE, and
   use the logic in Section 8.1.2 to end ICE. The lite
   implementation will just listen for connectivity checks, receive
   them and respond to them, and then conclude ICE as described in

Section 8.2. For the lite implementation, the state of ICE
processing for each media stream is considered to be Running, and
the state of ICE overall is Running.

Both Lite: The agent which generated the offer which started the ICE
   processing MUST take the controlling role, and the other MUST take
   the controlled role. In this case, no connectivity checks are
   ever sent. Rather, once the offer/answer exchange completes, each
   agent performs the processing described in Section 8 without
   connectivity checks. It is possible that both agents will believe
   they are controlled or controlling. In the latter case, the
   conflict is resolved through glare detection capabilities in the
   signaling protocol carrying the offer/answer exchange. The state
   of ICE processing for each media stream is considered to be
   Running, and the state of ICE overall is Running.

Once roles are determined for a session, they persist unless ICE is
restarted. A ICE restart (Section 9.1) causes a new selection of
roles and tie-breakers.

5.3. Gathering Candidates

The process for gathering candidates at the answerer is identical to
the process for the offerer as described in Section 4.1.1 for full
implementations and Section 4.2 for lite implementations. It is
RECOMMENDED that this process begin immediately on receipt of the
offer, prior to alerting the user. Such gathering MAY begin when an
agent starts.

5.4. Prioritizing Candidates

The process for prioritizing candidates at the answerer is identical
to the process followed by the offerer, as described in Section 4.1.2
for full implementations and Section 4.2 for lite implementations.

5.5. Choosing Default Candidates

The process for selecting default candidates at the answerer is
identical to the process followed by the offerer, as described in
Section 4.1.3 for full implementations and Section 4.2 for lite
implementations.

5.6. 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 4.3.

5.7. Forming the Check Lists

   Forming check lists is done only by full implementations. Lite
   implementations MUST skip the steps defined in this section.

   There is one check list per in-use media stream resulting from the
   offer/answer exchange. To form the check list for a media stream,
   the agent forms candidate pairs, computes a candidate pair priority,
   orders the pairs by priority, prunes them, and sets their states.
   These steps are described in this section.

5.7.1. Forming Candidate Pairs

   First, the agent takes each of its candidates for a media stream
   (called LOCAL CANDIDATES) and pairs them with the candidates it
   received from its peer (called REMOTE CANDIDATES) for that media
   stream. In order to prevent the attacks described in Section 18.5.2,
   agents MAY limit the number of candidates they'll accept in an offer
   or answer. A local candidate is paired with a remote candidate if
   and only if the two candidates have the same component ID and have
   the same IP address version. It is possible that some of the local
   candidates don't get paired with a remote candidate, and some of the
   remote candidates don't get paired with local candidates. This can
   happen if one agent didn't include candidates for the all of the
   components for a media stream. If this happens, the number of
   components for that media stream is effectively reduced, and
   considered to be equal to the minimum across both agents of the
   maximum component ID provided by each agent across all components for
   the media stream.

   In the case of RTP, this would happen when one agent provided
   candidates for RTCP, and the other did not. As another example, the
   offerer can multiplex RTP and RTCP on the same port and signals it
   can do that in the SDP through an SDP attribute
   [I-D.ietf-avt-rtp-and-rtcp-mux]. However, since the offerer doesn't
   know if the answerer can perform such multiplexing, the offerer
   includes candidates for RTP and RTCP on separate ports, so that the
   offer has two components per media stream. If the answerer can
   perform such multiplexing, it would include just a single component
   for each candidate - for the combined RTP/RTCP mux. ICE would end up
   acting as if there was just a single component for this candidate.

   The candidate pairs whose local and remote candidates were both the
   default candidates for a particular component is called,
   unsurprisingly, the default candidate pair for that component. This
   is the pair that would be used to transmit media if both agents had
   not been ICE aware.

In order to aid understanding, Figure 9 shows the relationships
between several key concepts - transport addresses, candidates,
candidate pairs, and check lists, in addition to indicating the main
properties of candidates and candidate pairs.

```
     +--------------------------------------------+
     |                                            |
     | +--------------------+                     |
     | |+----+ +----+ +----+ |    +Type           |
     | || IP | |Port| |Tran| |    +Priority       |
     | ||Addr| |    | |    | |    +Foundation     |
     | |+----+ +----+ +----+ |    +ComponentiD    |
     | |        Transport    |    +RelatedAddr    |
     | |          Addr       |                    |
     | +--------------------+    +Base            |
     |              Candidate                     |
     +--------------------------------------------+
     *                                          *
     *     ************************************
     *     *
   +--------------------------------+
  .|                                |
   | Local      Remote              |
   | +----+      +----+   +default? |
   | |Cand|      |Cand|   +valid?   |
   | +----+      +----+   +nominated?|
   |                      +State    |
   |                                |
   |                                |
   |         Candidate Pair         |
   +--------------------------------+
   *                              *
   *                 ***********
   *                 *
   +-----------------+
   |  Candidate Pair |
   +-----------------+
   +-----------------+
   |  Candidate Pair |
   +-----------------+
   +-----------------+
   |  Candidate Pair |
   +-----------------+


         Check
         List
```

Figure 9: Conceptual Diagram of a Check List

5.7.2. Computing Pair Priority and Ordering Pairs

Once the pairs are formed, a candidate pair priority is computed.
Let G be the priority for the candidate provided by the controlling
agent. Let D be the priority for the candidate provided by the
controlled agent. The priority for a pair is computed as:

pair priority = 2^32*MIN(G,D) + 2*MAX(G,D) + (G>D?1:0)

Where G>D?1:0 is an expression whose value is 1 if G is greater than
D, and 0 otherwise. This formula ensures a unique priority for each
pair. Once the priority is assigned, the agent sorts the candidate
pairs in decreasing order of priority. If two pairs have identical
priority, the ordering amongst them is arbitrary.

5.7.3. Pruning the Pairs

This sorted list of candidate pairs is used to determine a sequence
of connectivity checks that will be performed. Each check involves
sending a request from a local candidate to a remote candidate.
Since an agent cannot send requests directly from a reflexive
candidate, but only from its base, the agent next goes through the
sorted list of candidate pairs. For each pair where the local
candidate is server reflexive, the server reflexive candidate MUST be
replaced by its base. Once this has been done, the agent MUST prune
the list. This is done by removing a pair if its local and remote
candidates are identical to the local and remote candidates of a pair
higher up on the priority list. The result is a sequence of ordered
candidate pairs, called the check list for that media stream.

In addition, in order to limit the attacks described in
Section 18.5.2, an agent SHOULD limit the total number of
connectivity checks they perform across all check lists to 100, by
discarding the lower priority candidate pairs until there are less
than 100.

5.7.4. Computing States

Each candidate pair in the check list has a foundation and a state.
The foundation is the combination of the foundations of the local and
remote candidates in the pair. The state is assigned once the check
list for each media stream has been computed. There are five
potential values that the state can have:

Waiting: A check has not been performed for this pair, and can be
   performed as soon as it is the highest priority Waiting pair on
   the check list.

In-Progress: A check has been sent for this pair, but the
   transaction is in progress.

Succeeded: A check for this pair was already done and produced a
   successful result.

Failed: A check for this pair was already done and failed, either
   never producing any response or producing an unrecoverable failure
   response.

Frozen: A check for this pair hasn't been performed, and it can't
   yet be performed until some other check succeeds, allowing this
   pair to unfreeze and move into the Waiting state.

As ICE runs, the pairs will move between states as shown in
Figure 10.

```
          +-----------+
          |           |
          |           |
          |  Frozen   |
          |           |
          |           |
          +-----------+
               |
               |unfreeze
               |
               V
          +-----------+          +-----------+
          |           |          |           |
          |           | perform  |           |
          |  Waiting  |--------->|In-Progress|
          |           |          |           |
          |           |          |           |
          +-----------+          +-----------+
                                    / |
                                   // |
                                  //  |
                                 //   |
                                 /    |
                               //     |
                   failure //         |success
                         //           |
                         /            |
                       //             |
                      //              |
                     //               |
               V                      V
          +-----------+          +-----------+
          |           |          |           |
          |           |          |           |
          |  Failed   |          | Succeeded |
          |           |          |           |
          |           |          |           |
          +-----------+          +-----------+
```
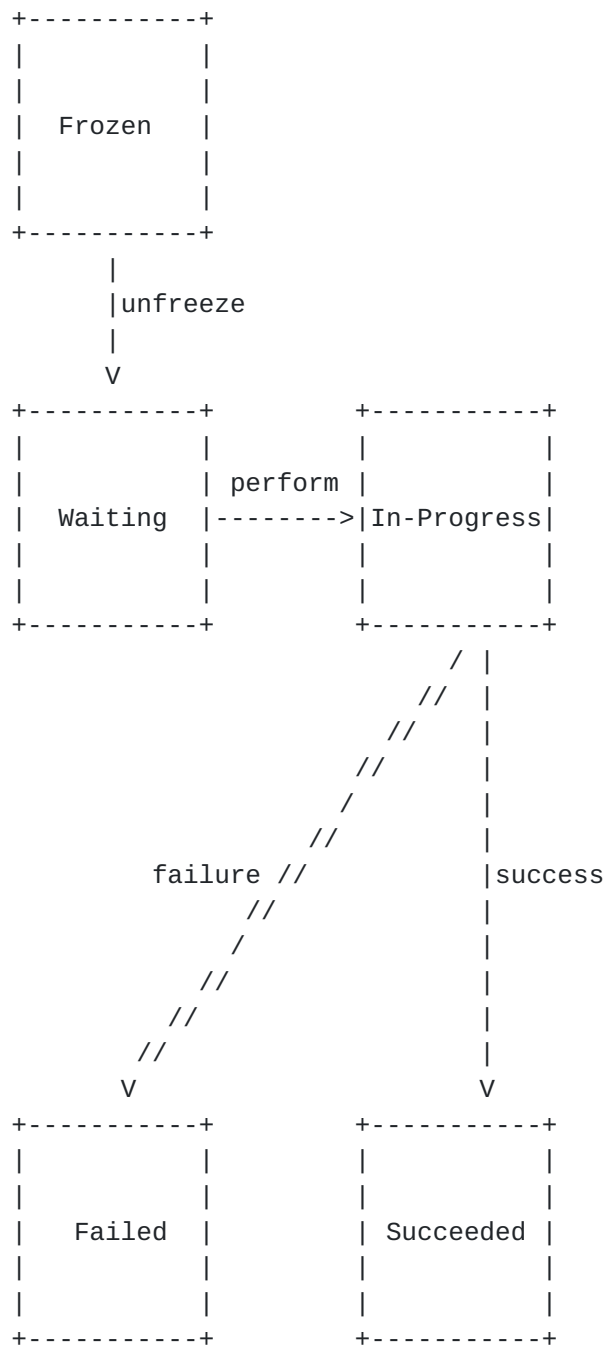
                       Figure 10: Pair State FSM

   The initial states for each pair in a check list are computed by
   performing the following sequence of steps:

   1. The agent sets all of the pairs in each check list to the Frozen
       state.

2.   The agent examines the check list for the first media stream (a
     media stream is the first media stream when it is described by
     the first m-line in the SDP offer and answer). For that media
     stream, it:

     * Groups together all of the pairs with the same foundation,

     * For each group, sets the state of the pair with the lowest
        component ID to Waiting. If there is more than one such pair,
        the one with the highest priority is used.

One of the check lists will have some number of pairs in the Waiting
state, and the other check lists will have all of their pairs in the
Frozen state. A check list with at least one pair that is Waiting is
called an active check list, and a check list with all pairs frozen
is called a frozen check list.

The check list itself is associated with a state, which captures the
state of ICE checks for that media stream. There are three states:

Running: In this state, ICE checks are still in progress for this
   media stream.

Completed: In this state, ICE checks have produced nominated pairs
   for each component of the media stream. Consequently, ICE has
   succeeded and media can be sent.

Failed: In this state, the ICE checks have not completed
   successfully for this media stream.

When a check list is first constructed as the consequence of an
offer/answer exchange, it is placed in the Running state.

ICE processing across all media streams also has a state associated
with it. This state is equal to Running while ICE processing is
underway. The state is Completed when ICE processing is complete and
Failed if it failed without success. Rules for transitioning between
states are described below.

5.8. Scheduling Checks

Checks are generated only by full implementations. Lite
implementations MUST skip the steps described in this section.

An agent performs ordinary checks and triggered checks. The
generation of both checks is governed by a timer which fires
periodically for each media stream. The agent maintains a FIFO
queue, called the triggered check queue, which contains candidate

pairs for which checks are to be sent at the next available
opportunity. When the timer fires, the agent removes the top pair
from triggered check queue, performs a connectivity check on that
pair, and sets the state of the candidate pair to In-Progress. If
there are no pairs in the triggered check queue, an ordinary check is
sent.

Once the agent has computed the check lists as described in
Section 5.7, it sets a timer for each active check list. The timer
fires every Ta*N seconds, where N is the number of active check lists
(initially, there is only one active check list). Implementations
MAY set the timer to fire less frequently than this. Implementations
SHOULD take care to spread out these timers so that they do not fire
at the same time for each media stream. Ta and the retransmit timer
RTO are computed as described in Section 16. Multiplying by N allows
this aggregate check throughput to be split between all active check
lists. The first timer fires immediately, so that the agent performs
a connectivity check the moment the offer/answer exchange has been
done, followed by the next check Ta seconds later (since there is
only one active check list).

When the timer fires, and there is no triggered check to be sent, the
agent MUST choose an ordinary check as follows:

o Find the highest priority pair in that check list that is in the
   Waiting state.

o If there is such a pair:

   * Send a STUN check from the local candidate of that pair to the
      remote candidate of that pair. The procedures for forming the
      STUN request for this purpose are described in Section 7.1.1.

   * Set the state of the candidate pair to In-Progress.

o If there is no such pair:

   * Find the highest priority pair in that check list that is in
      the Frozen state.

   * If there is such a pair:

      + Unfreeze the pair.

      + Perform a check for that pair, causing its state to
         transition to In-Progress.

> * If there is no such pair:
>
>> + Terminate the timer for that check list.

To compute the message integrity for the check, the agent uses the remote username fragment and password learned from the SDP from its peer. The local username fragment is known directly by the agent for its own candidate.

6. Receipt of the Initial Answer

This section describes the procedures that an agent follows when it receives the answer from the peer. It verifies that its peer supports ICE, determines its role, and for full implementations, forms the check list and begins performing ordinary checks.

When ICE is used with SIP, forking may result in a single offer generating a multiplicity of answers. In that each, ICE proceeds completely in parallel and independently for each answer, treating the combination of its offer and each answer as an independent offer/answer exchange, with its own set of pairs, check lists, states, and so on. The only case in which processing of one pair impacts another is freeing of candidates, discussed below in Section 8.3.

6.1. Verifying ICE Support

The logic at the offerer is identical to that of the answerer as described in Section 5.1, with the exception that an offerer would not ever generate a=ice-mismatch attributes in an SDP.

In some cases, the answer may omit a=candidate attributes for the media streams, and instead include an a=ice-mismatch attribute for one or more of the media streams in the SDP. This signals to the offerer that the answerer supports ICE, but that ICE processing was not used for the session because a signaling intermediary modified the default destination for media components without modifying the corresponding candidate attributes. See Section 18 for a discussion of cases where this can happen. This specification provides no guidance on how an agent should proceed in such a failure case.

6.2. Determining Role

The offerer follows the same procedures described for the answerer in Section 5.2.

6.3. Forming the Check List

   Formation of check lists is performed only by full implementations.
   The offerer follows the same procedures described for the answerer in
   Section 5.7.

6.4. Performing Ordinary Checks

   Ordinary checks are performed only by full implementations. The
   offerer follows the same procedures described for the answerer in
   Section 5.8.

7. Performing Connectivity Checks

   This section describes how connectivity checks are performed. All
   ICE implementations are required to be compliant to
   [I-D.ietf-behave-rfc3489bis], as opposed to the older [RFC3489].
   However, whereas a full implementation will both generate checks
   (acting as a STUN client) and receive them (acting as a STUN server),
   a lite implementation will only ever receive checks, and thus will
   only act as a STUN server.

7.1. STUN Client Procedures

   These procedures define how an agent sends a connectivity check,
   whether it is an ordinary or a triggered check. These procedures are
   only applicable to full implementations.

7.1.1. Sending the Request

   The check is generated by sending a Binding Request from a local
   candidate, to a remote candidate. [I-D.ietf-behave-rfc3489bis]
   describes how Binding Requests are constructed and generated. A
   connectivity check MUST utilize the STUN short term credential
   mechanism. Support for backwards compatibility with RFC 3489 MUST
   NOT be used or assumed with connectivity checks. The FINGERPRINT
   mechanism MUST be used for connectivity checks.

   ICE extends STUN by defining several new attributes, including
   PRIORITY, USE-CANDIDATE, ICE-CONTROLLED, and ICE-CONTROLLING. These
   new attributes are formally defined in Section 19.1, and their usage
   is described in the subsections below. These STUN extensions are
   applicable only to connectivity checks used for ICE.

7.1.1.1. **PRIORITY and USE-CANDIDATE**

   An agent MUST include the PRIORITY attribute in its Binding Request.
   The attribute MUST be set equal to the priority that would be
   assigned, based on the algorithm in Section 4.1.2, to a peer
   reflexive candidate, should one be learned as a consequence of this
   check (see Section 7.1.2.2.1 for how peer reflexive candidates are
   learned). This priority value will be computed identically to how
   the priority for the local candidate of the pair was computed, except
   that the type preference is set to the value for peer derived
   candidate types.

   The controlling agent MAY include the USE-CANDIDATE attribute in the
   Binding Request. The controlled agent MUST NOT include it in its
   Binding Request. This attribute signals that the controlling agent
   wishes to cease checks for this component, and use the candidate pair
   resulting from the check for this component. Section 8.1.1 provides
   guidance on determining when to include it.

7.1.1.2. **ICE-CONTROLLED and ICE-CONTROLLING**

   The agent MUST include the ICE-CONTROLLED attribute in the request if
   it is in the controlled role, and MUST include the ICE-CONTROLLING
   attribute in the request if it is in the controlling role. The
   content of either attribute MUST be the tie breaker that was
   determined in Section 5.2. These attributes are defined fully in
   Section 19.1.

7.1.1.3. **Forming Credentials**

   A Binding Request serving as a connectivity check MUST utilize the
   STUN short term credential mechanism. The username for the
   credential is formed by concatenating the username fragment provided
   by the peer with the username fragment of the agent sending the
   request, separated by a colon (":"). The password is equal to the
   password provided by the peer. For example, consider the case where
   agent L is the offerer, and agent R is the answerer. Agent L
   included a username fragment of LFRAG for its candidates, and a
   password of LPASS. Agent R provided a username fragment of RFRAG and
   a password of RPASS. A connectivity check from L to R (and its
   response of course) utilize the username RFRAG:LFRAG and a password
   of RPASS. A connectivity check from R to L (and its response)
   utilize the username LFRAG:RFRAG and a password of LPASS.

