MMUSIC
Internet-Draft
Expires: April 22, 2006

J. Rosenberg
Cisco Systems

October 19, 2005

Interactive Connectivity Establishment (ICE): A Methodology for Network Address Translator (NAT) Traversal for Offer/Answer Protocols

draft-ietf-mmusic-ice-06

Status of this Memo

This Internet-Draft will expire on April 22, 2006.

Copyright Notice

Abstract

This document describes a protocol for Network Address Translator (NAT) traversal for multimedia session signaling protocols based on the offer/answer model, such as the Session Initiation Protocol (SIP). This protocol is called Interactive Connectivity Establishment (ICE). ICE makes use of existing protocols, such as Simple Traversal of UDP Through NAT (STUN) and Traversal Using Relay NAT (TURN). ICE makes use of STUN in peer-to-peer cooperative fashion, allowing participants to discover, create and verify mutual

connectivity.

Table of Contents

1. Introduction

A multimedia session signaling protocol is a protocol that exchanges control messages between a pair of agents for the purposes of establishing the flow of media traffic between them. This media flow is distinct from the flow of control messages, and may take a different path through the network. Examples of such protocols are the Session Initiation Protocol (SIP) [3], the Real Time Streaming Protocol (RTSP) [17] and the International Telecommunications Union (ITU) H.323.

These protocols, by nature of their design, are difficult to operate through Network Address Translators (NAT). Because their purpose is to establish a flow of media packets, they tend to carry IP addresses within their messages, which is known to be problematic through NAT [18]. 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 [20], Simple Traversal of UDP through NAT (STUN) [1], Traversal Using Relay NAT [16], and Realm Specific IP [21] [22] along with session description extensions needed to make them work, such as the Session Description Protocol (SDP) [7] attribute for the Real Time Control Protocol (RTCP) [2]. 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 provides that solution for media streams established by signaling protocols based on the offer-answer model, RFC 3264 [5]. It is called Interactive Connectivity Establishment, or ICE. ICE makes use of STUN and TURN, but uses them in a specific methodology which avoids many of the pitfalls of using any one alone.

2. Terminology

Several new terms are introduced in this specification:

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 port.

Local Transport Address: A local transport address is a transport address that has been allocated from the operating system on the host. This includes transport addresses obtained through Virtual Private Networks (VPNs) and transport addresses obtained through Realm Specific IP (RSIP) [21] (which lives at the operating system level). Transport addresses are typically obtained by binding to an interface.

m/c line: The media and connection lines in the SDP, which together hold the transport address used for the receipt of media.

Derived Transport Address: A derived transport address is a transport address which is derived from a local transport address. The derived transport address is related to the associated local transport address in that packets sent to the derived transport address are received on the socket bound to its associated local transport address. Derived addresses are obtained using protocols like STUN and TURN, and more generally, any UNSAF protocol [23].

Associated Local Transport Address: When a peer sends a packet to a transport address, the associated local transport address is the local transport address at which those packets will actually arrive. For a local transport address, its associated local transport address is the same as the local transport address itself. For STUN derived and TURN derived transport addresses, however, they are not the same. The associated local transport address is the one from which the STUN or TURN transport was derived.

Peer Derived Transport Address: A peer derived transport address is a derived transport address learned from a STUN server running within a peer in a media session.

TURN Derived Transport Address: A derived transport address obtained from a TURN server.

STUN Derived Transport Address: A derived transport address obtained from a STUN server whose address has been provisioned or discovered by the UA. This, by definition, excludes Peer Derived Transport Addresses.

Candidate: A sequence of transport addresses that form an atomic set for usage with a particular media session. Here, atomic means that all of transport addresses in the candidate need to work before the candidate will be used for actual media transport. In the case of RTP, there can be one or more transport addresses per candidate. In the most common case, there are two - one for RTP, and another for RTCP. If the agent doesn't use RTCP, there would be just one. If Generic Forward Error Correction (FEC) [19] is in use, there may be more than two. The transport addresses that compose a candidate are all of the same type - local, STUN derived, TURN derived or peer derived.

Local Candidate: A candidate whose transport addresses are local transport addresses.

STUN Candidate: A candidate whose transport addresses are STUN derived transport addresses.