7.1.1.4. **DiffServ Treatment**

   If the agent is using Diffserv Codepoint markings [RFC2475] in its
   media packets, it SHOULD apply those same markings to its

connectivity checks.

7.1.2. Processing the Response

   When a Binding Response is received, it is correlated to its Binding
   Request using the transaction ID, as defined in
   [I-D.ietf-behave-rfc3489bis], which then ties it to the candidate
   pair for which the Binding Request was sent. This section defines
   additional procedures for processing Binding Responses, specific to
   this usage of STUN.

7.1.2.1. Failure Cases

   If the STUN transaction generates a 487 (Role Conflict) error
   response, the agent checks whether it had included the ICE-CONTROLLED
   or ICE-CONTROLLING attribute in the Binding Request. If the request
   had contained the ICE-CONTROLLED attribute, the agent MUST switch to
   the controlling role if it has not already done so. If the request
   had contained the ICE-CONTROLLING attribute, the agent MUST switch to
   the controlled role if it has not already done so. Once it has
   switched, the agent MUST enqueue the candidate pair whose check
   generated the 487 into the triggered check queue. The state of that
   pair is set to Waiting. When the triggered check is sent, it will
   contain an ICE-CONTROLLING or ICE-CONTROLLED attribute reflecting its
   new role. Note, however, that the tie-breaker value MUST NOT be
   reselected.

   Agents MAY support receipt of ICMP errors for connectivity checks.
   If the STUN transaction generates an ICMP error, the agent sets the
   state of the pair to Failed. If the STUN transaction generates a
   STUN error response that is unrecoverable (as defined in
   [I-D.ietf-behave-rfc3489bis]), or times out, the agent sets the state
   of the pair to Failed.

   The agent MUST check that the source IP address and port of the
   response equals the destination IP address and port that the Binding
   Request was sent to, and that the destination IP address and port of
   the response match the source IP address and port that the Binding
   Request was sent from. In other words, the source and destination
   transport addresses in the request and responses are the symmetric.
   If they are not symmetric, the agent sets the state of the pair to
   Failed.

7.1.2.2. Success Cases

   A check is considered to be a success if all of the following are
   true:

o the STUN transaction generated a success response

o the source IP address and port of the response equals the
   destination IP address and port that the Binding Request was sent
   to

o the destination IP address and port of the response match the
   source IP address and port that the Binding Request was sent from

7.1.2.2.1. Discovering Peer Reflexive Candidates

The agent checks the mapped address from the STUN response. If the
transport address does not match any of the local candidates that the
agent knows about, the mapped address represents a new candidate - a
peer reflexive candidate. Like other candidates, it has a type,
base, priority and foundation. They are computed as follows:

o Its type is equal to peer reflexive.

o Its base is set equal to the local candidate of the candidate pair
   from which the STUN check was sent.

o Its priority is set equal to the value of the PRIORITY attribute
   in the Binding Request.

o Its foundation is selected as described in Section 4.1.1.

This peer reflexive candidate is then added to the list of local
candidates for the media stream. Its username fragment and password
are the same as all other local candidates for that media stream.
However, the peer reflexive candidate is not paired with other remote
candidates. This is not necessary; a valid pair will be generated
from it momentarily based on the procedures in Section 7.1.2.2.2. If
an agent wishes to pair the peer reflexive candidate with other
remote candidates besides the one in the valid pair that will be
generated, the agent MAY generate an updated offer which includes the
peer reflexive candidate. This will cause it to be paired with all
other remote candidates.

7.1.2.2.2. Constructing a Valid Pair

The agent constructs a candidate pair whose local candidate equals
the mapped address of the response, and whose remote candidate equals
the destination address to which the request was sent. This is
called a valid pair, since it has been validated by a STUN
connectivity check. The valid pair may equal the pair that generated
the check, may equal a different pair in the check list, or may be a
pair not currently on any check list. If the pair equals the pair

that generated the check or is on a check list currently, it is also
added to the VALID LIST, which is maintained by the agent for each
media stream. This list is empty at the start of ICE processing, and
fills as checks are performed, resulting in valid candidate pairs.

It will be very common that the pair will not be on any check list.
Recall that the check list has pairs whose local candidates are never
server reflexive; those pairs had their local candidates converted to
the base of the server reflexive candidates, and then pruned if they
were redundant. When the response to the STUN check arrives, the
mapped address will be reflexive if there is a NAT between the two.
In that case, the valid pair will have a local candidate that doesn't
match any of the pairs in the check list.

If the pair is not on any check list, the agent computes the priority
for the pair based on the priority of each candidate, using the
algorithm in Section 5.7. The priority of the local candidate
depends on its type. If it is not peer reflexive, it is equal to the
priority signaled for that candidate in the SDP. If it is peer
reflexive, it is equal to the PRIORITY attribute the agent placed in
the Binding Request which just completed. The priority of the remote
candidate is taken from the SDP of the peer. If the candidate does
not appear there, then the check must have been a triggered check to
a new remote candidate. In that case, the priority is taken as the
value of the PRIORITY attribute in the Binding Request which
triggered the check that just completed. The pair is then added to
the VALID LIST.

7.1.2.2.3. Updating Pair States

The agent sets the state of the pair that generated the check to
Succeeded. The success of this check might also cause the state of
other checks to change as well. The agent MUST perform the following
two steps:

1. The agent changes the states for all other Frozen pairs for the
    same media stream and same foundation to Waiting. Typically
    these other pairs will have different component IDs but not
    always.

2. If there is a pair in the valid list for every component of this
    media stream (where this is the actual number of components being
    used, in cases where the number of components signaled in the SDP
    differs from offerer to answerer), the success of this check may
    unfreeze checks for other media streams. Note that this step is
    followed not just the first time the valid list under
    consideration has a pair for every component, but every
    subsequent time a check succeeds and adds yet another pair to

that valid list. The agent examines the check list for each
other media stream in turn:

* If the check list is active, the agent changes the state of
   all Frozen pairs in that check list whose foundation matches a
   pair in the valid list under consideration, to Waiting.

* If the check list is frozen, and there is at least one pair in
   the check list whose foundation matches a pair in the valid
   list under consideration, the state of all pairs in the check
   list whose foundation matches a pair in the valid list under
   consideration are set to Waiting. This will cause the check
   list to become active, and ordinary checks will begin for it,
   as described in Section 5.8.

* If the check list is frozen, and there are no pairs in the
   check list whose foundation matches a pair in the valid list
   under consideration, the agent

   + Groups together all of the pairs with the same foundation,

   + For each group, sets the state of the pair with the lowest
      component ID to Waiting. If there is more than one such
      pair, the one with the highest priority is used.

7.1.2.2.4. Updating the Nominated Flag

If the agent was a controlling agent, and it had included a USE-
CANDIDATE attribute in the Binding Request, the valid pair generated
from that check has its nominated flag set to true. This flag
indicates that this valid pair should be used for media if it is the
highest priority one amongst those whose nominated flag is set. This
may conclude ICE processing for this media stream or all media
streams; see Section 8.

If the agent is the controlled agent, the response may result in the
valid pair having its nominated flag set. See Section 7.2.1.5 for
the procedure.

7.1.2.3. Check List and Timer State Updates

Regardless of whether the check was successful or failed, the
completion of the transaction may require updating of check list and
timer states.

If all of the pairs in the check list are now either in the Failed or
Succeeded state:

   o 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.

   o For each frozen check list, the agent:

      * Groups together all of the pairs with the same foundation,

      * For each group, sets the state of the pair with the lowest
         component ID to Waiting. If there is more than one such pair,
         the one with the highest priority is used.

   If none of the pairs in the check list are in the Waiting or Frozen
   state, the check list is no longer considered active, and will not
   count towards the value of N in the computation of timers for
   ordinary checks as described in Section 5.8.

7.2. STUN Server Procedures

   An agent MUST be prepared to receive a Binding Request on the base of
   each candidate it included in its most recent offer or answer. This
   requirement holds even if the peer is a lite implementation.

   The agent MUST use a short term credential to authenticate the
   request and perform a message integrity check. The agent MUST
   consider the username to be valid if it consists of two values
   separated by a colon, where the first value is equal to the username
   fragment generated by the agent in an offer or answer for a session
   in-progress. It is possible (and in fact very likely) that an
   offerer will receive a Binding Request prior to receiving the answer
   from its peer. If this happens, the agent MUST immediately generate
   a response (including computation of the mapped address as described
   in Section 7.2.1.2. The agent has sufficient information at this
   point to generate the response; the password from the peer is not
   required. Once the answer is received, it MUST proceed with the
   remaining steps required, namely Section 7.2.1.3, Section 7.2.1.4,
   and Section 7.2.1.5 for full implementations. In cases where
   multiple STUN requests are received before the answer, this may cause
   several pairs to be queued up in the triggered check queue.

   An agent MUST NOT utilize the ALTERNATE-SERVER mechanism, and MUST
   NOT support the backwards compatibility mechanisms to RFC 3489. It
   MUST utilize the FINGERPRINT mechanism.

   If the agent is using Diffserv Codepoint markings [RFC2475] in its
   media packets, it SHOULD apply those same markings to its responses
   to Binding Requests. The same would apply to any layer 2 markings
   the endpoint might be applying to media packets.

7.2.1. Additional Procedures for Full Implementations

   This subsection defines the additional server procedures applicable
   to full implementations.

7.2.1.1. Detecting and Repairing Role Conflicts

   Normally, the rules for selection of a role in Section 5.2 will
   result in each agent selecting a different role - one controlling,
   and one controlled. However, in unusual call flows, typically
   utilizing third party call control, it is possible for both agents to
   select the same role. This section describes procedures for checking
   for this case and repairing it.

   An agent MUST examine the Binding Request for either the ICE-
   CONTROLLING or ICE-CONTROLLED attribute. It MUST follow these
   procedures:

   o If neither ICE-CONTROLLING or ICE-CONTROLLED are present in the
      request, the peer agent may have implemented a previous version of
      this specification. There may be a conflict, but it cannot be
      detected.

   o If the agent is in the controlling role, and the ICE-CONTROLLING
      attribute is present in the request:

      * If the agent's tie-breaker is larger than or equal to the
         contents of the ICE-CONTROLLING attribute, the agent generates
         a Binding Error Response and includes an ERROR-CODE attribute
         with a value of 487 (Role Conflict) but retains its role.

      * If the agent's tie-breaker is less than the contents of the
         ICE-CONTROLLING attribute, the agent switches to the controlled
         role.

   o If the agent is in the controlled role, and the ICE-CONTROLLED
      attribute is present in the request:

      * If the agent's tie-breaker is larger than or equal to the
         contents of the ICE-CONTROLLED attribute, the agent switches to
         the controlling role.

      * If the agent's tie-breaker is less than the contents of the
         ICE-CONTROLLED attribute, the agent generates a Binding Error
         Response and includes an ERROR-CODE attribute with a value of
         487 (Role Conflict) but retains its role.

o If the agent is in the controlled role and the ICE-CONTROLLING
   attribute was present in the request, or the agent was in the
   controlling role and the ICE-CONTROLLED attribute was present in
   the request, there is no conflict.

A change in roles will require an agent to recompute pair priorities
Section 5.7.2, since those priorities are a function of controlling
and controlled role. The change in role will also impact whether the
agent is responsible for selecting nominated pairs and generated
updated offers upon conclusion of ICE.

The remaining sections in Section 7.2.1 are followed if the server
generated a successful response to the Binding Request, even if the
agent changed roles.

7.2.1.2. Computing Mapped Address

For requests being received on a relayed candidate, the source
transport address used for STUN processing (namely, generation of the
XOR-MAPPED-ADDRESS attribute) is the transport address as seen by the
TURN server. That source transport address will be present in the
REMOTE-ADDRESS attribute of a Data Indication message, if the Binding
Request was delivered through a Data Indication (a TURN server
delivers packets encapsulated in a Data Indication when no active
destination is set). If the Binding Request was not encapsulated in
a Data Indication, that source address is equal to the current active
destination for the TURN session.

7.2.1.3. Learning Peer Reflexive Candidates

If the source transport address of the request does not match any
existing remote candidates, it represents a new peer reflexive remote
candidate. This candidate is constructed as follows:

o The priority of the candidate is set to the PRIORITY attribute
   from the request.

o The type of the candidate is set to peer reflexive.

o The foundation of the candidate is set to an arbitrary value,
   different from the foundation for all other remote candidates. If
   any subsequent offer/answer exchanges contain this peer reflexive
   candidate in the SDP, it will signal the actual foundation for the
   candidate.

o The component ID of this candidate is set to the component ID for
   the local candidate to which the request was sent.

This candidate is added to the list of remote candidates. However,
the agent does not pair this candidate with any local candidates.

7.2.1.4. Triggered Checks

Next, the agent constructs a pair whose local candidate is equal to
the transport address on which the STUN request was received, and a
remote candidate equal to the source transport address where the
request came from (which may be peer-reflexive remote candidate that
was just learned). Since both candidates are known to the agent, it
can obtain their priorities and compute the candidate pair priority.
This pair is then looked up in the check list. There can be one of
several outcomes:

o If the pair is already on the check list:

   * If the state of that pair is Waiting or Frozen, a check for
      that pair is enqueued into the triggered check queue.

   * If the state of that pair is In-Progress, the agent cancels the
      in-progress transaction. Cancellation means that the agent
      will not retransmit the request, will not treat the lack of
      response to be a failure, but will wait the duration of the
      transaction timeout for a response. In addition, the agent
      MUST create a new connectivity check for that pair
      (representing a new STUN Binding Request transaction) by
      enqueueing the pair in the triggered check queue. The state of
      the pair is then changed to Waiting.

   * If the state of the pair is Failed, it is changed to Waiting
      and the agent MUST create a new connectivity check for that
      pair (representing a new STUN Binding Request transaction), by
      enqueueing the pair in the triggered check queue.

   * If the state of that pair is Succeeded, nothing further is
      done.

o These steps are done to facilitate rapid completion of ICE when
   both agents are behind NAT.

o If the pair is not already on the check list:

   * The pair is inserted into the check list based on its priority

   * Its state is set to Waiting

   * The pair is enqueued into the triggered check queue.

When a triggered check is to be sent, it is constructed and processed
as described in Section 7.1.1. These procedures require the agent to
know the transport address, username fragment and password for the
peer. The username fragment for the remote candidate is equal to the
part after the colon of the USERNAME in the Binding Request that was
just received. Using that username fragment, the agent can check the
SDP messages received from its peer (there may be more than one in
cases of forking), and find this username fragment. The
corresponding password is then selected.

7.2.1.5. Updating the Nominated Flag

If the Binding Request received by the agent had the USE-CANDIDATE
attribute set, and the agent is in the controlled role, the agent
looks at the state of the pair computed in Section 7.2.1.4:

o If the state of this pair is Succeeded, it means that the check
   generated by this pair produced a successful response. This would
   have caused the agent to construct a valid pair when that success
   response was received (see Section 7.1.2.2.2). The agent now sets
   the nominated flag in the valid pair to true. This may end ICE
   processing for this media stream; see Section 8.

o If the state of this pair is In-Progress, if its check produces a
   successful result, the resulting valid pair has its nominated flag
   set when the response arrives. This may end ICE processing for
   this media stream when it arrives; see Section 8.

7.2.2. Additional Procedures for Lite Implementations

If the check that was just received contained a USE-CANDIDATE
attribute, the agent constructs a candidate pair whose local
candidate is equal to the transport address on which the request was
received, and whose remote candidate is equal to the source transport
address of the request that was received. This candidate pair is
assigned an arbitrary priority, and placed into a list of valid
candidates called the valid list. The agent sets the nominated flag
for that pair to true. ICE processing is considered complete for a
media stream if the valid list contains a candidate pair for each
component.

8. Concluding ICE Processing

This section describes how an agent completes ICE.

8.1. Procedures for Full Implementations

   Concluding ICE involves nominating pairs by the controlling agent and
   updating of state machinery.

8.1.1. Nominating Pairs

   The controlling agent nominates pairs to be selected by ICE by using
   one of two techniques: regular nomination or aggressive nomination.
   If its peer has a lite implementation, an agent MUST use a regular
   nomination algorithm. If its peer is using ICE options (present in
   an ice-options attribute from the peer) that the agent does not
   understand, the agent MUST use a regular nomination algorithm. If
   its peer is a full implementation and isn't using any ICE options or
   is using ICE options understood by the agent, the agent MAY use
   either the aggressive or the regular nomination algorithm. However,
   the regular algorithm is RECOMMENDED since it provides greater
   stability.

8.1.1.1. Regular Nomination

   With regular nomination, the agent lets some number of checks
   complete, each of which omit the USE-CANDIDATE attribute. Once one
   or more checks complete successfully for a component of a media
   stream, valid pairs are generated and added to the valid list. The
   agent lets the checks continue until some stopping criteria is met,
   and then picks amongst the valid pairs based on an evaluation
   criteria. The criteria for stopping the checks and for evaluating
   the valid pairs is entirely a matter of local optimization.

   When the controlling agent selects the valid pair, it repeats the
   check that produced this valid pair (by enqueuing the pair that
   generated the check into the triggered check queue), this time with
   the USE-CANDIDATE attribute. This check should succeed (since the
   previous did), causing the nominated flag of that and only that pair
   to be set. Consequently, there will be only a single nominated pair
   in the valid list for each component, and when the state of the check
   list moves to completed, that exact pair is selected by ICE for
   sending and receiving media for that component.

   Regular nomination provides the most flexibility, since the agent has
   control over the stopping and selection criteria for checks. The
   only requirement is that the agent MUST eventually pick one and only
   one candidate pair and generate a check for that pair with the USE-
   CANDIDATE attribute present. Regular nomination also improves ICE's
   resilience to variations in implementation (see Section 14). Regular
   nomination is also more stable, allowing both agents to converge on a
   single pair for media without any transient selections, which can

happen with the aggressive algorithm. The drawback of regular
nomination is that it is guaranteed to increase latencies because it
requires an additional check to be done.

8.1.1.2. Aggressive Nomination

With aggressive nomination, the controlling agent includes the USE-
CANDIDATE attribute in every check it sends. Once the first check
for a component succeeds, it will be added to the valid list, and
have its nominated flag set. When all components have a nominated
pair in the valid list, it will cause ICE processing to cease for
this check list. However, because the agent included the USE-
CANDIDATE attribute in all of its checks, another check may yet
complete, causing another valid pair to have its nominated flag set.
ICE always selects the highest priority nominated candidate pair from
the valid list as the one used for media. Consequently, the selected
pair may actually change briefly as ICE checks complete, resulting in
a set of transient selections until it stabilizes.

8.1.2. Updating States

For both controlling and controlled agents, the state of ICE
processing depends on the presence of nominated candidate pairs in
the valid list and on the state of the check list. Note that, at any
time, more than one of the following cases can apply:

o If there are no nominated pairs in the valid list for a media
   stream and the state of the check list is Running, ICE processing
   continues.

o If there is at least one nominated pair in the valid list for a
   media stream and the state of the check list is Running:

   * The agent MUST remove all Waiting and Frozen pairs in the check
      list and triggered check queue for the same component as the
      nominated pairs for that media stream

   * If an In-Progress pair in the check list is for the same
      component as a nominated pair, the agent SHOULD cease
      retransmissions for its check if its pair priority is lower
      than the lowest priority nominated pair for that component

o Once there is at least one nominated pair in the valid list for
   every component of at least one media stream and the state of the
   check list is Running:

   * The agent MUST change the state of processing for its check
      list for that media stream to Completed.