TURN Candidate: A candidate whose transport addresses are TURN derived transport addresses.

Peer Derived Candidate: A candidate whose transport addresses are peer derived transport addresses.

Generating Candidate: The candidate from which a peer derived candidate is derived.

Active Candidate: The candidate that is in use for exchange of media. This is the one that an agent places in the m/c line of an offer or answer.

Candidate ID: An identifier for a candidate.

Component: When a media stream, and as a consequence, its candidate, require several IP addresses and ports to work atomically, each of the constituent IP addresses and ports represents a component of that media stream. For example, RTP-based media streams typically have two components - one for RTP, and one for RTCP.

Component ID: An integer, starting with one within each candidate and incrementing by one for each component, which identifies the component.

Transport Address ID (tid): An identifier for a transport address, formed by concatenating the candidate ID with the component ID, separated by a "colon".

Candidate Pair: The combination of a candidate from one agent along with a candidate from its peer.

Native Candidate: From the perspective of each agent, the candidate in a candidate pair which represents a set of addresses obtained by that agent.

Remote Candidate: From the perspective of each agent, the candidate in a candidate pair which represents the set of addresses obtained by that agents peer.

Transport Address Pair: The combination of the transport address for one component of a candidate with the transport address of the same component for the matching candidate in a candidate pair.

Transport Address Pair ID: An identifier for a transport address pair. Formed by concatenating the native transport address ID with the remote transport address ID, separated by a "colon".

Matching Transport Address Pair: When a STUN Binding Request is received on a local transport address, the matching transport address pair is the transport address pair whose connectivity is being checked by that Binding Request.

Candidate Pair Priority Ordering: An ordering of candidate pairs based on a combination of the qvalues of each candidate and the candidate IDs of each candidate.

Candidate Pair Check Ordering: An ordering of candidate pairs that is similar to the candidate pair priority ordering, except that the active candidate appears at the top of the list, regardless of its priority.

Transport Address Pair Check Ordering: An ordering of transport address pairs that determines the sequence of connectivity checks performed for the pairs.

Transport Address Pair Count: The number of transport address pairs in a candidate pair. This is equal to the minimum of the number of transport addresses in the native candidate and the number of transport addresses in the remote candidate.

3. Overview of ICE

ICE makes the fundamental assumption that clients exist in a network of segmented connectivity. This segmentation is the result of a number of addressing realms in which a client can simultaneously be connected. We use "realms" here in the broadest sense. A realm is defined purely by connectivity. Two clients are in the same realm if, when they exchange the addresses each has in that realm, they are able to send packets to each other. This includes IPv6 and IPv4 realms, which actually use different address spaces, in addition to private networks connected to the public Internet through NAT.

The key assumption in ICE is that a client cannot know, apriori, which address realms it shares with any peer it may wish to communicate with. Therefore, in order to communicate, it has to try connecting to addresses in all of the realms.

```
          Agent A          TURN,STUN Servers          Agent B
             |(1) Gather Addresses |                     |
             |-------------------->|                     |
             |(2) Offer            |                     |
             |------------------------------------------>|
             |                     |(3) Gather Addresses |
             |                     |<--------------------|
             |(4) Answer           |                     |
             |<------------------------------------------|
             |(5) STUN Check       |                     |
             |<------------------------------------------|
             |(6) STUN Check       |                     |
             |------------------------------------------>|
             |(7) Offer            |                     |
             |------------------------------------------>|
             |(8) Answer           |                     |
             |<------------------------------------------|
             |(9) Media            |                     |
             |<------------------------------------------|
             |(10) Media           |                     |
             |------------------------------------------>|
```

Figure 1

The basic flow of operation for ICE is shown in Figure 1. Before the offerer establishes a session, it obtains local transport addresses from its operating system on as many interfaces as it has access to. These interfaces can include IPv4 and IPv6 interfaces, in addition to Virtual Private Network (VPN) interfaces or ones associated with

RSIP. It then obtains transport addresses for the media from each interface. Though the ICE framework can support any type of transport protocol, this specification only defines mechanisms for UDP. In addition, the agent obtains derived transport addresses from each local transport address using protocols such as STUN and TURN. These are paced at a fixed rate in order to limit network load and avoid NAT overload. The local and derived transport addresses are formed into candidates, each of which represents a possible set of transport addresses that might be viable for a media stream.