* The agent MUST continue to respond to any checks it may still receive for that media stream, and MUST perform triggered checks if required by the processing of Section 7.2.

* The agent MAY begin transmitting media for this media stream as described in Section 11.1

o Once the state of each check list is Completed:

* The agent sets the state of ICE processing overall to Completed.

* If an agent is controlling, it examines the highest priority nominated candidate pair for each component of each media stream. If any of those candidate pairs differ from the default candidate pairs in the most recent offer/answer exchange, the controlling agent MUST generate an updated offer as described in Section 9. If the controlling agent is using an aggressive nomination algorithm, this may result in several updated offers as the pairs selected for media change. An agent MAY delay sending the offer for a brief interval (one second is RECOMMENDED) in order to allow the selected pairs to stabilize.

o If the state of the check list is Failed, ICE has not been able to complete for this media stream. The correct behavior depends on the state of the check lists for other media streams:

* If all check lists are Failed, ICE processing overall is considered to be in the Failed state, and the agent SHOULD consider the session a failure, SHOULD NOT restart ICE, and the controlling agent SHOULD terminate the entire session.

* If at least one of the check lists for other media streams is Completed, the controlling agent SHOULD remove the failed media stream from the session in its updated offer.

* If none of the check lists for other media streams are Completed, but at least one is Running, the agent SHOULD let ICE continue.

8.2. Procedures for Lite Implementations

Concluding ICE for a lite implementation is relatively straightforward. There are two cases to consider:

The implementation is lite, and its peer is full.

      The implementation is lite, and its peer is lite.

   The effect of ICE concluding is that the agent can free any allocated
   host candidates that were not utilized by ICE, as described in
   Section 8.3.

8.2.1. Peer is Full

   In this case, the agent will receive connectivity checks from its
   peer. When an agent has received a connectivity check that includes
   the USE-CANDIDATE attribute for each component of a media stream, the
   state of ICE processing for that media stream moves from Running to
   Completed. When the state of ICE processing for all media streams is
   Completed, the state of ICE processing overall is Completed.

   The lite implementation will never itself determine that ICE
   processing has failed for a media stream; rather, the full peer will
   make that determination and then remove or restart the failed media
   stream in a subsequent offer.

8.2.2. Peer is Lite

   Once the offer/answer exchange has completed, both agents examine
   their candidates and those of its peer. For each media stream, each
   agent pairs up its own candidates with the candidates of its peer for
   that media stream. Two candidates are paired up when they are for
   the same component, utilize the same transport protocol (UDP in this
   specification), and are from the same IP address family (IPv4 or
   IPv6).

   o If there is a single pair per component, that pair is added to the
      Valid list. If all of the components for a media stream had one
      pair, the state of ICE processing for that media stream is set to
      Completed. If all media streams are Completed, the state of ICE
      processing is set to Completed overall. This will always be the
      case for implementations that are IPv4 only.

   o If there is more than one pair per component:

      * The agent MUST select a pair based on local policy. Since this
         case only arises for IPv6, it is RECOMMENDED that an agent
         follow the procedures of RFC 3484 [RFC3484] to select a single
         pair.

      * The agent adds the selected pair for each component to the
         valid list. As described in Section 11.1, this will permit
         media to begin flowing. However, it is possible (and in fact
         likely) that both agents have chosen different pairs.

   * To reconcile this, the controlling agent MUST send an updated
      offer as described in Section 9.1.3, which will include the
      remote-candidates attribute.

   * The agent MUST NOT update the state of ICE processing when the
      offer is sent. If this subsequent offer completes, the
      controlling agent MUST change the state of ICE processing to
      Completed for all media streams, and the state of ICE
      processing overall to Completed. The states for the controlled
      agent are set based on the logic in Section 9.2.3.

8.3. Freeing Candidates

8.3.1. Full Implementation Procedures

   The procedures in Section 8 require that an agent continue to listen
   for STUN requests and continue to generate triggered checks for a
   media stream, even once processing for that stream completes. The
   rules in this section describe when it is safe for an agent to cease
   sending or receiving checks on a candidate that was not selected by
   ICE, and then free the candidate.

   When ICE is used with SIP, and an offer is forked to multiple
   recipients, ICE proceeds in parallel and independently with each
   answerer, all using the same local candidates. Once ICE processing
   has reached the Completed state for all peers for media streams using
   those candidates, the agent SHOULD wait an additional three seconds,
   and then it MAY cease responding to checks or generating triggered
   checks on that candidate. It MAY free the candidate at that time.
   Freeing of server reflexive candidates is never explicit; it happens
   by lack of a keepalive. The three second delay handles cases when
   aggressive nomination is used, and the selected pairs can quickly
   change after ICE has completed.

8.3.2. Lite Implementations

   A lite implementation MAY free candidates not selected by ICE as soon
   as ICE processing has reached the completed state for all peers for
   all media streams using those candidates.

9. Subsequent Offer/Answer Exchanges

   Either agent MAY generate a subsequent offer at any time allowed by
   RFC 3264 [RFC3264]. The rules in Section 8 will cause the
   controlling agent to send an updated offer at the conclusion of ICE
   processing when ICE has selected different candidate pairs from the
   default pairs. This section defines rules for construction of

subsequent offers and answers.

Should a subsequent offer be rejected, ICE processing continues as if
the subsequent offer had never been made.

9.1. Generating the Offer

9.1.1. Procedures for All Implementations

9.1.1.1. ICE Restarts

An agent MAY restart ICE processing for an existing media stream. An
ICE restart, as the name implies, will cause all previous state of
ICE processing to be flushed and checks to start anew. The only
difference between an ICE restart and a brand new media session is
that, during the restart, media can continue to be sent to the
previously validated pair.

An agent MUST restart ICE for a media stream if:

o The offer is being generated for the purposes of changing the
   target of the media stream. In other words, if an agent wants to
   generated an updated offer which, had ICE not been in use, would
   result in a new value for the destination of a media component.

o An agent is changing its implementation level. This typically
   only happens in third party call control use cases, where the
   entity performing the signaling is not the entity receiving the
   media, and it has changed the target of media mid-session to
   another entity that has a different ICE implementation.

These rules imply that setting the IP address in the c line to
0.0.0.0 will cause an ICE restart. Consequently, ICE implementations
MUST NOT utilize this mechanism for call hold, and instead MUST use
a=inactive and a=sendonly as described in [RFC3264]

To restart ICE, an agent MUST change both the ice-pwd and the ice-
ufrag for the media stream in an offer. Note that it is permissible
to use a session-level attribute in one offer, but to provide the
same ice-pwd or ice-ufrag as a media-level attribute in a subsequent
offer. This is not a change in password, just a change in its
representation, and does not cause an ICE restart.

An agent sets the rest of the fields in the SDP for this media stream
as it would in an initial offer of this media stream (see
Section 4.3). Consequently, the set of candidates MAY include some,
none, or all of the previous candidates for that stream and MAY
include a totally new set of candidates gathered as described in

Section 4.1.1.

9.1.1.2. Removing a Media Stream

   If an agent removes a media stream by setting its port to zero, it
   MUST NOT include any candidate attributes for that media stream and
   SHOULD NOT include any other ICE-related attributes defined in
   Section 15 for that media stream.

9.1.1.3. Adding a Media Stream

   If an agent wishes to add a new media stream, it sets the fields in
   the SDP for this media stream as if this was an initial offer for
   that media stream (see Section 4.3). This will cause ICE processing
   to begin for this media stream.

9.1.2. Procedures for Full Implementations

   This section describes additional procedures for full
   implementations, covering existing media streams.

   The username fragments, password, and implementation level MUST
   remain the same as used previously. If an agent needs to change one
   of these it MUST restart ICE for that media stream.

   Additional behavior depends on the state ICE processing for that
   media stream.

9.1.2.1. Existing Media Streams with ICE Running

   If an agent generates an updated offer including media stream that
   was previously established, and for which ICE checks are in the
   Running state, the agent follows the procedures defined here.

   An agent MUST include candidate attributes for all local candidates
   it had signaled previously for that media stream. The properties of
   that candidate as signaled in SDP - the priority, foundation, type
   and related transport address SHOULD remain the same. The IP
   address, port and transport protocol, which fundamentally identify
   that candidate, MUST remain the same (if they change, it would be a
   new candidate). The component ID MUST remain the same. The agent
   MAY include additional candidates it did not offer previously, but
   which it has gathered since the last offer/answer exchange, including
   peer reflexive candidates.

   The agent MAY change the default destination for media. As with
   initial offers, there MUST be a set of candidate attributes in the
   offer matching this default destination.

9.1.2.2. Existing Media Streams with ICE Completed

If an agent generates an updated offer including media stream that
was previously established, and for which ICE checks are in the
Completed state, the agent follows the procedures defined here.

The default destination for media (i.e., the values of the IP
addresses and ports in the m and c line used for that media stream)
MUST be the local candidate from the highest priority nominated pair
in the valid list for each component. This "fixes" the default
destination for media to equal the destination ICE has selected for
media.

The agent MUST include a candidate attributes for candidates matching
the default destination for each component of the media stream, and
MUST NOT include any other candidates.

In addition, if the agent is controlling, it MUST include the
a=remote-candidates attribute for each media stream whose check list
is in the Completed state. The attribute contains the remote
candidates from the highest priority nominated pair in the valid list
for each component of that media stream. It is needed to avoid a
race condition whereby the controlling agent chooses its pairs, but
the updated offer beats the connectivity checks to the controlled
agent, which doesn't even know these pairs are valid, let alone
selected. See Appendix B.6 for elaboration on this race condition.

9.1.3. Procedures for Lite Implementations

9.1.3.1. Existing Media Streams with ICE Running

This section describes procedures for lite implementations for
existing streams for which ICE is running.

A lite implementation MUST include all of its candidates for each
component of each media stream in an a=candidate attribute in any
subsequent offer. These candidates are formed identically to the
procedures for initial offers, as described in Section 4.2.

A lite implementation MUST NOT add additional host candidates in a
subsequent offer. If an agent needs to offer additional candidates,
it MUST restart ICE.

The username fragments, password, and implementation level MUST
remain the same as used previously. If an agent needs to change one
of these it MUST restart ICE for that media stream.

9.1.3.2. Existing Media Streams with ICE Completed

   If ICE has completed for a media stream, the default destination for
   that media stream MUST be set to the remote candidate of the
   candidate pair for that component in the valid list. For a lite
   implementation, there is always just a single candidate pair in the
   valid list for each component of a media stream. Additionally, the
   agent MUST include a candidate attribute for each default
   destination.

   Additionally, if the agent is controlling (which only happens when
   both agents are lite), the agent MUST include the a=remote-candidates
   attribute for each media stream. The attribute contains the remote
   candidates from the candidate pairs in the valid list (one pair for
   each component of each media stream).

9.2. Receiving the Offer and Generating an Answer

9.2.1. Procedures for All Implementations

   When receiving a subsequent offer within an existing session, an
   agent MUST re-apply the verification procedures in Section 5.1
   without regard to the results of verification from any previous
   offer/answer exchanges. Indeed, it is possible that a previous
   offer/answer exchange resulted in ICE not being used, but it is used
   as a consequence of a subsequent exchange.

9.2.1.1. Detecting ICE Restart

   If the offer contained a change in the a=ice-ufrag or a=ice-pwd
   attributes compared to the previous SDP from the peer, it indicates
   that ICE is restarting for this media stream. If all media streams
   are restarting, than ICE is restarting overall.

   If ICE is restarting for a media stream:

   o The agent MUST change the a=ice-ufrag and a=ice-pwd attributes in
      the answer.

   o The agent MAY change its implementation level in the answer.

   An agent sets the rest of the fields in the SDP for this media stream
   as it would in an initial answer to this media stream (see
   Section 4.3). Consequently, the set of candidates MAY include some,
   none, or all of the previous candidates for that stream and MAY
   include a totally new set of candidates gathered as described in
   Section 4.1.1.

9.2.1.2. New Media Stream

   If the offer contains a new media stream, the agent sets the fields
   in the answer as if it had received an initial offer containing that
   media stream (see Section 4.3). This will cause ICE processing to
   begin for this media stream.

9.2.1.3. Removed Media Stream

   If an offer contains a media stream whose port is zero, the agent
   MUST NOT include any candidate attributes for that media stream in
   its answer and SHOULD NOT include any other ICE-related attributes
   defined in Section 15 for that media stream.

9.2.2. Procedures for Full Implementations

   Unless the agent has detected an ICE restart from the offer, the
   username fragments, password, and implementation level MUST remain
   the same as used previously. If an agent needs to change one of
   these it MUST restart ICE for that media stream by generating an
   offer; ICE cannot be restarted in an answer.

   Additional behaviors depend on the state of ICE processing for that
   media stream.

9.2.2.1. Existing Media Streams with ICE Running and no remote-candidates

   If ICE is running for a media stream, and the offer for that media
   stream lacked the remote-candidates attribute, the rules for
   construction of the answer are identical to those for the offerer as
   described in Section 9.1.2.1.

9.2.2.2. Existing Media Streams with ICE Completed and no remote-candidates

   If ICE is Completed for a media stream, and the offer for that media
   stream lacked the remote-candidates attribute, the rules for
   construction of the answer are identical to those for the offerer as
   described in Section 9.1.2.2, except that the answerer MUST NOT
   include the a=remote-candidates attribute in the answer.

9.2.2.3. Existing Media Streams and remote-candidates

   A controlled agent will receive an offer with the a=remote-candidates
   attribute for a media stream when its peer has concluded ICE
   processing for that media stream. This attribute is present in the
   offer to deal with a race condition between the receipt of the offer,

and the receipt of the Binding Response which tells the answerer the
candidate which will be selected by ICE. See Appendix B.6 for an
explanation of this race condition. Consequently, processing of an
offer with this attribute depends on the winner of the race.

The agent forms a candidate pair for each component of the media
stream by:

o Setting the remote candidate equal to the offerers default
   destination for that component (e.g., the contents of the m and
   c-lines for RTP, and the a=rtcp attribute for RTCP)

o Setting the local candidate equal to the transport address for
   that same component in the a=remote-candidates attribute in the
   offer.

The agent then sees if each of these candidate pairs are present in
the valid list. If a particular pair is not in the valid list, the
check has "lost" the race. Call such a pair a "losing pair".

The agent finds all the pairs in the check list whose remote
candidates equal the remote candidate in the losing pair:

o If none of the pairs are In-Progress, and at least one is Failed,
   it is most likely that a network failure, such as a network
   partition or serious packet loss, has occurred. The agent SHOULD
   generate an answer for this media stream as if the remote-
   candidates attribute had not been present, and then restart ICE
   for this stream.

o If at least one of the pairs are In-Progress, the agent SHOULD
   wait for those checks to complete, and as each completes, redo the
   processing in this section until there are no losing pairs.

Once there are no losing pairs, the agent can generate the answer.
It MUST set the default destination for media to the candidates in
the remote-candidates attribute from the offer (each of which will
now be the local candidate of a candidate pair in the valid list).
It MUST include a candidate attribute in the answer for each
candidate in the remote-candidates attribute in the offer.

9.2.3. Procedures for Lite Implementations

If the received offer contains the remote-candidates attribute for a
media stream, the agent forms a candidate pair for each component of
the media stream by:

o Setting the remote candidate equal to the offerers default
   destination for that component (e.g., the contents of the m and
   c-lines for RTP, and the a=rtcp attribute for RTCP)

o Setting the local candidate equal to the transport address for
   that same component in the a=remote-candidates attribute in the
   offer.

It then places those candidates into the Valid list for the media
stream. The state of ICE processing for that media stream is set to
Completed.

Furthermore, if the agent believed it was controlling, but the offer
contained the remote-candidates attribute, both agents believe they
are controlling. In this case, both would have sent updated offers
around the same time. However, the signaling protocol carrying the
offer/answer exchanges will have resolved this glare condition, so
that one agent is always the 'winner' by having its offer received
before its peer has sent an offer. The winner takes the role of
controlled, so that the loser (the answerer under consideration in
this section MUST change its role to controlled. Consequently, if
the agent was going to send an updated offer since, based on the
rules in Section 8.2.2, it was controlling, it no longer needs to.

Besides the potential role change, change in the Valid list, and
state changes, the construction of the answer is performed
identically to the construction of an offer as described in
Section 9.1.3.

9.3. Updating the Check and Valid Lists

9.3.1. Procedures for Full Implementations

9.3.1.1. ICE Restarts

The agent MUST remember the highest priority nominated pairs in the
Valid list for each component of the media stream, called the
previous selected pairs, prior to the restart. The agent will
continue to send media using these pairs, as described in
Section 11.1. Once these destinations are noted, the agent MUST
flush the valid and check lists, and then recompute the check list
and its states as described in Section 5.7.

9.3.1.2. New Media Stream

If the offer/answer exchange added a new media stream, the agent MUST
create a new check list for it (and an empty Valid list to start of
course), as described in Section 5.7.

9.3.1.3. Removed Media Stream

   If the offer/answer exchange removed a media stream, or an answer
   rejected an offered media stream, an agent MUST flush the Valid list
   for that media stream. It MUST terminate any STUN transactions in
   progress for that media stream. An agent MUST remove the check list
   for that media stream and cancel any pending ordinary checks for it.

9.3.1.4. ICE Continuing for Existing Media Stream

   The valid list is not affected by an updated offer/answer exchange
   unless ICE is restarting.

   If an agent is in the Running state for that media stream, the check
   list is updated (the check list is irrelevant if the state is
   completed). To do that, the agent recomputes the check list using
   the procedures described in Section 5.7. If a pair on the new check
   list was also on the previous check list, and its state was Waiting,
   In-Progress, Succeeded or Failed, its state is copied over.
   Otherwise, its state is set to Frozen.

   If none of the check lists are active (meaning that the pairs in each
   check list are Frozen), the full-mode agent sets the first pair in
   the check list for the first media stream to Waiting, and then sets
   the state of all other pairs in that check list for the same
   component ID and with the same foundation to Waiting as well.

   Next, the agent goes through each check list, starting with the
   highest priority pair. If a pair has a state of Succeeded, and it
   has a component ID of 1, then all Frozen pairs in the same check list
   with the same foundation whose component IDs are not 1, have their
   state set to Waiting. If, for a particular check list, there are
   pairs for each component of that media stream in the Succeeded state,
   the agent moves the state of all Frozen pairs for the first component
   of all other media streams (and thus in different check lists) with
   the same foundation to Waiting.

9.3.2. Procedures for Lite Implementations

   If ICE is restarting for a media stream, the agent MUST start a new
   Valid list for that media stream. It MUST remember the pairs in the
   previous Valid list for each component of the media stream, called
   the previous selected pairs, and continue to send media there as
   described in Section 11.1. The state of ICE processing for each
   media stream MUST change to Running, and the state of ICE processing
   MUST change to running.