Each candidate is listed in a set of a=candidate attributes in the offer. Each candidate is given a priority. Priority is a matter of local policy, but typically, lowest priority would be given to transport addresses learned from a TURN server (i.e., TURN derived transport addresses). Each candidate is also assigned a distinct ID, called a candidate ID.

The agent will choose one of its candidates as its active candidate for inclusion in the connection and media lines in the offer. Media can be sent to this candidate immediately following its validation. Media is not sent without validation in order to avoid denial-of-service attacks. In particular, without ICE, an offerer can send an offer to another agent, and list the IP address and port of a target in the offer. If the agent is an automata that answers a call automatically, it will do so and then proceed to send media to the target. This provides substantial packet amplifications. ICE fixes this by using STUN-based validation of addresses.

The offer is then sent to the answerer. This specification does not address the issue of how the signaling messages themselves traverse NAT. It is assumed that signaling protocol specific mechanisms are used for that purpose. The answerer follows a similar process as the offerer followed; it obtains addresses from local interfaces, obtains derived transport addresses from those, and then groups them into candidates for inclusion in a=candidate attributes in the answer. It picks one candidate as its active candidate and places it into the m/c line in the answer.

Once the offer/answer exchange has completed, both agents pair up the candidates, and then determine an ordered set of transport address pairs. This ordering is based primarily on the priority of the candidates, with the exception of the active candidate, whose addresses are at the top of the list. Both agents start at the top of this list, beginning a connectivity check for that transport address pair. At a fixed interval, checks for the next transport address on the list begin. This results in a pacing of the connectivity checks. These connectivity checks are performed through peer-to-peer STUN requests, sent from one agent to the other. In

addition to pacing the checks out at regular intervals, the offerer will generate a connectivity check for a transport address pair when it receives one from its peer. As soon as the active candidate has been verified by the STUN checks, media can begin to flow. Once a higher priority candidate has been verified by the offerer, it ceases additional connectivity checks, and sends an updated offer which promotes this higher priority candidate to the m/c-line. That candidate is also listed in a=candidate attributes, resulting in periodic STUN keepalives through the duration of the media session.

If an agent receives a STUN connectivity check with a new source IP address and port, or a response to such a check with a new IP address and port indicated in the MAPPED-ADDRESS attribute, this new address might be a viable candidate for the receipt of media. This happens when there is a symmetric NAT between the agents. In such a case, the agents algorithmically construct a new candidate. Like other candidates, connectivity checks begin for it, and if they succeed, its transport addresses can be used for receipt of media by promoting it to the m/c-line.

The gathering of addresses and connectivity checks take time. As a consequence, in order to have no impact on the call setup time or post-pickup delay for SIP, these offer/answer exchanges and checks happen while the call is ringing.

4. Sending the Initial Offer

When an agent wishes to begin a session by sending an initial offer, it starts by gathering transport addresses, as described in Section 7.1. This will produce a set of candidates, including local ones, STUN-derived ones, and TURN-derived ones.

This process of gathering candidates can actually happen at any time before sending the initial offer. A agent can pre-gather transport addresses, using a user interface cue (such as picking up the phone, or entry into an address book) as a hint that communications is imminent. Doing so eliminates any additional perceivable call setup delays due to address gathering.

When it comes time to offer communications, the agent determines a priority for each candidate and identifies the active candidate that will be used for receipt of media, as described in Section 7.2.

The next step is to construct the offer message. For each media stream, it places its candidates into a=candidate attributes in the offer and puts its active candidate into the m/c line. The process for doing this is described in Section 7.3. The offer is then sent.

5. Receipt of the Offer and Generation of the Answer

Upon receipt of the offer message, the agent checks if the offer contains any a=candidate attributes. If it does, the offerer supports ICE. In that case, it starts gathering candidates, as described in Section 7.1, and prioritizes them as described in Section 7.2. This processing is done immediately on receipt of the offer, to prepare for the case where the user should accept the call, or early media needs to be generated. By gathering candidates (and performing connectivity checks) while the user is being alerted to the request for communications, session establishment delays due to that gathering can be eliminated.