10. Keepalives

   All endpoints MUST send keepalives for each media session. These
   keepalives serve the purpose of keeping NAT bindings alive for the
   media session. These keepalives MUST be sent regardless of whether
   the media stream is currently inactive, sendonly, recvonly or
   sendrecv, and regardless of the presence or value of the bandwidth
   attribute. These keepalives MUST be sent even if ICE is not being
   utilized for the session at all. The keepalive SHOULD be sent using
   a format which is supported by its peer. ICE endpoints allow for
   STUN-based keepalives for UDP streams, and as such, STUN keepalives
   MUST be used when an agent is a full ICE implementation and is
   communicating with a peer that supports ICE (lite or full). An agent
   can determine that its peer supports ICE by the presence of
   a=candidate attributes for each media session. If the peer does not
   support ICE, the choice of a packet format for keepalives is a matter
   of local implementation. A format which allows packets to easily be
   sent in the absence of actual media content is RECOMMENDED. Examples
   of formats which readily meet this goal are RTP No-Op
   [I-D.ietf-avt-rtp-no-op], and in cases where both sides support it,
   RTP comfort noise [RFC3389]. If the peer doesn't support any formats
   that are particularly well suited for keepalives, an agent SHOULD
   send RTP packets with an incorrect version number, or some other form
   of error which would cause them to be discarded by the peer.

   If there has been no packet sent on the candidate pair ICE is using
   for a media component for Tr seconds (where packets include those
   defined for the component (RTP or RTCP) and previous keepalives), an
   agent MUST generate a keepalive on that pair. Tr SHOULD be
   configurable and SHOULD have a default of 15 seconds. Alternatively,
   if an agent has a dynamic way to discover the binding lifetimes of
   the intervening NATs, it can use that value to determine Tr.

   If STUN is being used for keepalives, a STUN Binding Indication is
   used [I-D.ietf-behave-rfc3489bis]. The Indication MUST NOT utilize
   any authentication mechanism, and SHOULD NOT contain any attributes.
   It is used solely to keep the NAT bindings alive. The Binding
   Indication is sent using the same local and remote candidates that
   are being used for media. Though Binding Indications are used for
   keepalives, an agent MUST be prepared to receive a connectivity check
   as well. If a connectivity check is received, a response is
   generated as discussed in [I-D.ietf-behave-rfc3489bis], but there is
   no impact on ICE processing otherwise.

   An agent MUST begin the keepalive processing once ICE has selected
   candidates for usage with media, or media begins to flow, whichever
   happens first. Keepalives end once the session terminates or the
   media stream is removed.

11. **Media Handling**

11.1. **Sending Media**

   Procedures for sending media differ for full and lite
   implementations.

11.1.1. **Procedures for Full Implementations**

   Agents always send media using a candidate pair, called the selected
   candidate pair. An agent will send media to the remote candidate in
   the selected pair (setting the destination address and port of the
   packet equal to that remote candidate), and will send it from the
   local candidate of the selected pair. When the local candidate is
   server or peer reflexive, media is originated from the base. Media
   sent from a relayed candidate is sent from the base through that TURN
   server, using procedures defined in [I-D.ietf-behave-turn].

   The selected pair for a component of a media stream is:

   o empty if the state of the check list for that media stream is
      Running, and there is no previous selected pair for that component
      due to an ICE restart

   o equal to the previous selected pair for a component of a media
      stream if the state of the check list for that media stream is
      Running, and there was a previous selected pair for that component
      due to an ICE restart

   o equal to the highest priority nominated pair for that component in
      the valid list if the state of the check list is Completed

   If the selected pair for at least one component of a media stream is
   empty, an agent MUST NOT send media for any component of that media
   stream. If the selected pair for each component of a media stream
   has a value, an agent MAY send media for all components of that media
   stream.

   Note that the selected pair for a component of a media stream may not
   equal the default pair for that same component from the most recent
   offer/answer exchange. When this happens, the selected pair is used
   for media, not the default pair. When ICE first completes, if the
   selected pairs aren't a match for the default pairs, the controlling
   agent sends an updated offer/answer exchange to remedy this
   disparity. However, until that updated offer arrives, there will not
   be a match. Furthermore, in very unusual cases, the default
   candidates in the updated offer/answer will not be a match.

11.1.2. Procedures for Lite Implementations

   A lite implementation MUST NOT send media until it has a Valid list
   that contains a candidate pair for each component of that media
   stream. Once that happens, the agent MAY begin sending media
   packets. To do that, it sends media to the remote candidate in the
   pair (setting the destination address and port of the packet equal to
   that remote candidate), and will send it from the local candidate.

11.1.3. Procedures for All Implementations

   ICE has interactions with jitter buffer adaptation mechanisms. An
   RTP stream can begin using one candidate, and switch to another one,
   though this happens rarely with ICE. The newer candidate may result
   in RTP packets taking a different path through the network - one with
   different delay characteristics. As discussed below, agents are
   encouraged to re-adjust jitter buffers when there are changes in
   source or destination address of media packets. Furthermore, many
   audio codecs use the marker bit to signal the beginning of a
   talkspurt, for the purposes of jitter buffer adaptation. For such
   codecs, it is RECOMMENDED that the sender set the marker bit
   [RFC3550] when an agent switches transmission of media from one
   candidate pair to another.

11.2. Receiving Media

   ICE implementations MUST be prepared to receive media on each
   component on any candidates provided for that component in the most
   recent offer/answer exchange (in the case of RTP, this would include
   both RTP and RTCP if candidates were provided for both).

   It is RECOMMENDED that, when an agent receives an RTP packet with a
   new source or destination IP address for a particular media stream,
   that the agent re-adjust its jitter buffers.

   RFC 3550 [RFC3550] describes an algorithm in Section 8.2 for
   detecting SSRC collisions and loops. These algorithms are based, in
   part, on seeing different source transport addresses with the same
   SSRC. However, when ICE is used, such changes will sometimes occur
   as the media streams switch between candidates. An agent will be
   able to determine that a media stream is from the same peer as a
   consequence of the STUN exchange that proceeds media transmission.
   Thus, if there is a change in source transport address, but the media
   packets come from the same peer agent, this SHOULD NOT be treated as
   an SSRC collision.

12. Usage with SIP

12.1. Latency Guidelines

   ICE requires a series of STUN-based connectivity checks to take place
   between endpoints. These checks start from the answerer on
   generation of its answer, and start from the offerer when it receives
   the answer. These checks can take time to complete, and as such, the
   selection of messages to use with offers and answers can effect
   perceived user latency. Two latency figures are of particular
   interest. These are the post-pickup delay and the post-dial delay.
   The post-pickup delay refers to the time between when a user "answers
   the phone" and when any speech they utter can be delivered to the
   caller. The post-dial delay refers to the time between when a user
   enters the destination address for the user, and ringback begins as a
   consequence of having successfully started ringing the phone of the
   called party.

   Two cases can be considered - one where the offer is present in the
   initial INVITE, and one where it is in a response.

12.1.1. Offer in INVITE

   To reduce post-dial delays, it is RECOMMENDED that the caller begin
   gathering candidates prior to actually sending its initial INVITE.
   This can be started upon user interface cues that a call is pending,
   such as activity on a keypad or the phone going offhook.

   If an offer is received in an INVITE request, the answerer SHOULD
   begin to gather its candidates on receipt of the offer and then
   generate an answer in a provisional response once it has completed
   that process. ICE requires that a provisional response with an SDP
   be transmitted reliably. This can be done through the existing PRACK
   mechanism [RFC3262], or through an optimization that is specific to
   ICE. With this optimization, provisional responses containing an SDP
   answer that begins ICE processing for one or more media streams can
   be sent reliably without RFC 3262. To do this, the agent retransmits
   the provisional response with the exponential backoff timers
   described in RFC 3262. Retransmits MUST cease on receipt of a STUN
   Binding Request for one of the media streams signaled in that SDP
   (because receipt of a binding request indicates the offerer has
   received the answer) or on transmission of the answer in a 2xx
   response. If the peer agent is lite, there will never be a STUN
   Binding Request. In such a case, the agent MUST cease retransmitting
   the 18x after sending it four times (ICE will actually work even if
   the peer never receives the 18x; however, experience has shown that
   sending it is important for middleboxes and firewall traversal). If
   no Binding Request is received prior to the last retransmit, the

agent does not consider the session terminated. Despite the fact
that the provisional response will be delivered reliably, the rules
for when an agent can send an updated offer or answer do not change
from those specified in RFC 3262. Specifically, if the INVITE
contained an offer, the same answer appears in all of the 1xx and in
the 2xx response to the INVITE. Only after that 2xx has been sent
can an updated offer/answer exchange occur. This optimization SHOULD
NOT be used if both agents support PRACK. Note that the optimization
is very specific to provisional response carrying answers that start
ICE processing; it is not a general technique for 1xx reliability.

Alternatively, an agent MAY delay sending an answer until the 200 OK,
however this results in a poor user experience and is NOT
RECOMMENDED.

Once the answer has been sent, the agent SHOULD begin its
connectivity checks. Once candidate pairs for each component of a
media stream enter the valid list, the answerer can begin sending
media on that media stream.

However, prior to this point, any media that needs to be sent towards
the caller (such as SIP early media [RFC3960] MUST NOT be
transmitted. For this reason, implementations SHOULD delay alerting
the called party until candidates for each component of each media
stream have entered the valid list. In the case of a PSTN gateway,
this would mean that the setup message into the PSTN is delayed until
this point. Doing this increases the post-dial delay, but has the
effect of eliminating 'ghost rings'. Ghost rings are cases where the
called party hears the phone ring, picks up, but hears nothing and
cannot be heard. This technique works without requiring support for,
or usage of, preconditions [RFC3312], since its a localized decision.
It also has the benefit of guaranteeing that not a single packet of
media will get clipped, so that post-pickup delay is zero. If an
agent chooses to delay local alerting in this way, it SHOULD generate
a 180 response once alerting begins.

12.1.2. Offer in Response

In addition to uses where the offer is in an INVITE, and the answer
is in the provisional and/or 200 OK response, ICE works with cases
where the offer appears in the response. In such cases, which are
common in third party call control [RFC3725], ICE agents SHOULD
generate their offers in a reliable provisional response (which MUST
utilize RFC 3262), and not alert the user on receipt of the INVITE.
The answer will arrive in a PRACK. This allows for ICE processing to
take place prior to alerting, so that there is no post-pickup delay,
at the expense of increased call setup delays. Once ICE completes,
the callee can alert the user and then generate a 200 OK when they

answer. The 200 OK would contain no SDP, since the offer/answer
exchange has completed.

Alternatively, agents MAY place the offer in a 2xx instead (in which
case the answer comes in the ACK). When this happens, the callee
will alert the user on receipt of the INVITE, and the ICE exchanges
will take place only after the user answers. This has the effect of
reducing call setup delay, but can cause substantial post-pickup
delays and media clipping.

12.2. SIP Option Tags and Media Feature Tags

[I-D.ietf-sip-ice-option-tag] specifies a SIP option tag and media
feature tag for usage with ICE. ICE implementations using SIP SHOULD
support this specification, which uses a feature tag in registrations
to facilitate interoperability through signaling intermediaries

12.3. Interactions with Forking

ICE interacts very well with forking. Indeed, ICE fixes some of the
problems associated with forking. Without ICE, when a call forks and
the caller receives multiple incoming media streams, it cannot
determine which media stream corresponds to which callee.

With ICE, this problem is resolved. The connectivity checks which
occur prior to transmission of media carry username fragments, which
in turn are correlated to a specific callee. Subsequent media
packets which arrive on the same candidate pair as the connectivity
check will be associated with that same callee. Thus, the caller can
perform this correlation as long as it has received an answer.

12.4. Interactions with Preconditions

Quality of Service (QoS) preconditions, which are defined in RFC 3312
[RFC3312] and RFC 4032 [RFC4032], apply only to the transport
addresses listed as the default targets for media in an offer/answer.
If ICE changes the transport address where media is received, this
change is reflected in an updated offer which changes the default
destination for media to match ICE's selection. As such, it appears
like any other re-INVITE would, and is fully treated in RFC 3312 and
4032, which apply without regard to the fact that the destination for
media is changing due to ICE negotiations occurring "in the
background".

Indeed, an agent SHOULD NOT indicate that Qos preconditions have been
met until the checks have completed and selected the candidate pairs
to be used for media.

ICE also has (purposeful) interactions with connectivity
preconditions [I-D.ietf-mmusic-connectivity-precon].   Those
interactions are described there.   Note that the procedures described
in Section 12.1 describe their own type of "preconditions", albeit
with less functionality than those provided by the explicit
preconditions in [I-D.ietf-mmusic-connectivity-precon].

12.5. Interactions with Third Party Call Control

ICE works with Flows I, III and IV as described in [RFC3725].   Flow I
works without the controller supporting or being aware of ICE.   Flow
IV will work as long as the controller passes along the ICE
attributes without alteration.   Flow II is fundamentally incompatible
with ICE; each agent will believe itself to be the answerer and thus
never generate a re-INVITE.

The flows for continued operation, as described in Section 7 of RFC
3725, require additional behavior of ICE implementations to support.
In particular, if an agent receives a mid-dialog re-INVITE that
contains no offer, it MUST restart ICE for each media stream and go
through the process of gathering new candidates.   Furthermore, that
list of candidates SHOULD include the ones currently being used for
media.

13. Relationship with ANAT

RFC 4091 [RFC4091] defines a mechanism for indicating that an agent
can support both IPv4 and IPv6 for a media stream, and it does so by
including two m-lines, one for v4, and one for v6.   This is similar
to ICE, which allows for an agent to indicate multiple transport
addresses using the candidate attribute.   However, ANAT relies on
static selection to pick between choices, rather than a dynamic
connectivity check used by ICE.

This specification deprecates RFC 4091.   Instead, agents wishing to
support dual-stack will utilize ICE.   Because a dual-stack agent will
require at least two candidates, one for IPv4 and one for IPv6, dual-
stack agents MUST be full implementations.   However, agents that are
implementing dual-stack and are running on closed networks where it
is known that there are no NAT, MAY include only host candidates in
their offers, skipping server reflexive and relayed candidates.

14. Extensibility Considerations

This specification makes very specific choices about how both agents
in a session coordinate to arrive at the set of candidate pairs that

are selected for media. It is anticipated that future specifications
will want to alter these algorithms, whether they are simple changes
like timer tweaks, or larger changes like a revamp of the priority
algorithm. When such a change is made, providing interoperability
between the two agents in a session is critical.

First, ICE provides the a=ice-options SDP attribute. Each extension
or change to ICE is associated with a token. When an agent
supporting such an extension or change generates an offer or an
answer, it MUST include the token for that extension in this
attribute. This allows each side to know what the other side is
doing. This attribute MUST NOT be present if the agent doesn't
support any ICE extensions or changes.

At this time, no IANA registry or registration procedures are defined
for these option tags. At time of writing, it is unclear whether ICE
changes and extensions will be sufficiently common to warrrant a
registry.

One of the complications in achieving interoperability is that ICE
relies on a distributed algorithm running on both agents to converge
on an agreed set of candidate pairs. If the two agents run different
algorithms, it can be difficult to guarantee convergence on the same
candidate pairs. The regular nomination procedure described in
Section 8 eliminates some of the tight coordination by delegating the
selection algorithm completely to the controlling agent.
Consequently, when a controlling agent is communicating with a peer
that supports options it doesn't know about, the agent MUST run a
regular nomination algorithm. When regular nomination is used, ICE
will converge perfectly even when both agents use different pair
prioritization algorithms. One of the keys to such convergence are
triggered checks, which ensure that the nominated pair is validated
by both agents. Consequently, any future ICE enhancements MUST
preserve triggered checks.

ICE is also extensible to other media streams beyond RTP, and for
transport protocols beyond UDP. Extensions to ICE for non-RTP media
streams need to specify how many components they utilize, and assign
component IDs to them, starting at 1 for the most important component
ID. Specifications for new transport protocols must define how, if
at all, various steps in the ICE processing differ from UDP.

15. Grammar

This specification defines seven new SDP attributes - the
"candidate", "remote-candidates", "ice-lite", "ice-mismatch", "ice-
ufrag", "ice-pwd" and "ice-options" attributes.

15.1. "candidate" Attribute

   The candidate attribute is a media-level attribute only. It contains
   a transport address for a candidate that can be used for connectivity
   checks.

   The syntax of this attribute is defined using Augmented BNF as
   defined in RFC 4234 [RFC4234]:

   candidate-attribute   = "candidate" ":" foundation SP component-id SP
                            transport SP
                            priority SP
                            connection-address SP   ;from RFC 4566
                            port   ;port from RFC 4566
                            SP cand-type
                            [SP rel-addr]
                            [SP rel-port]
                            *(SP extension-att-name SP
                                extension-att-value)

   foundation   = 1*32ice-char
   component-id   = 1*5DIGIT
   transport   = "UDP" / transport-extension
   transport-extension   = token   ; from RFC 3261
   priority   = 1*10DIGIT
   cand-type   = "typ" SP candidate-types
   candidate-types   = "host" / "srflx" / "prflx" / "relay" / token
   rel-addr   = "raddr" SP connection-address
   rel-port   = "rport" SP port
   extension-att-name   = byte-string   ;from RFC 4566
   extension-att-value   = byte-string
   ice-char   = ALPHA / DIGIT / "+" / "/"

   This grammar encodes the primary information about a candidate: its
   IP address, port and transport protocol, and its properties: the
   foundation, component ID, priority, type, and related transport
   address:

   <connection-address>: is taken from RFC 4566 [RFC4566]. It is the
      IP address of the candidate, allowing for IPv4 addresses, IPv6
      addresses and FQDNs. An IP address SHOULD be used, but an FQDN
      MAY be used in place of an IP address. In that case, when
      receiving an offer or answer containing an FQDN in an a=candidate
      attribute, the FQDN is looked up in the DNS first using an AAAA
      record (assuming the agent supports IPv6), and if no result is
      found or the agent only supports IPv4, using an A. If the DNS

query returns more than one IP address, one is chosen, and then
used for the remainder of ICE processing.

<port>: is also taken from RFC 4566 [RFC4566]. It is the port of
the candidate.

<transport>: indicates the transport protocol for the candidate.
This specification only defines UDP. However, extensibility is
provided to allow for future transport protocols to be used with
ICE, such as TCP or the Datagram Congestion Control Protocol
(DCCP) [RFC4340].

<foundation>: is composed of one to thirty two <ice-char>. It is an
identifier that is equivalent for two candidates that are of the
same type, share the same base, and come from the same STUN
server. The foundation is used to optimize ICE performance in the
Frozen algorithm.

<component-id>: is a positive integer between 1 and 256 which
identifies the specific component of the media stream for which
this is a candidate. It MUST start at 1 and MUST increment by 1
for each component of a particular candidate. For media streams
based on RTP, candidates for the actual RTP media MUST have a
component ID of 1, and candidates for RTCP MUST have a component
ID of 2. Other types of media streams which require multiple
components MUST develop specifications which define the mapping of
components to component IDs. See Section 14 for additional
discussion on extending ICE to new media streams.

<priority>: is a positive integer between 1 and (2**32 - 1).