The agent then constructs its answer, encoding its candidates into a=candidate attributes and including the active one in the m/c-line, as described in Section 7.3. The agent then forms candidate pairs as described in Section 7.4. These are ordered as described in Section 7.5. The agent then begins connectivity checks, as described in Section 7.6. It follows the logic in Section 7.10 on receipt of Binding Requests and responses to learn new candidates from the checks themselves.

Transmission of media is performed according to the procedures in Section 7.13.

6. Processing the Answer

There are two possible cases for processing of the answer. If the answerer did not support ICE, the answer will not contain any a=candidate attributes. As a result, the offerer knows that it cannot perform its connectivity checks. In this case, it proceeds with normal media processing as if ICE was not in use. The procedures for sending media, described in Section 7.13, MUST be followed however.

If the answer contains candidates, it implies that the answerer supports ICE. The agent then forms candidate pairs as described in Section 7.4. These are ordered as described in Section 7.5. The agent then begins connectivity checks, as described in Section 7.6. It follows the logic in Section 7.10 on receipt of Binding Requests and responses to learn new candidates from the checks themselves.

Transmission of media is performed according to the procedures in Section 7.13.

7. Common Procedures

This section discusses procedures that are common between offerer and

answerer.

7.1 Gathering Candidates

An agent gathers candidates when it believes that communications is imminent. For offerers, this occurs before sending an offer (Section 4). For answerers, it occurs before sending an answer (Section 5).

Each candidate has one or more components, each of which is associated with a sequence number, starting at 1 for the first component of each candidate, and incrementing by 1 for each additional component within that candidate. These components represent a set of transport addresses for which connectivity must be validated. For a particular media stream, all of the candidates SHOULD have the same number of components. The number of components that are needed are a function of the type of media stream.

For traditional RTP-based media streams, it is RECOMMENDED that there be two components per candidate - one for RTP and one for RTCP. The component with the component ID of 1 MUST be RTP, and the one with component ID of 2 MUST be RTCP. If an agent doesn't implement RTCP, it SHOULD have a single component for the RTP stream (which will have a component ID of 1 by definition). Each component of a candidate has a single transport address.

The first step is to gather local candidates. Local candidates are obtained by binding to ephemeral ports on an interface (physical or virtual, including VPN interfaces) on the host. The process for gathering local candidates depends on the transport protocol. Procedures are specified here for UDP. Extensions to ICE that define procedures for other transport protocols MUST specify how local transport addresses are gathered.

For each UDP media stream the agent wishes to use, the agent SHOULD obtain a set of candidates (one for each interface) by binding to N ephemeral UDP ports on each interface, where N is the number of components needed for the candidate. For RTP, N is typically two. If a host has K local interfaces, this will result in K candidates for each UDP stream , requiring K*N local transport addresses.

Once the agent has obtained local candidates, it obtains candidates with derived transport addresses. The process for gathering derived candidates depends on the transport protocol. Procedures are specified here for UDP. Extensions to ICE that define procedures for other transport protocols MUST specify how derived transport addresses are gathered.

Agents which serve end users directly, such as softphones, hardphones, terminal adapters and so on, MUST implement STUN and SHOULD use it to obtain STUN candidates. These devices SHOULD implement and SHOULD use TURN to obtain TURN candidates. They MAY implement and MAY use other protocols that provide derived transport addresses, such as TEREDO [31]. Usage of STUN and TURN is at SHOULD strength to allow for provider variation. If it is not to be used, it is RECOMMENDED that it be implemented and just disabled through configuration, so that it can re-enabled through configuration if conditions change in the future.

Agents which represent network servers under the control of a service provider, such as gateways to the telephone network, media servers, or conferencing servers that are targeted at deployment only in networks with public IP addresses MAY use STUN, TURN or other similar protocols to obtain candidates.

> Why would these types of endpoints even bother to implement ICE? The answer is that such an implementation greatly facilitates NAT traversal for clients that connect to it. The ability to process STUN connectivity checks allows for clients to obtain peer-derived transport addresses that can be used by the network server to reach them without a relay, even through symmetric NAT. Furthermore, implementation of the STUN connectivity checks allows for NAT bindings along the way to be kept open. ICE also provides numerous security properties that are independent of NAT traversal, and would benefit any multimedia endpoint. See Section 13 for a discussion on these benefits.

Obtaining STUN, TURN and other derived candidates requires transmission of packets which have the effect of creating bindings on NAT devices between the client and the STUN or TURN servers. Experience has shown that many NAT devices have upper limits on the rate at which they will create new bindings. Furthermore, transmission of these packets on the network makes use of bandwidth and needs to be rate limited by the agent. As a consequence, a client SHOULD pace its STUN and TURN transactions, such that the start of each new transaction occurs at least Ta seconds after the start of the previous transaction. The value of Ta SHOULD be configurable, and SHOULD have a default of 50ms. Note that this pacing applies only to the start of a new transaction; pacing of retransmissions within a STUN or TURN transaction is governed by the retransmission rules defined in those protocols.

To obtain STUN candidates, the client takes a local UDP candidate, and for each configured STUN server, produces a STUN candidate. It is anticipated that clients may have a multiplicity of STUN servers that it discovers or is configured with in network environments where

there are multiple layers of NAT. To produce the STUN candidate from the local candidate, it follows the procedures of Section 9 of RFC 3489 for each local transport address in the local candidate. It obtains a shared secret from the STUN server and then initiates a Binding Request transaction from each local transport address to that server. The Binding Response will provide the client with its STUN derived transport address in the MAPPED-ADDRESS attribute. If the client had K local candidates, this will produce S*K STUN candidates, where S is the number of STUN servers.

It is anticipated that clients may have a multiplicity of TURN servers configured or discovered in network environments where there are multiple layers of NAT, and that layering is known to the provider of the client. To obtain TURN candidates, for each configured TURN server, the client initiates an Allocate Request transaction using the procedures of Section 8 of [16] from each transport address of a particular local candidate. The Allocate Response will provide the client with its TURN derived transport address in the MAPPED-ADDRESS attribute. Once the TURN allocations against a particular TURN server succeed from all of the transport addresses in a particular local candidate, the client SHOULD NOT attempt any further TURN allocations to that particular server from the transport addresses in any other local candidates. This is to reduce the number of bindings allocated from the NATs. Only a single TURN candidate is needed from a particular TURN server. The order in which local candidates are tried against the TURN server is a matter of local policy.

Since a client will pace its STUN and TURN allocations at a rate of one new transaction every Ta seconds, it will take a certain amount of time for these allocations to occur. It is RECOMMENDED that implementations have a configurable upper bound on the total number of such allocations they will perform before generation of their offer or answer. Any allocations not completed at that point SHOULD be abandoned, but MAY continue and be used in an updated offer once they complete. A default value of 10 is RECOMMENDED. Since the total number of allocations that could be done (based on the number of STUN servers, TURN servers and local interfaces) might exceed this value, clients SHOULD prioritize their allocations and perform higher priority ones first. It is RECOMMENDED that STUN allocations be prioritized over TURN allocations.

Once the allocations are complete, any redundant candidates are discarded. A candidate is redundant if its transport addresses for each component match the transport addresses for each component of another candidate.

7.2 Prioritizing the Candidates and Choosing an Active One

The prioritization process takes the set of candidates and associates each with a priority. This priority reflects the desire that the agent has to receive media on that address, and is assigned as a value from 0 to 1 (1 being most preferred). Priorities are ordinal, so that their significance is only meaningful relative to other candidates from that agent for a particular media stream. Candidates MAY have the same priority. However, it is RECOMMENDED that each candidate have a distinct priority. Doing so improves the efficiency of ICE.

This specification makes no normative recommendations on how the prioritization is done. However, some useful guidelines are suggested on how such a prioritization can be determined.

One criteria for choosing one candidate over another is whether or not that candidate involves the use of a relay. That is, if media is sent to that candidate, will the media first transit a relay before being received. TURN candidates make use of relays (the TURN server), as do any local candidates associated with a VPN server. When media is transited through a relay, 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 relay run by the provider. If these concerns are important, candidates with this property can be listed with lower priority.

Another criteria for choosing one candidate over another 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) [26]. It can also help with hosts that have both a native IPv6 address and a 6to4 address. In such a case, higher priority could be afforded to the native v6 address, followed by the 6to4 address, followed by a native v4 address. This allows a site to obtain and begin using native v6 addresss 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 choosing one candidate over another 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.