<cand-type>: encodes the type of candidate. This specification
defines the values "host", "srflx", "prflx" and "relay" for host,
server reflexive, peer reflexive and relayed candidates,
respectively. The set of candidate types is extensible for the
future.

<rel-addr> and <rel-port>: convey transport addresses related to the
candidate, useful for diagnostics and other purposes. <rel-addr>
and <rel-port> MUST be present for server reflexive, peer
reflexive and relayed candidates. If a candidate is server or
peer reflexive, <rel-addr> and <rel-port> is equal to the base for
that server or peer reflexive candidate. If the candidate is
relayed, <rel-addr> and <rel-port> is equal to the mapped address
in the Allocate Response that provided the client with that
relayed candidate (see Appendix B.3 for a discussion of its
purpose). If the candidate is a host candidate <rel-addr> and
<rel-port> MUST be omitted.

The candidate attribute can itself be extended. The grammar allows
for new name/value pairs to be added at the end of the attribute. An
implementation MUST ignore any name/value pairs it doesn't
understand.

15.2. "remote-candidates" Attribute

The syntax of the "remote-candidates" attribute is defined using
Augmented BNF as defined in RFC 4234 [RFC4234]. The remote-
candidates attribute is a media level attribute only.


```
remote-candidate-att = "remote-candidates" ":" remote-candidate
                       0*(SP remote-candidate)
remote-candidate = component-ID SP connection-address SP port
```

The attribute contains a connection-address and port for each
component. The ordering of components is irrelevant. However, a
value MUST be present for each component of a media stream. This
attribute MUST be included in an offer by a controlling agent for a
media stream that is Completed, and MUST NOT be included in any other
case.

15.3. "ice-lite" and "ice-mismatch" Attributes

The syntax of the "ice-lite" and "ice-mismatch" attributes, both of
which are flags, is:


```
ice-lite               = "ice-lite"
ice-mismatch           = "ice-mismatch"
```

"ice-lite" is a session level attribute only, and indicates that an
agent is a lite implementation. "ice-mismatch" is a media level
attribute only, and when present in an answer, indicates that the
offer arrived with a default destination for a media component that
didn't have a corresponding candidate attribute.

15.4. "ice-ufrag" and "ice-pwd" Attributes

The "ice-ufrag" and "ice-pwd" attributes convey the username fragment
and password used by ICE for message integrity. Their syntax is:


```
ice-pwd-att            = "ice-pwd" ":" password
ice-ufrag-att          = "ice-ufrag" ":" ufrag
password               = 22*1024ice-char
ufrag                  = 4*1024ice-char
```

The "ice-pwd" and "ice-ufrag" attributes can appear at either the
session-level or media-level. When present in both, the value in the
media-level takes precedence. Thus, the value at the session level
is effectively a default that applies to all media streams, unless
overriden by a media-level value. Whether present at the session or
media level, there MUST be an ice-pwd and ice-ufrag attribute for
each media stream. If two media streams have identical ice-ufrag's,
they MUST have identical ice-pwd's.

The ice-ufrag and ice-pwd attributes MUST be chosen randomly at the
beginning of a session. The ice-ufrag attribute MUST contain at
least 24 bits of randomness, and the ice-pwd attribute MUST contain
at least 128 bits of randomness. This means that the ice-ufrag
attribute will be at least 4 characters long, and the ice-pwd at
least 22 characters long, since the grammar for these attributes
allows for 6 bits of randomness per character. The attributes MAY be
longer than 4 and 22 characters respectively, of course, up to 1024
characters. The upper limit allows for buffer sizing in
implementations. Its large upper limit allows for increased amounts
of randomness to be added over time.

15.5. "ice-options" Attribute

The "ice-options" attribute is a session level attribute. It
contains a series of tokens which identify the options supported by
the agent. Its grammar is:


```
ice-options            = "ice-options" ":" ice-option-tag
                            0*(SP ice-option-tag)
ice-option-tag         = 1*ice-char
```


16. Setting Ta and RTO

During the gathering phase of ICE (Section 4.1.1) and while ICE is
performing connectivity checks (Section 7), an agent sends STUN and
TURN transactions. These transcations are paced at a rate of one
every Ta milliseconds, and utilize a specific RTO. This section
describes how the value of Ta and RTO are computed. Their values
change during the lifetime of ICE processing. One set of values
applies during the gathering phase, and the other, for connectivity
checks.

The value of Ta SHOULD be configurable, and SHOULD have a default of:

For each media stream i:
 Ta_i = (stun_packet_size / rtp_packet_size) * rtp_ptime

```
                        1
   Ta = MAX (20ms, ------------------ )
                        k
                      ----
                      \        1
                       >    ------
                      /       Ta_i
                      ----
                       i=1
```

Where k is the number of media streams. During the gathering phase,
Ta is computed based on the number of media streams the agent has
indicated in its offer or answer, and the RTP packet size and RTP
ptime are those of the most preferred codec for each media stream.
Once an offer and answer have been exchanged, the agent recomputes Ta
to pace the connectivity checks. In that case, the value of Ta is
based on the number of media streams that will actually be used in
the session, and the RTP packet size and RTP ptime are those of the
most preferred codec that the agent will send with.

In addition, the retransmission timer for the STUN transactions, RTO,
defined in [I-D.ietf-behave-rfc3489bis], SHOULD be configurable and
during the gathering phase, SHOULD have a default of:

   RTO = MAX (100ms, Ta * (number of pairs))

Where the number of pairs refers to the number of pairs of candidates
with STUN or TURN servers.

For connectivity checks, RTO SHOULD be configurable and SHOULD have a
default of:

   RTO = MAX (100ms, Ta*N * (Num-Waiting))

Where Num-Waiting are the number of checks in the check list in the
Waiting state. Note that the RTO will be different for each
transaction as the number of checks in the Waiting state changes.

These formulas are aimed at causing STUN transactions to be paced at
the same rate as media. This ensures that ICE will work properly
under the same network conditions needed to support the media as
well. See Appendix B.1 for additional discussion and motivations.

Because of this pacing, it will take a certain amount of time to
obtain all of the server reflexive and relayed candidates.
Implementations should be aware of the time required to do this, and
if the application requires a time budget, limit the number of
candidates which are gathered.

17. **Example**

The example is based on the simplified topology of Figure 19.

```
                        +-----+
                        |     |
                        |STUN |
                        | Srvr|
                        +-----+
                           |
                +---------------------+
                |                     |
                |      Internet       |
                |                     |
                |                     |
                +---------------------+
                   |                 |
                   |                 |
             +---------+             |
             |   NAT   |             |
             +---------+             |
                 |                   |
                 |                   |
                 |                   |
               +-----+             +-----+
               |     |             |     |
               |  L  |             |  R  |
               |     |             |     |
               +-----+             +-----+
```
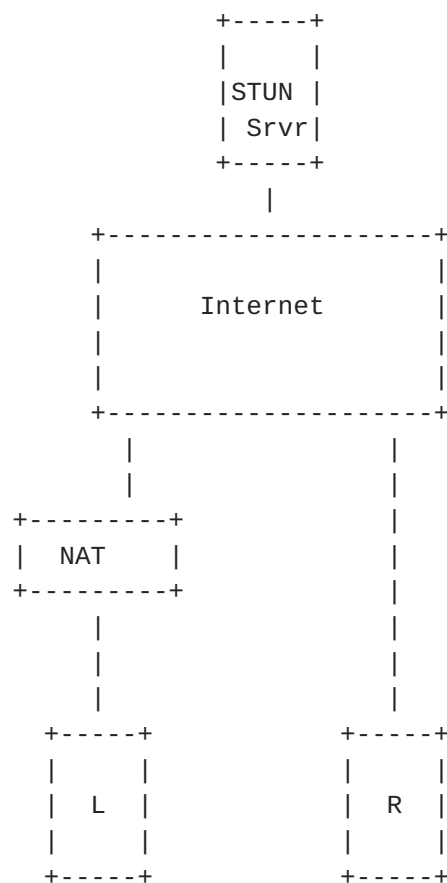
                   Figure 19: Example Topology

Two agents, L and R, are using ICE. Both are full-mode ICE
implementations. Both agents have a single IPv4 interface. For
agent L, it is 10.0.1.1 in private address space [RFC1918], and for
agent R, 192.0.2.1 on the public Internet. Both are configured with
the same STUN server (shown in this example for simplicity, although
in practice the agents do not need to use the same STUN server),
which is listening for STUN Binding Requests at an IP address of
192.0.2.2 and port 3478. TURN servers are not used in this example.

Agent L is behind a NAT, and agent R is on the public Internet. The
NAT has an endpoint independent mapping property and an address
dependent filtering property. The public side of the NAT has an IP
address of 192.0.2.3.

To facilitate understanding, transport addresses are listed using
variables that have mnemonic names. The format of the name is
entity-type-seqno, where entity refers to the entity whose interface
the transport address is on, and is one of "L", "R", "STUN", or
"NAT". The type is either "PUB" for transport addresses that are
public, and "PRIV" for transport addresses that are private.
Finally, seq-no is a sequence number that is different for each
transport address of the same type on a particular entity. Each
variable has an IP address and port, denoted by varname.IP and
varname.PORT, respectively, where varname is the name of the
variable.

The STUN server has advertised transport address STUN-PUB-1 (which is
192.0.2.2:3478).

In the call flow itself, STUN messages are annotated with several
attributes. The "S=" attribute indicates the source transport
address of the message. The "D=" attribute indicates the destination
transport address of the message. The "MA=" attribute is used in
STUN Binding Response messages and refers to the mapped address.
"USE-CAND" implies the presence of the USE-CANDIDATE attribute.

The call flow examples omit STUN authentication operations and RTCP,
and focus on RTP for a single media stream between two full
implementations.

```
            L               NAT             STUN               R
           |RTP STUN alloc.                 |                  |
           |(1) STUN Req   |                |                  |
           |S=$L-PRIV-1    |                |                  |
           |D=$STUN-PUB-1  |                |                  |
           |------------->|                 |                  |
           |               |(2) STUN Req   |                  |
           |               |S=$NAT-PUB-1   |                  |
           |               |D=$STUN-PUB-1  |                  |
           |               |------------->|                   |
           |               |(3) STUN Res   |                  |
           |               |S=$STUN-PUB-1  |                  |
           |               |D=$NAT-PUB-1   |                  |
           |               |MA=$NAT-PUB-1  |                  |
           |               |<-------------|                   |
           |(4) STUN Res   |                |                  |
```

```
          |S=$STUN-PUB-1 |                  |              |
          |D=$L-PRIV-1   |                  |              |
          |MA=$NAT-PUB-1 |                  |              |
          |<-------------|                  |              |
          |(5) Offer     |                  |              |
          |--------------------------------------------------->|
          |              |                  |              |RTP STUN alloc.
          |              |                  |(6) STUN Req  |
          |              |                  |S=$R-PUB-1    |
          |              |                  |D=$STUN-PUB-1 |
          |              |                  |<-------------|
          |              |                  |(7) STUN Res  |
          |              |                  |S=$STUN-PUB-1 |
          |              |                  |D=$R-PUB-1    |
          |              |                  |MA=$R-PUB-1   |
          |              |                  |------------->|
          |(8) answer    |                  |              |
          |<-------------------------------------------------|
          |              |(9) Bind Req  |                  |
          |              |S=$R-PUB-1    |                  |
          |              |D=L-PRIV-1    |                  |
          |              |<---------------------------------|
          |              |Dropped       |                  |
          |(10) Bind Req |              |                  |
          |S=$L-PRIV-1   |              |                  |
          |D=$R-PUB-1    |              |                  |
          |USE-CAND      |              |                  |
          |------------->|              |                  |
          |              |(11) Bind Req |                  |
          |              |S=$NAT-PUB-1  |                  |
          |              |D=$R-PUB-1    |                  |
          |              |USE-CAND      |                  |
          |              |--------------------------------->|
          |              |(12) Bind Res |                  |
          |              |S=$R-PUB-1    |                  |
          |              |D=$NAT-PUB-1  |                  |
          |              |MA=$NAT-PUB-1 |                  |
          |              |<---------------------------------|
          |(13) Bind Res |              |                  |
          |S=$R-PUB-1    |              |                  |
          |D=$L-PRIV-1   |              |                  |
          |MA=$NAT-PUB-1 |              |                  |
          |<-------------|              |                  |
          |RTP flows     |              |                  |
          |              |(14) Bind Req |                  |
          |              |S=$R-PUB-1    |                  |
          |              |D=$NAT-PUB-1  |                  |
          |              |<---------------------------------|
```

```
         |(15) Bind Req |                 |                |
         |S=$R-PUB-1    |                 |                |
         |D=$L-PRIV-1   |                 |                |
         |<-------------|                 |                |
         |(16) Bind Res |                 |                |
         |S=$L-PRIV-1   |                 |                |
         |D=$R-PUB-1    |                 |                |
         |MA=$R-PUB-1   |                 |                |
         |------------->|                 |                |
         |              |(17) Bind Res |                 |
         |              |S=$NAT-PUB-1  |                 |
         |              |D=$R-PUB-1    |                 |
         |              |MA=$R-PUB-1   |                 |
         |              |--------------------------->|
         |              |              |                 |RTP flows
```
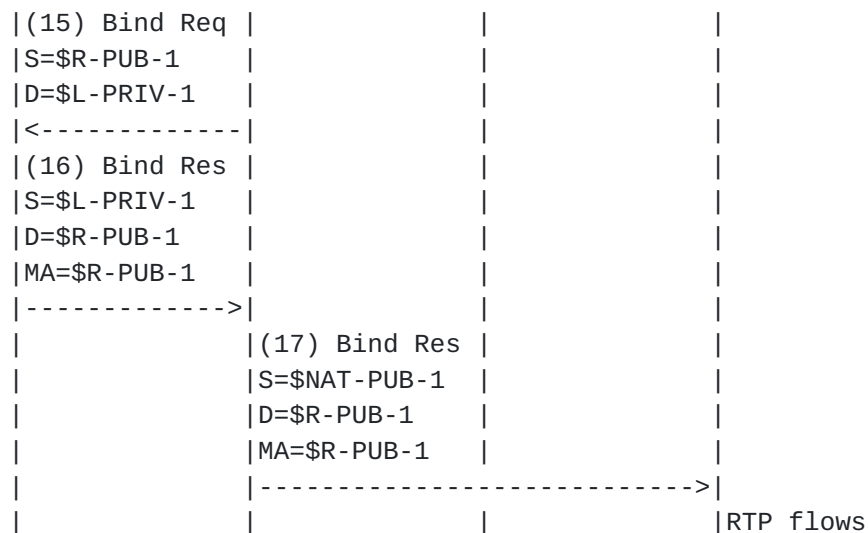
                        Figure 20: Example Flow

   First, agent L obtains a host candidate from its local interface (not
   shown), and from that, sends a STUN Binding Request to the STUN
   server to get a server reflexive candidate (messages 1-4). Recall
   that the NAT has the address and port independent mapping property.
   Here, it creates a binding of NAT-PUB-1 for this UDP request, and
   this becomes the server reflexive candidate for RTP.

   Agent L sets a type preference of 126 for the host candidate and 100
   for the server reflexive. The local preference is 65535. Based on
   this, the priority of the host candidate is 2130706431 and for the
   server reflexive candidate is 1694498815. The host candidate is
   assigned a foundation of 1, and the server reflexive, a foundation of
   2. It chooses its server reflexive candidate as the default
   candidate, and encodes it into the m and c lines. The resulting
   offer (message 5) looks like (lines folded for clarity):

     v=0
     o=jdoe 2890844526 2890842807 IN IP4 $L-PRIV-1.IP
     s=
     c=IN IP4 $NAT-PUB-1.IP
     t=0 0
     a=ice-pwd:asd88fgpdd777uzjYhagZg
     a=ice-ufrag:8hhY
     m=audio $NAT-PUB-1.PORT RTP/AVP 0
     b=RS:0
     b=RR:0
     a=rtpmap:0 PCMU/8000
     a=candidate:1 1 UDP 2130706431 $L-PRIV-1.IP $L-PRIV-1.PORT typ host
```

```
   a=candidate:2 1 UDP 1694498815 $NAT-PUB-1.IP $NAT-PUB-1.PORT typ
    srflx raddr $L-PRIV-1.IP rport $L-PRIV-1.PORT
```

The offer, with the variables replaced with their values, will look
like (lines folded for clarity):

```
   v=0
   o=jdoe 2890844526 2890842807 IN IP4 10.0.1.1
   s=
   c=IN IP4 192.0.2.3
   t=0 0
   a=ice-pwd:asd88fgpdd777uzjYhagZg
   a=ice-ufrag:8hhY
   m=audio 45664 RTP/AVP 0
   b=RS:0
   b=RR:0
   a=rtpmap:0 PCMU/8000
   a=candidate:1 1 UDP 2130706431 10.0.1.1 8998 typ host
   a=candidate:2 1 UDP 1694498815 192.0.2.3 45664 typ srflx raddr
10.0.1.1 rport 8998
```

This offer is received at agent R. Agent R will obtain a host
candidate, and from it, obtain a server reflexive candidate (messages
6-7). Since R is not behind a NAT, this candidate is identical to
its host candidate, and they share the same base. It therefore
discards this redundant candidate and ends up with a single host
candidate. With identical type and local preferences as L, the
priority for this candidate is 2130706431. It chooses a foundation
of 1 for its single candidate. Its resulting answer looks like:

```
   v=0
   o=bob 2808844564 2808844564 IN IP4 $R-PUB-1.IP
   s=
   c=IN IP4 $R-PUB-1.IP
   t=0 0
   a=ice-pwd:YH75Fviy6338Vbrhrlp8Yh
   a=ice-ufrag:9uB6
   m=audio $R-PUB-1.PORT RTP/AVP 0
   b=RS:0
   b=RR:0
   a=rtpmap:0 PCMU/8000
   a=candidate:1 1 UDP 2130706431 $R-PUB-1.IP $R-PUB-1.PORT typ host
```

With the variables filled in:

```
        v=0
        o=bob 2808844564 2808844564 IN IP4 192.0.2.1
        s=
        c=IN IP4 192.0.2.1
        t=0 0
        a=ice-pwd:YH75Fviy6338Vbrhrlp8Yh
        a=ice-ufrag:9uB6
        m=audio 3478 RTP/AVP 0
        b=RS:0
        b=RR:0
        a=rtpmap:0 PCMU/8000
        a=candidate:1 1 UDP 2130706431 192.0.2.1 3478 typ host
```

Since neither side indicated that they are lite, the agent which sent
the offer that began ICE processing (agent L) becomes the controlling
agent.

Agents L and R both pair up the candidates. They both initially have
two pairs. However, agent L will prune the pair containing its
server reflexive candidate, resulting in just one. At agent L, this
pair has a local candidate of $L_PRIV_1 and remote candidate of
$R_PUB_1, and has a candidate pair priority of 4.57566E+18 (note that
an implementation would represent this as a 64 bit integer so as not
to lose precision). At agent R, there are two pairs. The highest
priority has a local candidate of $R_PUB_1 and remote candidate of
$L_PRIV_1 and has a priority of 4.57566E+18, and the second has a
local candidate of $R_PUB_1 and remote candidate of $NAT_PUB_1 and
priority 3.63891E+18.

Agent R begins its connectivity check (message 9) for the first pair
(between the two host candidates). Since R is the controlled agent
for this session, the check omits the USE-CANDIDATE attribute. The
host candidate from agent L is private and behind a NAT, and thus
this check won't be successful, because the packet cannot be routed
from R to L.

When agent L gets the answer, it performs its one and only
connectivity check (messages 10-13). It implements the aggressive
nomination algorithm, and thus includes a USE-CANDIDATE attribute in
this check. Since the check succeeds, agent L creates a new pair,
whose local candidate is from the mapped address in the binding
response (NAT-PUB-1 from message 13) and whose remote candidate is
the destination of the request (R-PUB-1 from message 10). This is
added to the valid list. In addition, it is marked as selected since
the Binding Request contained the USE-CANDIDATE attribute. Since
there is a selected candidate in the Valid list for the one component
of this media stream, ICE processing for this stream moves into the
Completed state. Agent L can now send media if it so chooses.

Soon after receipt of the STUN Binding Request from agent L (message 11), agent R will generate its triggered check. This check happens to match the next one on its check list - from its host candidate to agent L's server reflexive candidate. This check (messages 14-17) will succeed. Consequently, agent R constructs a new candidate pair using the mapped address from the response as the local candidate (R-PUB-1) and the destination of the request (NAT-PUB-1) as the remote candidate. This pair is added to the Valid list for that media stream. Since the check was generated in the reverse direction of a check that contained the USE-CANDIDATE attribute, the candidate pair is marked as selected. Consequently, processing for this stream moves into the Completed state, and agent R can also send media.

18. Security Considerations

There are several types of attacks possible in an ICE system. This section considers these attacks and their countermeasures.

18.1. Attacks on Connectivity Checks

An attacker might attempt to disrupt the STUN connectivity checks.
Ultimately, all of these attacks fool an agent into thinking something incorrect about the results of the connectivity checks.
The possible false conclusions an attacker can try and cause are:

False Invalid: An attacker can fool a pair of agents into thinking a
   candidate pair is invalid, when it isn't. This can be used to
   cause an agent to prefer a different candidate (such as one
   injected by the attacker), or to disrupt a call by forcing all
   candidates to fail.

False Valid: An attacker can fool a pair of agents into thinking a
   candidate pair is valid, when it isn't. This can cause an agent
   to proceed with a session, but then not be able to receive any
   media.

False Peer-Reflexive Candidate: An attacker can cause an agent to
   discover a new peer reflexive candidate, when it shouldn't have.
   This can be used to redirect media streams to a DoS target or to
   the attacker, for eavesdropping or other purposes.

False Valid on False Candidate: An attacker has already convinced an
   agent that there is a candidate with an address that doesn't
   actually route to that agent (for example, by injecting a false
   peer reflexive candidate or false server reflexive candidate). It
   must then launch an attack that forces the agents to believe that
   this candidate is valid.

   If an attacker can cause a false per-reflexive candidate or false
   valid on a false candidate, it can launch any of the attacks
   described in draft-ietf-behave-rfc3489bis
   [I-D.ietf-behave-rfc3489bis].

To force the false invalid result, the attacker has to wait for the
connectivity check from one of the agents to be sent. When it is,
the attacker needs to inject a fake response with an unrecoverable
error response, such as a 400. However, since the candidate is, in
fact, valid, the original request may reach the peer agent, and
result in a success response. The attacker needs to force this
packet or its response to be dropped, through a DoS attack, layer 2
network disruption, or other technique. If it doesn't do this, the
success response will also reach the originator, alerting it to a
possible attack. Fortunately, this attack is mitigated completely
through the STUN short term credential mechanism. The attacker needs
to inject a fake response, and in order for this response to be
processed, the attacker needs the password. If the offer/answer
signaling is secured, the attacker will not have the password and its
response will be discarded.

Forcing the fake valid result works in a similar way. The agent
needs to wait for the Binding Request from each agent, and inject a
fake success response. The attacker won't need to worry about
disrupting the actual response since, if the candidate is not valid,
it presumably wouldn't be received anyway. However, like the fake
invalid attack, this attack is mitigated by the STUN short term
credential mechanism in conjunction with a secure offer/answer
exchange.

Forcing the false peer reflexive candidate result can be done either
with fake requests or responses, or with replays. We consider the
fake requests and responses case first. It requires the attacker to
send a Binding Request to one agent with a source IP address and port
for the false candidate. In addition, the attacker must wait for a
Binding Request from the other agent, and generate a fake response
with a XOR-MAPPED-ADDRESS attribute containing the false candidate.
Like the other attacks described here, this attack is mitigated by
the STUN message integrity mechanisms and secure offer/answer
exchanges.

Forcing the false peer reflexive candidate result with packet replays
is different. The attacker waits until one of the agents sends a
check. It intercepts this request, and replays it towards the other
agent with a faked source IP address. It must also prevent the
original request from reaching the remote agent, either by launching
a DoS attack to cause the packet to be dropped, or forcing it to be
dropped using layer 2 mechanisms. The replayed packet is received at

the other agent, and accepted, since the integrity check passes (the integrity check cannot and does not cover the source IP address and port). It is then responded to. This response will contain a XOR-MAPPED-ADDRESS with the false candidate, and will be sent to that false candidate. The attacker must then receive it and relay it towards the originator.

The other agent will then initiate a connectivity check towards that false candidate. This validation needs to succeed. This requires the attacker to force a false valid on a false candidate. Injecting of fake requests or responses to achieve this goal is prevented using the integrity mechanisms of STUN and the offer/answer exchange. Thus, this attack can only be launched through replays. To do that, the attacker must intercept the check towards this false candidate, and replay it towards the other agent. Then, it must intercept the response and replay that back as well.

This attack is very hard to launch unless the attacker is identified by the fake candidate. This is because it requires the attacker to intercept and replay packets sent by two different hosts. If both agents are on different networks (for example, across the public Internet), this attack can be hard to coordinate, since it needs to occur against two different endpoints on different parts of the network at the same time.

If the attacker themself is identified by the fake candidate the attack is easier to coordinate. However, if SRTP is used [RFC3711], the attacker will not be able to play the media packets, they will only be able to discard them, effectively disabling the media stream for the call. However, this attack requires the agent to disrupt packets in order to block the connectivity check from reaching the target. In that case, if the goal is to disrupt the media stream, its much easier to just disrupt it with the same mechanism, rather than attack ICE.

18.2. Attacks on Server Reflexive Address Gathering

ICE endpoints make use of STUN Binding requests for gathering server reflexive andidates from a STUN server. These requests are not authenticated in any way. As a consequence, there are numerous techinques an attacker can employ to provide the client with a false server reflexive candidate:

o An attacker can compromise the DNS, causing DNS queries to return a rogue STUN server address. That server can provide the client with fake server reflexive candidates. This attack is mitigated by DNS security, though DNS-SEC is not required to address it.

   o An attacker that can observe STUN messages (such as an attacker on
      a shared network segment, like WiFi), can inject a fake response
      that is valid and will be accepted by the client.

   o An attacker can compromise a STUN server by means of a virus, and
      cause it to send responses with incorrect mapped addresses.

   A false mapped address learned by these attacks will be used as a
   server reflexive candidate in the ICE exchange. For this candidate
   to actually be used for media, the attacker must also attack the
   connectivity checks, and in particular, force a false valid on a
   false candidate. This attack is very hard to launch if the false
   address identifies a fourth party (neither the offerer, answerer, or
   attacker), since it requires attacking the checks generated by each
   agent in the session, and is prevented by SRTP if it identifies the
   attacker themself.

   If the attacker elects not to attack the connectivity checks, the
   worst it can do is prevent the server reflexive candidate from being
   used. However, if the peer agent has at least one candidate that is
   reachable by the agent under attack, the STUN connectivity checks
   themselves will provide a peer reflexive candidate that can be used
   for the exchange of media. Peer reflexive candidates are generally
   preferred over server reflexive candidates. As such, an attack
   solely on the STUN address gathering will normally have no impact on
   a session at all.

18.3. Attacks on Relayed Candidate Gathering

   An attacker might attempt to disrupt the gathering of relayed
   candidates, forcing the client to believe it has a false relayed
   candidate. Exchanges with the TURN server are authenticated using a
   long term credential. Consequently, injection of fake responses or
   requests will not work. In addition, unlike Binding requests,
   Allocate requests are not susceptible to replay attacks with modified
   source IP addresses and ports, since the source IP address and port
   is not utilized to provide the client with its relayed candidate.

   However, TURN servers are susceptible to DNS attacks, or to viruses
   aimed at the TURN server, for purposes of turning it into a zombie or
   rogue server. These attacks can be mitigated by DNS-SEC and through
   good box and software security on TURN servers.

   Even if an attacker has caused the client to believe in a false
   relayed candidate, the connectivity checks cause such a candidate to
   be used only if they succeed. Thus, an attacker must launch a false
   valid on a false candidate, per above, which is a very difficult
   attack to coordinate.

18.4. Attacks on the Offer/Answer Exchanges

   An attacker that can modify or disrupt the offer/answer exchanges
   themselves can readily launch a variety of attacks with ICE. They
   could direct media to a target of a DoS attack, they could insert
   themselves into the media stream, and so on. These are similar to
   the general security considerations for offer/answer exchanges, and
   the security considerations in RFC 3264 [RFC3264] apply. These
   require techniques for message integrity and encryption for offers
   and answers, which are satisfied by the SIPS mechanism [RFC3261] when
   SIP is used. As such, the usage of SIPS with ICE is RECOMMENDED.

18.5. Insider Attacks

   In addition to attacks where the attacker is a third party trying to
   insert fake offers, answers or stun messages, there are several
   attacks possible with ICE when the attacker is an authenticated and
   valid participant in the ICE exchange.

18.5.1. The Voice Hammer Attack

   The voice hammer attack is an amplification attack. In this attack,
   the attacker initiates sessions to other agents, and maliciously
   includes the IP address and port of a DoS target as the destination
   for media traffic signaled in the SDP. This causes substantial
   amplification; a single offer/answer exchange can create a continuing
   flood of media packets, possibly at high rates (consider video
   sources). This attack is not specific to ICE, but ICE can help
   provide remediation.

   Specifically, if ICE is used, the agent receiving the malicious SDP
   will first perform connectivity checks to the target of media before
   sending media there. If this target is a third party host, the
   checks will not succeed, and media is never sent.

   Unfortunately, ICE doesn't help if its not used, in which case an
   attacker could simply send the offer without the ICE parameters.
   However, in environments where the set of clients are known, and
   limited to ones that support ICE, the server can reject any offers or
   answers that don't indicate ICE support.

18.5.2. STUN Amplification Attack

   The STUN amplification attack is similar to the voice hammer.
   However, instead of voice packets being directed to the target, STUN
   connectivity checks are directed to the target. The attacker sends
   an offer with a large number of candidates, say 50. The answerer
   receives the offer, and starts its checks, which are directed at the

target, and consequently, never generate a response. The answerer
will start a new connectivity check every Ta ms (say Ta=20ms).
However, the retransmission timers are set to a large number due to
the large number of candidates. As a consequence, packets will be
sent at an interval of one every Ta milliseconds, and then with
increasing intervals after that. Thus, STUN will not send packets at
a rate faster than media would be sent, and the STUN packets persist
only briefly, until ICE fails for the session. Nonetheless, this is
an amplification mechanism.

It is impossible to eliminate the amplification, but the volume can
be reduced through a variety of heuristics. Agents SHOULD limit the
total number of connectivity checks they perform to 100.
Additionally, agents MAY limit the number of candidates they'll
accept in an offer or answer.

18.6. Interactions with Application Layer Gateways and SIP

Application Layer Gateways (ALGs) are functions present in a NAT
device which inspect the contents of packets and modify them, in
order to facilitate NAT traversal for application protocols. Session
Border Controllers (SBC) are close cousins of ALGs, but are less
transparent since they actually exist as application layer SIP
intermediaries. ICE has interactions with SBCs and ALGs.

If an ALG is SIP aware but not ICE aware, ICE will work through it as
long as the ALG correctly modifies the SDP. A correct ALG
implementation behave as follows:

o The ALG does not modify the m and c lines or the rtcp attribute if
   they contain external addresses.

o If the m and c lines contain internal addresses, the modification
   depends on the state of the ALG:

      If the ALG already has a binding established that maps an
      external port to an internal IP address and port matching the
      values in the m and c lines or rtcp attribute , the ALG uses
      that binding instead of creating a new one.

      If the ALG does not already have a binding, it creates a new
      one and modifies the SDP, rewriting the m and c lines and rtcp
      attribute.

Unfortunately, many ALG are known to work poorly in these corner
cases. ICE does not try to work around broken ALGs, as this is
outside the scope of its functionality. ICE can help diagnose these
conditions, which often show up as a mismatch between the set of

candidates and the m and c lines and rtcp attributes. The ice-
mismatch attribute is used for this purpose.

ICE works best through ALGs when the signaling is run over TLS. This
prevents the ALG from manipulating the SDP messages and interfering
with ICE operation. Implementations which are expected to be
deployed behind ALGs SHOULD provide for TLS transport of the SDP.

If an SBC is SIP aware but not ICE aware, the result depends on the
behavior of the SBC. If it is acting as a proper Back-to-Back User
Agent (B2BUA), the SBC will remove any SDP attributes it doesn't
understand, including the ICE attributes. Consequently, the call
will appear to both endpoints as if the other side doesn't support
ICE. This will result in ICE being disabled, and media flowing
through the SBC, if the SBC has requested it. If, however, the SBC
passes the ICE attributes without modification, yet modifies the
default destination for media (contained in the m and c lines and
rtcp attribute), this will be detected as an ICE mismatch, and ICE
processing is aborted for the call. It is outside of the scope of
ICE for it to act as a tool for "working around" SBCs. If one is
present, ICE will not be used and the SBC techniques take precedence.

19. STUN Extensions

19.1. New Attributes

This specification defines four new attributes, PRIORITY, USE-
CANDIDATE, ICE-CONTROLLED and ICE-CONTROLLING.

The PRIORITY attribute indicates the priority that is to be
associated with a peer reflexive candidate, should one be discovered
by this check. It is a 32 bit unsigned integer, and has an attribute
value of 0x0024.

The USE-CANDIDATE attribute indicates that the candidate pair
resulting from this check should be used for transmission of media.
The attribute has no content (the Length field of the attribute is
zero); it serves as a flag. It has an attribute value of 0x0025.

The ICE-CONTROLLED attribute is present in a Binding Request, and
indicates that the client believes it is currently in the controlled
role. The content of the attribute is a 64 bit unsigned integer in
network byte ordering, which contains a random number used for tie-
breaking of role conflicts.

The ICE-CONTROLLING attribute is present in a Binding Request, and
indicates that the client believes it is currently in the controlling

role. The content of the attribute is a 64 bit unsigned integer in
network byte ordering, which contains a random number used for tie-
breaking of role conflicts.

19.2. New Error Response Codes

This specification defines a single error response code:

487 (Role Conflict): The Binding Request contained either the ICE-
   CONTROLLING or ICE-CONTROLLED attribute, indicating a role that
   conflicted with the server. The server ran a tie-breaker based on
   the tie-breaker value in the request, and determined that the
   client needs to switch roles.

20. IANA Considerations

This specification registers new SDP attributes, four new STUN
attributes and one new STUN error response.

20.1. SDP Attributes

This specification defines seven new SDP attributes per the
procedures of Section 8.2.4 of [RFC4566]. The required information
for the registrations are included here.

20.1.1. candidate Attribute

Contact Name: Jonathan Rosenberg, jdrosen@jdrosen.net.

Attribute Name: candidate

Long Form: candidate

Type of Attribute: media level

Charset Considerations: The attribute is not subject to the charset
   attribute.

Purpose: This attribute is used with Interactive Connectivity
   Establishment (ICE), and provides one of many possible candidate
   addresses for communication. These addresses are validated with
   an end-to-end connectivity check using Simple Traversal Underneath
   NAT (STUN).

Appropriate Values: See Section 15 of RFC XXXX [Note to RFC-ed:
   please replace XXXX with the RFC number of this specification].

20.1.2. remote-candidates Attribute

Contact Name: Jonathan Rosenberg, jdrosen@jdrosen.net.

Attribute Name: remote-candidates

Long Form: remote-candidates

Type of Attribute: media level

Charset Considerations: The attribute is not subject to the charset
   attribute.

Purpose: This attribute is used with Interactive Connectivity
   Establishment (ICE), and provides the identity of the remote
   candidates that the offerer wishes the answerer to use in its
   answer.

Appropriate Values: See Section 15 of RFC XXXX [Note to RFC-ed:
   please replace XXXX with the RFC number of this specification].

20.1.3. ice-lite Attribute

Contact Name: Jonathan Rosenberg, jdrosen@jdrosen.net.

Attribute Name: ice-lite

Long Form: ice-lite

Type of Attribute: session level

Charset Considerations: The attribute is not subject to the charset
   attribute.

Purpose: This attribute is used with Interactive Connectivity
   Establishment (ICE), and indicates that an agent has the minimum
   functionality required to support ICE inter-operation with a peer
   that has a full implementation.

Appropriate Values: See Section 15 of RFC XXXX [Note to RFC-ed:
   please replace XXXX with the RFC number of this specification].

20.1.4. ice-mismatch Attribute

   Contact Name: Jonathan Rosenberg, jdrosen@jdrosen.net.

   Attribute Name: ice-mismatch

   Long Form: ice-mismatch

   Type of Attribute: session level

   Charset Considerations: The attribute is not subject to the charset
      attribute.

   Purpose: This attribute is used with Interactive Connectivity
      Establishment (ICE), and indicates that an agent is ICE capable,
      but did not proceed with ICE due to a mismatch of candidates with
      the default destination for media signaled in the SDP.

   Appropriate Values: See Section 15 of RFC XXXX [Note to RFC-ed:
      please replace XXXX with the RFC number of this specification].

20.1.5. ice-pwd Attribute

   Contact Name: Jonathan Rosenberg, jdrosen@jdrosen.net.

   Attribute Name: ice-pwd

   Long Form: ice-pwd

   Type of Attribute: session or media level

   Charset Considerations: The attribute is not subject to the charset
      attribute.

   Purpose: This attribute is used with Interactive Connectivity
      Establishment (ICE), and provides the password used to protect
      STUN connectivity checks.

   Appropriate Values: See Section 15 of RFC XXXX [Note to RFC-ed:
      please replace XXXX with the RFC number of this specification].

20.1.6. ice-ufrag Attribute

   Contact Name: Jonathan Rosenberg, jdrosen@jdrosen.net.

Attribute Name: ice-ufrag

Long Form: ice-ufrag

Type of Attribute: session or media level

Charset Considerations: The attribute is not subject to the charset
   attribute.

Purpose: This attribute is used with Interactive Connectivity
   Establishment (ICE), and provides the fragments used to construct
   the username in STUN connectivity checks.

Appropriate Values: See Section 15 of RFC XXXX [Note to RFC-ed:
   please replace XXXX with the RFC number of this specification].

20.1.7. ice-options Attribute

Contact Name: Jonathan Rosenberg, jdrosen@jdrosen.net.

Attribute Name: ice-options

Long Form: ice-options

Type of Attribute: session level

Charset Considerations: The attribute is not subject to the charset
   attribute.

Purpose: This attribute is used with Interactive Connectivity
   Establishment (ICE), and indicates the ICE options or extensions
   used by the agent.

Appropriate Values: See Section 15 of RFC XXXX [Note to RFC-ed:
   please replace XXXX with the RFC number of this specification].

20.2. STUN Attributes

This section registers four new STUN attributes per the procedures in
[I-D.ietf-behave-rfc3489bis].

    0x0024 PRIORITY
    0x0025 USE-CANDIDATE
    0x8029 ICE-CONTROLLED
    0x802a ICE-CONTROLLING

20.3. STUN Error Responses

This section registers one new STUN error response code per the procedures in [I-D.ietf-behave-rfc3489bis].

    487   Role Conflict: The client asserted an ICE role (controlling or
           controlled) that is in conflict with the role of the server.

21. IAB Considerations

The IAB has studied the problem of "Unilateral Self Address Fixing", which is the general process by which a agent attempts to determine its address in another realm on the other side of a NAT through a collaborative protocol reflection mechanism [RFC3424]. ICE is an example of a protocol that performs this type of function. Interestingly, the process for ICE is not unilateral, but bilateral, and the difference has a signficant impact on the issues raised by IAB. Indeed, ICE can be considered a B-SAF (Bilateral Self-Address Fixing) protocol, rather than an UNSAF protocol. Regardless, the IAB has mandated that any protocols developed for this purpose document a specific set of considerations. This section meets those requirements.

21.1. Problem Definition

From RFC 3424 any UNSAF proposal must provide:

   Precise definition of a specific, limited-scope problem that is to
   be solved with the UNSAF proposal. A short term fix should not be
   generalized to solve other problems; this is why "short term fixes
   usually aren't".

The specific problems being solved by ICE are:

   Provide a means for two peers to determine the set of transport
   addresses which can be used for communication.

   Provide a means for a agent to determine an address that is
   reachable by another peer with which it wishes to communicate.

21.2. Exit Strategy

From RFC 3424, any UNSAF proposal must provide:

Description of an exit strategy/transition plan. The better short
term fixes are the ones that will naturally see less and less use
as the appropriate technology is deployed.

ICE itself doesn't easily get phased out. However, it is useful even
in a globally connected Internet, to serve as a means for detecting
whether a router failure has temporarily disrupted connectivity, for
example. ICE also helps prevent certain security attacks which have
nothing to do with NAT. However, what ICE does is help phase out
other UNSAF mechanisms. ICE effectively selects amongst those
mechanisms, prioritizing ones that are better, and deprioritizing
ones that are worse. Local IPv6 addresses can be preferred. As NATs
begin to dissipate as IPv6 is introduced, server reflexive and
relayed candidates (both forms of UNSAF addresses) simply never get
used, because higher priority connectivity exists to the native host
candidates. Therefore, the servers get used less and less, and can
eventually be remove when their usage goes to zero.

Indeed, ICE can assist in the transition from IPv4 to IPv6. It can
be used to determine whether to use IPv6 or IPv4 when two dual-stack
hosts communicate with SIP (IPv6 gets used). It can also allow a
network with both 6to4 and native v6 connectivity to determine which
address to use when communicating with a peer.

21.3. Brittleness Introduced by ICE

From RFC3424, any UNSAF proposal must provide:

Discussion of specific issues that may render systems more
"brittle". For example, approaches that involve using data at
multiple network layers create more dependencies, increase
debugging challenges, and make it harder to transition.

ICE actually removes brittleness from existing UNSAF mechanisms. In
particular, classic STUN (as described in RFC 3489 [RFC3489]) has
several points of brittleness. One of them is the discovery process
which requires a agent to try and classify the type of NAT it is
behind. This process is error-prone. With ICE, that discovery
process is simply not used. Rather than unilaterally assessing the
validity of the address, its validity is dynamically determined by
measuring connectivity to a peer. The process of determining
connectivity is very robust.

Another point of brittleness in classic STUN and any other unilateral
mechanism is its absolute reliance on an additional server. ICE
makes use of a server for allocating unilateral addresses, but allows
agents to directly connect if possible. Therefore, in some cases,
the failure of a STUN server would still allow for a call to progress

when ICE is used.

Another point of brittleness in classic STUN is that it assumes that the STUN server is on the public Internet. Interestingly, with ICE, that is not necessary. There can be a multitude of STUN servers in a variety of address realms. ICE will discover the one that has provided a usable address.

The most troubling point of brittleness in classic STUN is that it doesn't work in all network topologies. In cases where there is a shared NAT between each agent and the STUN server, traditional STUN may not work. With ICE, that restriction is removed.

Classic STUN also introduces some security considerations. Fortunately, those security considerations are also mitigated by ICE.

Consequently, ICE serves to repair the brittleness introduced in classic STUN, and does not introduce any additional brittleness into the system.

The penalty of these improvements is that ICE increases session establishment times.

21.4. Requirements for a Long Term Solution

From RFC 3424, any UNSAF proposal must provide:

   Identify requirements for longer term, sound technical solutions -- contribute to the process of finding the right longer term solution.

Our conclusions from RFC 3489 remain unchanged. However, we feel ICE actually helps because we believe it can be part of the long term solution.

21.5. Issues with Existing NAPT Boxes

From RFC 3424, any UNSAF proposal must provide:

   Discussion of the impact of the noted practical issues with existing, deployed NA[P]Ts and experience reports.

A number of NAT boxes are now being deployed into the market which try and provide "generic" ALG functionality. These generic ALGs hunt for IP addresses, either in text or binary form within a packet, and rewrite them if they match a binding. This interferes with classic STUN. However, the update to STUN [I-D.ietf-behave-rfc3489bis] uses an encoding which hides these binary addresses from generic ALGs.

Existing NAPT boxes have non-deterministic and typically short
expiration times for UDP-based bindings. This requires
implementations to send periodic keepalives to maintain those
bindings. ICE uses a default of 15s, which is a very conservative
estimate. Eventually, over time, as NAT boxes become compliant to
behave [RFC4787], this minimum keepalive will become deterministic
and well-known, and the ICE timers can be adjusted. Having a way to
discover and control the minimum keepalive interval would be far
better still.

22. Acknowledgements

The authors would like to thank Dan Wing, Eric Rescorla, Flemming
Andreasen, Rohan Mahy, Dean Willis, Eric Cooper, Jason Fischl,
Douglas Otis, Tim Moore, Jean-Francois Mule, Kevin Johns, Jonathan
Lennox and Francois Audet for their comments and input. A special
thanks goes to Bill May, who suggested several of the concepts in
this specification, Philip Matthews, who suggested many of the key
performance optimizations in this specification, Eric Rescorla, who
drafted the text in the introduction, and Magnus Westerlund, for
doing several detailed reviews on the various revisions of this
specification.

23. References

23.1. Normative References

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

[RFC3605]   Huitema, C., "Real Time Control Protocol (RTCP) attribute
            in Session Description Protocol (SDP)", RFC 3605,
            October 2003.

[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.

[RFC3556]   Casner, S., "Session Description Protocol (SDP) Bandwidth
            Modifiers for RTP Control Protocol (RTCP) Bandwidth",
            RFC 3556, July 2003.

   [RFC3312] Camarillo, G., Marshall, W., and J. Rosenberg,
              "Integration of Resource Management and Session Initiation
              Protocol (SIP)", RFC 3312, October 2002.

   [RFC4032] Camarillo, G. and P. Kyzivat, "Update to the Session
              Initiation Protocol (SIP) Preconditions Framework",
              RFC 4032, March 2005.

   [RFC4234] Crocker, D. and P. Overell, "Augmented BNF for Syntax
              Specifications: ABNF", RFC 4234, October 2005.

   [RFC3262] Rosenberg, J. and H. Schulzrinne, "Reliability of
              Provisional Responses in Session Initiation Protocol
              (SIP)", RFC 3262, June 2002.

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

   [RFC4091] Camarillo, G. and J. Rosenberg, "The Alternative Network
              Address Types (ANAT) Semantics for the Session Description
              Protocol (SDP) Grouping Framework", RFC 4091, June 2005.

   [RFC3484] Draves, R., "Default Address Selection for Internet
              Protocol version 6 (IPv6)", RFC 3484, February 2003.

   [I-D.ietf-behave-rfc3489bis]
              Rosenberg, J., "Session Traversal Utilities for (NAT)
              (STUN)", draft-ietf-behave-rfc3489bis-06 (work in
              progress), March 2007.

   [I-D.ietf-behave-turn]
              Rosenberg, J., "Obtaining Relay Addresses from Simple
              Traversal Underneath NAT (STUN)",
              draft-ietf-behave-turn-03 (work in progress), March 2007.

   [I-D.ietf-sip-ice-option-tag]
              Rosenberg, J., "Indicating Support for Interactive
              Connectivity Establishment (ICE) in the Session
              Initiation Protocol (SIP)",
              draft-ietf-sip-ice-option-tag-02 (work in progress),
              June 2007.

23.2. Informative References

   [RFC3489] Rosenberg, J., Weinberger, J., Huitema, C., and R. Mahy,
              "STUN - Simple Traversal of User Datagram Protocol (UDP)
              Through Network Address Translators (NATs)", RFC 3489,
              March 2003.

   [RFC3235] Senie, D., "Network Address Translator (NAT)-Friendly
              Application Design Guidelines", RFC 3235, January 2002.

   [RFC3303] Srisuresh, P., Kuthan, J., Rosenberg, J., Molitor, A., and
              A. Rayhan, "Middlebox communication architecture and
              framework", RFC 3303, August 2002.

   [RFC3725] Rosenberg, J., Peterson, J., Schulzrinne, H., and G.
              Camarillo, "Best Current Practices for Third Party Call
              Control (3pcc) in the Session Initiation Protocol (SIP)",
              BCP 85, RFC 3725, April 2004.

   [RFC3102] Borella, M., Lo, J., Grabelsky, D., and G. Montenegro,
              "Realm Specific IP: Framework", RFC 3102, October 2001.

   [RFC3103] Borella, M., Grabelsky, D., Lo, J., and K. Taniguchi,
              "Realm Specific IP: Protocol Specification", RFC 3103,
              October 2001.

   [RFC3424] Daigle, L. and IAB, "IAB Considerations for UNilateral
              Self-Address Fixing (UNSAF) Across Network Address
              Translation", RFC 3424, November 2002.

   [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V.
              Jacobson, "RTP: A Transport Protocol for Real-Time
              Applications", RFC 3550, July 2003.

   [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K.
              Norrman, "The Secure Real-time Transport Protocol (SRTP)",
              RFC 3711, March 2004.

   [RFC3056] Carpenter, B. and K. Moore, "Connection of IPv6 Domains
              via IPv4 Clouds", RFC 3056, February 2001.

   [RFC3389] Zopf, R., "Real-time Transport Protocol (RTP) Payload for
              Comfort Noise (CN)", RFC 3389, September 2002.

   [RFC3960] Camarillo, G. and H. Schulzrinne, "Early Media and Ringing
              Tone Generation in the Session Initiation Protocol (SIP)",
              RFC 3960, December 2004.

   [RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z.,
              and W. Weiss, "An Architecture for Differentiated
              Services", RFC 2475, December 1998.

   [RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and
              E. Lear, "Address Allocation for Private Internets",
              BCP 5, RFC 1918, February 1996.

   [RFC4787]   Audet, F. and C. Jennings, "Network Address Translation
               (NAT) Behavioral Requirements for Unicast UDP", BCP 127,
               RFC 4787, January 2007.

   [I-D.ietf-mmusic-connectivity-precon]
               Andreasen, F., "Connectivity Preconditions for Session
               Description Protocol Media Streams",
               draft-ietf-mmusic-connectivity-precon-02 (work in
               progress), June 2006.

   [I-D.ietf-avt-rtp-no-op]
               Andreasen, F., "A No-Op Payload Format for RTP",
               draft-ietf-avt-rtp-no-op-04 (work in progress), May 2007.

   [I-D.ietf-avt-rtp-and-rtcp-mux]
               Perkins, C. and M. Westerlund, "Multiplexing RTP Data and
               Control Packets on a Single Port",
               draft-ietf-avt-rtp-and-rtcp-mux-05 (work in progress),
               May 2007.

   [RFC4340]   Kohler, E., Handley, M., and S. Floyd, "Datagram
               Congestion Control Protocol (DCCP)", RFC 4340, March 2006.

   [RFC4103]   Hellstrom, G. and P. Jones, "RTP Payload for Text
               Conversation", RFC 4103, June 2005.

   [I-D.ietf-sip-outbound]
               Jennings, C. and R. Mahy, "Managing Client Initiated
               Connections in the Session Initiation Protocol (SIP)",
               draft-ietf-sip-outbound-09 (work in progress), June 2007.

Appendix A. Lite and Full Implementations

   ICE allows for two types of implementations. A full implementation
   supports the controlling and controlled roles in a session, and can
   also perform address gathering. In contrast, a lite implementation
   is a minimalist implementation that does little but respond to STUN
   checks.

   Because ICE requires both endpoints to support it in order to bring
   benefits to either endpoint, incremental deployment of ICE in a
   network is more complicated. Many sessions involve an endpoint which
   is, by itself, not behind a NAT and not one that would worry about
   NAT traversal. A very common case is to have one endpoint that
   requires NAT traversal (such as a VoIP hard phone or soft phone) make
   a call to one of these devices. Even if the phone supports a full
   ICE implementation, ICE won't be used at all if the other device

doesn't support it. The lite implementation allows for a low-cost
entry point for these devices. Once they support the lite
implementation, full implementations can connect to them and get the
full benefits of ICE.

Consequently, a lite implementation is only appropriate for devices
that will *always* be connected to the public Internet and have a
public IP address at which it can receive packets from any
correspondent. ICE will not function when a lite implementation is
placed behind a NAT.

ICE allows a lite implementation to have a single IPv4 host candidate
and several IPv6 addresses. In that case, candidate pairs are
selected by the controlling agent using a static algorithm, such as
the one in RFC 3484, which is recommended by this specification.
However, static mechanisms for address selection are always prone to
error, since they cannot ever reflect the actual topology and can
never provide actual guarantees on connectivity. They are always
heuristics. Consequently, if an agent is implementing ICE just to
select between its IPv4 and IPv6 addresses, and it is none of its
interfaces are behind NAT, usage of full ICE is still RECOMMENDED in
order to provide the most robust form of address selection possible.

It is important to note that the lite implementation was added to
this specification to provide a stepping stone to full
implementation. Even for devices that are always connected to the
public Internet with just a single IPv4 address, a full
implementation is preferable if achievable. A full implementation
will reduce call setup times, since ICE's aggressive mode can be
used. Full implementations also obtain the security benefits of ICE
unrelated to NAT traversal; in particular, the voice hammer attack
described in Section 18 is prevented only for full implementations,
not lite. Finally, it is often the case that a device which finds
itself with a public address today will be placed in a network
tomorrow where it will be behind a NAT. It is difficult to
definitively know, over the lifetime of a device or product, that it
will always be used on the public Internet. Full implementation
provides assurance that communications will always work.

Appendix B. Design Motivations

ICE contains a number of normative behaviors which may themselves be
simple, but derive from complicated or non-obvious thinking or use
cases which merit further discussion. Since these design motivations
are not neccesary to understand for purposes of implementation, they
are discussed here in an appendix to the specification. This section
is non-normative.

B.1. Pacing of STUN Transactions

   STUN transactions used to gather candidates and to verify
   connectivity are paced out at an approximate rate of one new
   transaction every Ta milliseconds. Each transaction, in turn, has a
   retransmission timer RTO that is a function of Ta as well. Why are
   these transactions paced, and why are these formulas used?

   Sending of these STUN requests will often have the effect of creating
   bindings on NAT devices between the client and the STUN servers.
   Experience has shown that many NAT devices have upper limits on the
   rate at which they will create new bindings. Experiments have shown
   that once every 20ms is well supported, but not much lower than that.
   This is why Ta has a lower bound of 20ms. Furthermore, transmission
   of these packets on the network makes use of bandwidth and needs to
   be rate limited by the agent. As a consequence, the pacing ensures
   that the NAT devices does not get overloaded and that traffic is kept
   at a reasonable rate.

   The definition of a "reasonable" rate is that STUN should not use
   more bandwidth than the RTP itself will use, once media starts
   flowing. The formula for Ta is designed so that, if a STUN packet
   were sent every Ta seconds, it would consume the same amount of
   bandwidth as RTP packets, summed across all media streams. Of
   course, STUN has retransmits, and the desire is to pace those as
   well. For this reason, RTO is set such that the first retransmit on
   the first transaction happens just as the first STUN request on the
   last transaction occurs. Pictorially:

              First Packets   Retransmits

                    | |
                    | |
              -------+------   -------+------
              /   \   /   \
             /   \   /   \

             +--+   +--+   +--+   +--+   +--+   +--+
             |A1| |B1| |C1| |A2| |B2| |C2|
             +--+   +--+   +--+   +--+   +--+   +--+

           ---+-------+-------+-------+-------+-------+------------ Time
              0   Ta   2Ta   3Ta   4Ta   5Ta

In this picture, there are three transactions that will be sent (for
example, in the case of candidate gathering, there are three host
candidate/STUN server pairs). These are transactions A, B and C. The
retransmit timer is set so that the first retransmission on the first
transaction (packet A2) is sent at time 3Ta.

Subsequent retransmits after the first will occur even less
frequently than Ta milliseconds apart, since STUN uses an exponential
back-off on its retransmissions.

B.2. **Candidates with Multiple Bases**

Section 4.1.1.3 talks about eliminating candidates that have the same
transport address and base. However, candidates with the same
transport addresses but different bases are not redundant . When can
an agent have two candidates that have the same IP address and port,
but different bases? Consider the topology of Figure 28:

```
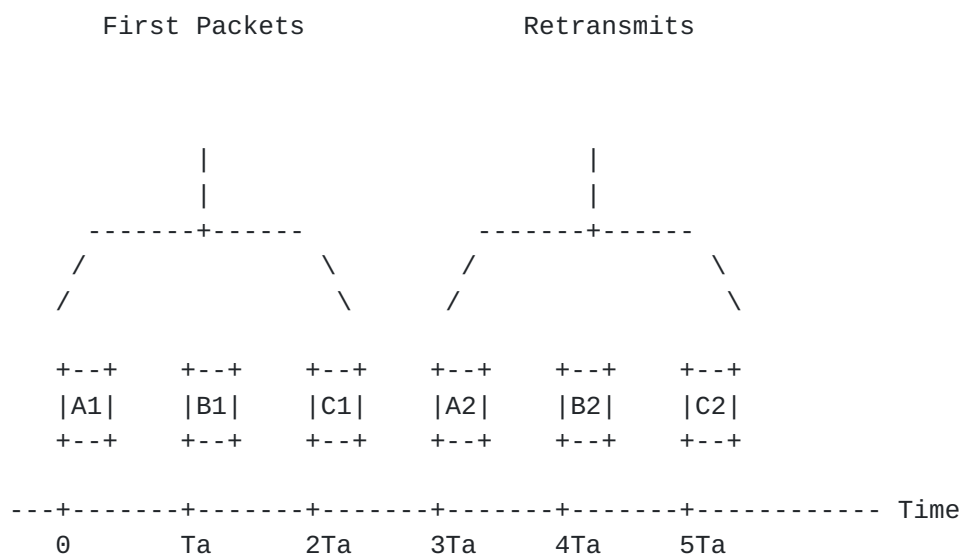                 +----------+
                 | STUN Srvr|
                 +----------+
                      |
                      |
                    -----
                  //     \\
                  |       |
                  |  B:net10  |
                  |       |
                   \\     //
                    -----
                      |
                      |
                 +----------+
                 |   NAT    |
                 +----------+
                      |
                      |
                    -----
                  //     \\
                  |   A   |
                  |192.168/16 |
                  |       |
                   \\     //
                    -----
                      |
                      |
                  |192.168.1.100       -----
        +----------+            //     \\            +----------+
        |          |            |       |            |          |
        | Offerer  |---------|  C:net10  |-----------| Answerer |
        |          |10.0.1.100|         | 10.0.1.101 |          |
        +----------+            \\     //            +----------+
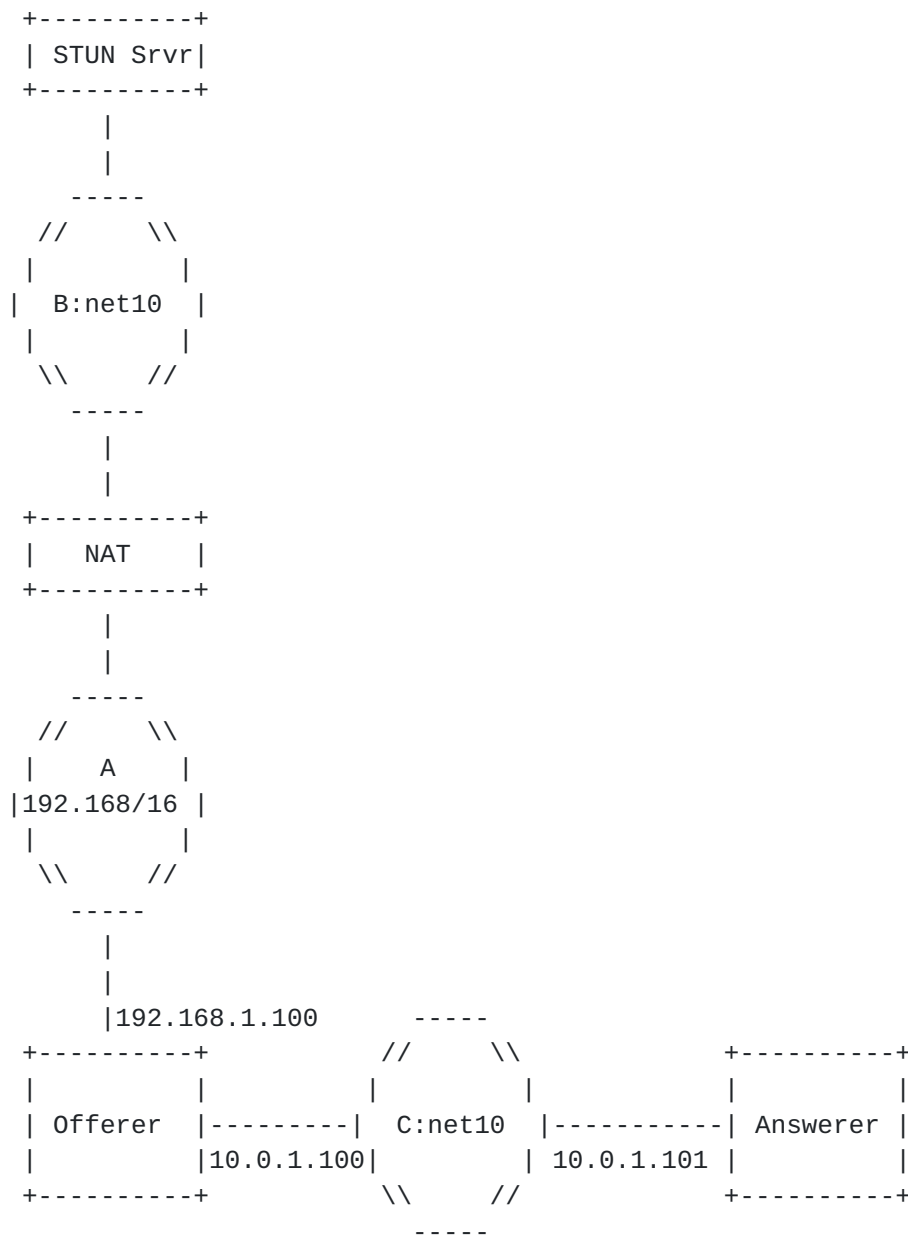                                 -----
```

Figure 28: Identical Candidates with Different Bases

   In this case, the offerer is multi-homed. It has one interface,
   10.0.1.100, on network C, which is a net 10 private network. The
   Answerer is on this same network. The offerer is also connected to
   network A, which is 192.168/16. The offerer has an interface of
   192.168.1.100 on this network. There is a NAT on this network,
   natting into network B, which is another net 10 private network, but
   not connected to network C. There is a STUN server on network B.

The offerer obtains a host candidate on its interface on network C
(10.0.1.100:2498) and a host candidate on its interface on network A
(192.168.1.100:3344). It performs a STUN query to its configured
STUN server from 192.168.1.100:3344. This query passes through the
NAT, which happens to assign the binding 10.0.1.100:2498. The STUN
server reflects this in the STUN Binding Response. Now, the offerer
has obtained a server reflexive candidate with a transport address
that is identical to a host candidate (10.0.1.100:2498). However,
the server reflexive candidate has a base of 192.168.1.100:3344, and
the host candidate has a base of 10.0.1.100:2498.

B.3. Purpose of the <rel-addr> and <rel-port> Attributes

The candidate attribute contains two values that are not used at all
by ICE itself - <rel-addr> and <rel-port>. Why is it present?

There are two motivations for its inclusion. The first is
diagnostic. It is very useful to know the relationship between the
different types of candidates. By including it, an agent can know
which relayed candidate is associated with which reflexive candidate,
which in turn is associated with a specific host candidate. When
checks for one candidate succeed and not the others, this provides
useful diagnostics on what is going on in the network.

The second reason has to do with off-path Quality of Service (QoS)
mechanisms. When ICE is used in environments such as PacketCable
2.0, proxies will, in addition to performing normal SIP operations,
inspect the SDP in SIP messages, and extract the IP address and port
for media traffic. They can then interact, through policy servers,
with access routers in the network, to establish guaranteed QoS for
the media flows. This QoS is provided by classifying the RTP traffic
based on 5-tuple, and then providing it a guaranteed rate, or marking
its Diffserv codepoints appropriately. When a residential NAT is
present, and a relayed candidate gets selected for media, this
relayed candidate will be a transport address on an actual TURN
server. That address says nothing about the actual transport address
in the access router that would be used to classify packets for QoS
treatment. Rather, the server reflexive candidate towards the TURN
server is needed. By carrying the translation in the SDP, the proxy
can use that transport address to request QoS from the access router.

B.4. Importance of the STUN Username

ICE requires the usage of message integrity with STUN using its short
term credential functionality. The actual short term credential is
formed by exchanging username fragments in the SDP offer/answer
exchange. The need for this mechanism goes beyond just security; it
is actual required for correct operation of ICE in the first place.

Consider agents L, R, and Z. L and R are within private enterprise 1,
which is using 10.0.0.0/8. Z is within private enterprise 2, which
is also using 10.0.0.0/8. As it turns out, R and Z both have IP
address 10.0.1.1. L sends an offer to Z. Z, in its answer, provides
L with its host candidates. In this case, those candidates are
10.0.1.1:8866 and 10.0.1.1:8877. As it turns out, R is in a session
at that same time, and is also using 10.0.1.1:8866 and 10.0.1.1:8877
as host candidates. This means that R is prepared to accept STUN
messages on those ports, just as Z is. L will send a STUN request to
10.0.1.1:8866 and and another to 10.0.1.1:8877. However, these do
not go to Z as expected. Instead, they go to R! If R just replied
to them, L would believe it has connectivity to Z, when in fact it
has connectivity to a completely different user, R. To fix this, the
STUN short term credential mechanisms are used. The username
fragments are sufficiently random that it is highly unlikely that R
would be using the same values as Z. Consequently, R would reject the
STUN request since the credentials were invalid. In essence, the
STUN username fragments provide a form of transient host identifiers,
bound to a particular offer/answer session.

An unfortunate consequence of the non-uniqueness of IP addresses is
that, in the above example, R might not even be an ICE agent. It
could be any host, and the port to which the STUN packet is directed
could be any ephemeral port on that host. If there is an application
listening on this socket for packets, and it is not prepared to
handle malformed packets for whatever protocol is in use, the
operation of that application could be affected. Fortunately, since
the ports exchanged in SDP are ephemeral and usually drawn from the
dynamic or registered range, the odds are good that the port is not
used to run a server on host R, but rather is the agent side of some
protocol. This decreases the probability of hitting an allocated
port, due to the transient nature of port usage in this range.
However, the possibility of a problem does exist, and network
deployers should be prepared for it. Note that this is not a problem
specific to ICE; stray packets can arrive at a port at any time for
any type of protocol, especially ones on the public Internet. As
such, this requirement is just restating a general design guideline
for Internet applications - be prepared for unknown packets on any
port.

B.5. The Candidate Pair Sequence Number Formula

The sequence number for a candidate pair has an odd form. It is:

   pair priority = 2^32*MIN(G,D) + 2*MAX(G,D) + (G>D?1:0)

Why is this? When the candidate pairs are sorted based on this
value, the resulting sorting has the MAX/MIN property. This means

that the pairs are first sorted based on decreasing value of the
maximum of the two sequence numbers. For pairs that have the same
value of the maximum sequence number, the minimum sequence number is
used to sort amongst them. If the max and the min sequence numbers
are the same, the offerers priority is used as the tie breaker in the
last part of the expression. The factor of 2*32 is used since the
priority of a single candidate is always less than 2*32, resulting in
the pair priority being a "concatenation" of the two component
priorities. This creates the MAX/MIN sorting. MAX/MIN ensures that,
for a particular agent, a lower priority candidate is never used
until all higher priority candidates have been tried.

B.6. The remote-candidates attribute

The a=remote-candidates attribute exists to eliminate a race
condition between the updated offer and the response to the STUN
Binding Request that moved a candidate into the Valid list. This
race condition is shown in Figure 29. On receipt of message 4, agent
L adds a candidate pair to the valid list. If there was only a
single media stream with a single component, agent L could now send
an updated offer. However, the check from agent R has not yet
generated a response, and agent R receives the updated offer (message
7) before getting the response (message 9). Thus, it does not yet
know that this particular pair is valid. To eliminate this
condition, the actual candidates at R that were selected by the
offerer (the remote candidates) are included in the offer itself, and
the answerer delays its answer until those pairs validate.