Another criteria for choosing one address over another is topological awareness. This is most useful for candidates which make use of relays (including TURN and VPN). In those cases, if an agent has preconfigured or dynamically discovered knowledge of the topological proximity of the relays to itself, it can use that to select closer relays with higher priority.

There may be transport-specific reasons for preferring one candidate over another. In such a case, specifications defining usage of ICE with other transport protocols SHOULD document such considerations.

Once the candidates have been prioritized, one may be selected as the active one. This is the candidate that will be used for actual exchange of media if and when its validated, until replaced by an updated offer or answer. The active candidate will also be used to receive media from ICE-unaware peers. As such, it is RECOMMENDED that one be chosen based on the likelihood of that candidate to work with the peer that is being contacted. Unfortunately, it is difficult to ascertain which candidate that might be. As an example, consider a user within an enterprise. To reach non-ICE capable agents within the enterprise, a local candidate has to be used, since the enterprise policies may prevent communication between elements using a relay on the public network. However, when communicating to peers outside of the enterprise, a TURN-based candidate from a publically accessible TURN server is needed.

Indeed, the difficulty in picking just one address that will work is the whole problem that motivated the development of this specification in the first place. As such, it is RECOMMENDED that the active candidate be a TURN derived candidate from a TURN server providing public IP addresses. Furthermore, ICE is only truly effective when it is supported on both sides of the session. It is therefore most prudent to deploy it to close-knit communities as a whole, rather than piecemeal. In the example above, this would mean that ICE would ideally be deployed completely within the enterprise, rather than just to parts of it.

An additional consideration for selection of the active candidate is the switching of media stream destinations between the initial offer and the subsequent offer. If the active candidate pair in the initial offer is be validated, media will flow once that pair is validated. When the ICE checks complete and yield a higher priority candidate pair, there will be an updated offer/answer exchange that will change the active candidate. This will result in a change in the destination of the media packets. This may also cause a different path for the media packets. That path might have different

delay and jitter characteristics. As a consequence, the jitter buffers may see a glitch, causing possible media artifacts. If these issues are a concern, the initial offer MAY omit an active candidate. In such a case, an updated offer will need to be sent immediately when communicating with an ICE-unaware agent, setting an active candidate.

There may be transport-specific reasons for selection of an active candidate. In such a case, specifications defining usage of ICE with other transport protocols SHOULD document such considerations.

7.3 Encoding Candidates into SDP

For each candidate for a media stream, the agent includes a series of a=candidate attributes as media-level attributes, one for each component in the candidate. Each candidate has a unique identifier, called the candidate-id. The candidate-id MUST be chosen randomly and contain at least 128 bits of randomness (this does not mean that the candidate-id is 128 bits long; just that it has at least 128 bits of randomness). It is chosen only when the candidate is placed into the SDP for the first time; subsequent offers or answers within the same session containing that same candidate MUST use the same candidate-id used previously.

Each component of the candidate has an identifier, called the component-id. The component-id is a sequence number. For each candidate, it starts at one, and increments by one for each component. As discussed below, ICE will perform connectivity checks such that, between a pair of candidates, checks only occur between transport addresses with the same component-id. As a consequence, if one candidate has three components, and it is paired with a candidate that has two, there will only be two transport address pairs and two connectivity checks.

ICE will work without a standardized mapping between the components of a media stream and the numerical value of the component-id. This allows ICE to be used with media streams with multiple components without development of standards around such a mapping. However, a specific mapping has been defined in this specification for RTP - component-id 1 corresponds to RTP, and component-id of 2 corresponds to RTCP. Like the candidate-id, the component-id is assigned at the time the candidate is first placed into the SDP; subsequent offers or answers within the same session containing that same candidate MUST use the same component-id used previously.

The transport, addr and port of the a=candidate attribute (all defined in Section 12) are set to the transport protocol, unicast address and port of the tranport address. A Fully Qualified Domain

Name (FQDN) for a host MAY be used in place of a unicast 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 using an A or AAAA record, and the resulting IP address is used for the remainder of ICE processing. The qvalue is set to the priority of the candidate, and MUST be the same for all components of the candidate. Each transport address also includes a password that will be used for securing the STUN connectivity checks. This password MUST be chosen randomly with 128 bits of randomness (though it can be longer than 128 bits). Like the candidate-id, it is chosen when the candidate is placed into an SDP for the first time for a particular session; subsequent offers and answers within the same session conveying the same candidate MUST use the same password. The converse is true; if a new offer is generated as part of a new multimedia session, a new password (and candidate-id) would be used even if the transport address from a previous session was being recycled.