```
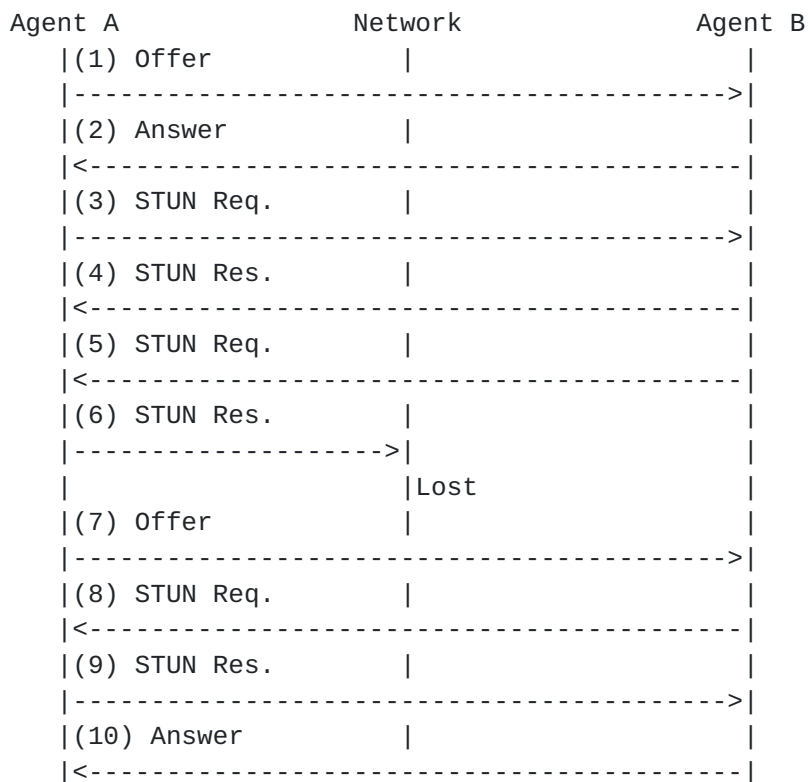         Agent A                 Network              Agent B
            |(1) Offer            |                    |
            |--------------------------------------------->|
            |(2) Answer           |                    |
            |<---------------------------------------------|
            |(3) STUN Req.        |                    |
            |--------------------------------------------->|
            |(4) STUN Res.        |                    |
            |<---------------------------------------------|
            |(5) STUN Req.        |                    |
            |<---------------------------------------------|
            |(6) STUN Res.        |                    |
            |-------------------->|                    |
            |                     |Lost                |
            |(7) Offer            |                    |
            |--------------------------------------------->|
            |(8) STUN Req.        |                    |
            |<---------------------------------------------|
            |(9) STUN Res.        |                    |
            |--------------------------------------------->|
            |(10) Answer          |                    |
            |<---------------------------------------------|
```

                   Figure 29: Race Condition Flow

B.7. Why are Keepalives Needed?

   Once media begins flowing on a candidate pair, it is still necessary
   to keep the bindings alive at intermediate NATs for the duration of
   the session. Normally, the media stream packets themselves (e.g.,
   RTP) meet this objective. However, several cases merit further
   discussion. Firstly, in some RTP usages, such as SIP, the media
   streams can be "put on hold". This is accomplished by using the SDP
   "sendonly" or "inactive" attributes, as defined in RFC 3264
   [RFC3264]. RFC 3264 directs implementations to cease transmission of
   media in these cases. However, doing so may cause NAT bindings to
   timeout, and media won't be able to come off hold.

   Secondly, some RTP payload formats, such as the payload format for
   text conversation [RFC4103], may send packets so infrequently that
   the interval exceeds the NAT binding timeouts.

   Thirdly, if silence suppression is in use, long periods of silence
   may cause media transmission to cease sufficiently long for NAT
   bindings to time out.

   For these reasons, the media packets themselves cannot be relied

upon. ICE defines a simple periodic keepalive that operates
independently of media transmission. This makes its bandwidth
requirements highly predictable, and thus amenable to QoS
reservations.

B.8. Why Prefer Peer Reflexive Candidates?

Section 4.1.2 describes procedures for computing the priority of
candidate based on its type and local preferences. That section
requires that the type preference for peer reflexive candidates
always be higher than server reflexive. Why is that? The reason has
to do with the security considerations in Section 18. It is much
easier for an attacker to cause an agent to use a false server
reflexive candidate than it is for an attacker to cause an agent to
use a false peer reflexive candidate. Consequently, attacks against
address gathering with Binding requests are thwarted by ICE by
preferring the peer reflexive candidates.

B.9. Why Send an Updated Offer?

Section 11.1 describes rules for sending media. Both agents can send
media once ICE checks complete, without waiting for an updated offer.
Indeed, the only purpose of the updated offer is to "correct" the SDP
so that the default destination for media matches where media is
being sent based on ICE procedures (which will be the highest
priority nominated candidate pair).

This begs the question - why is the updated offer/answer exchange
needed at all? Indeed, in a pure offer/answer environment, it would
not be. The offerer and answerer will agree on the candidates to use
through ICE, and then can begin using them. As far as the agents
themselves are concerned, the updated offer/answer provides no new
information. However, in practice, numerous components along the
signaling path look at the SDP information. These include entities
performing off-path QoS reservations, NAT traversal components such
as ALGs and Session Border Controllers (SBCs) and diagnostic tools
that passively monitor the network. For these tools to continue to
function without change, the core property of SDP - that the
existing, pre-ICE definitions of the addresses used for media - the m
and c lines and the rtcp attribute - must be retained. For this
reason, an updated offer must be sent.

B.10. Why are Binding Indications Used for Keepalives?

Media keepalives are described in Section 10. These keepalives make
use of STUN when both endpoints are ICE capable. However, rather
than using a Binding Request transaction (which generates a
response), the keepalives use an Indication. Why is that?

The primary reason has to do with network QoS mechanisms. Once media
begins flowing, network elements will assume that the media stream
has a fairly regular structure, making use of periodic packets at
fixed intervals, with the possibility of jitter. If an agent is
sending media packets, and then receives a Binding Request, it would
need to generate a response packet along with its media packets.
This will increase the actual bandwidth requirements for the 5-tuple
carrying the media packets, and introduce jitter in the delivery of
those packets. Analysis has shown that this is a concern in certain
layer 2 access networks that use fairly tight packet schedulers for
media.

Additionally, using a Binding Indication allows integrity to be
disabled, allowing for better performance. This is useful for large
scale endpoints, such as PSTN gateways and SBCs.

B.11. Why is the Conflict Resolution Mechanism Needed?

When ICE runs between two peers, one agent acts as controlled, and
the other as controlling. Rules are defined as a function of
implementation type and offerer/answerer to determine who is
controlling and who is controlled. However, the specification
mentions that, in some cases, both sides might believe they are
controlling, or both sides might believe they are controlled. How
can this happen?

The condition when both agents believe they are controlled shows up
in third party call control cases. Consider the following flow:

```
            A           Controller          B
            |(1) INV()      |                |
            |<------------|                  |
            |(2) 200(SDP1) |                 |
            |------------>|                   |
            |              |(3) INV()      |
            |              |------------->|
            |              |(4) 200(SDP2) |
            |              |<-------------|
            |(5) ACK(SDP2) |                 |
            |<------------|                   |
            |              |(6) ACK(SDP1) |
            |              |------------->|
```

Figure 30: Role Conflict Flow

This flow is a variation on flow III of RFC 3725 [RFC3725]. In fact,

it works better than flow III since it produces fewer messages. In
this flow, the controller sends an offerless INVITE to agent A, which
responds with its offer, SDP1. The agent then sends an offerless
INVITE to agent B, which it responds to with its offer, SDP2. The
controller then uses the offer from each agent to generate the
answers. When this flow is used, ICE will run between agents A and
B, but both will believe they are in the controlling role. With the
role conflict resolution procedures, this flow will function properly
when ICE is used.

At this time, there are no documented flows which can result in the
case where both agents believe they are controlled. However, the
conflict resolution procedures allow for this case, should a flow
arise which would fit into this category.

Author's Address

Jonathan Rosenberg
Cisco
Edison, NJ
US

Phone: +1 973 952-5000
Email: jdrosen@cisco.com
URI: http://www.jdrosen.net