The combination of candidate-id and component-id uniquely identify each transport address. As a consequence, each transport address has a unique identifier, called the tid. The tid is formed by concatenating the candidate-id with the component-id, separated by the colon (":"). The tid is not explicitly encoded in the SDP; it is derived from the candidate-id and component-id, which are present in the SDP. The usage of the colon as a separator allows the candidate-id and component-id to be extracted from the tid, since the colon is not a valid character for the candidate-id.

The tid gets combined, through further concatenation, with the tid of a transport address from the remote candidate (separated again by another colon) to form the username that is placed in the STUN checks between the peers. This allows the STUN message to uniquely identify the pairing whose connectivity it is checking. The tid is needed as a unique identifier because the IP address within the candidate fails to provide that uniqueness as a consequence of NAT.

Consider agents A, B, and C. A and B are within private enterprise 1, which is using 10.0.0.0/8. C is within private enterprise 2, which is also using 10.0.0.0/8. As it turns out, B and C both have IP address 10.0.1.1. A sends an offer to C. C, in its answer, provides A with its transport addresses. In this case, thats 10.0.1.1:8866 and 8877. As it turns out, B is in a session at that same time, and is also using 10.0.1.1:8866 and 8877. This means that B is prepared to accept STUN messages on those ports, just as C is. A will send a STUN request to 10.0.1.1:8866 and 8877. However, these do not go to C as expected. Instead, they go to B. If B just replied to them, A would believe it has connectivity to C, when in fact it has connectivity to a completely different user, B. To fix this, tid takes on the role of a unique identifier. C provides A with an

identifier for its transport address, and A provides one to C. A concatenates these two identifiers (with a colon between) and uses the result as the username in its STUN query to 10.0.1.1:8866. This STUN query arrives at B. However, the username is unknown to B, and so the request is rejected. A treats the rejected STUN request as if there were no connectivity to C (which is actually true). Therefore, the error is avoided.

An unfortunate consequence of the non-uniqueness of IP addresses is that, in the above example, B 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 ususally drawn from the dynamic or registered range, the odds are good that the port is not used to run a server on host B, but rather is the agent side of some protocol. This decreases the probability of hitting a port in-use, 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.

The active candidate, if there is one, is placed into the m/c lines of the SDP. For RTP streams, this is done by placing the RTP address and port into the c and m lines in the SDP respectively. If the agent is utilizing RTCP, it MUST encode its address and port using the a=rtcp attribute as defined in RFC 3605 [2]. If RTCP is not in use, the agent MUST signal that using b=RS:0 and b=RR:0 as defined in RFC 3556 [8].

If there is no active candidate, the agent MUST include an a=inactive attribute. The RTP address and port in the m/c-line is inconsequential, since it won't be used.

Encoding of candidates may involve transport protocol specific considerations. There are none for UDP. However, extensions that define usage of ICE with other transport protocols SHOULD specify any special encoding considerations.

7.4 Forming Candidate Pairs

Once the offer/answer exchange has completed, both agents will have a set of candidates for each media stream. Each agent forms a set of

candidate pairs for each media stream by combining each of its candidates with each of the candidates of its peer. Candidates can be paired up only if their transport protocols are identical. If an offer/answer exchange took place for a session comprised of an audio and a video stream, and each agent had two candidates per media stream, there would be 8 candidate pairs, 4 for audio and 4 for video. One agent can offer two candidates for a media stream, and the answer can contain three candidates for the same media stream. In that case, there would be six candidate pairs.

Each candidate has a number of components, each of which has a transport address. Within a candidate pair, the components themselves are paired up such that transport addresses with the same component ID are combined to form a transport address pair. Returning to the previous example, for each of the 8 candidate pairs, there would be two transport address pairs - one for RTP, and one for RTCP. If one candidate has more components than the other, those extra components will not be part of a transport address pair, won't be validated, and will effectively be treated as if they weren't included in the candidate pair in the first place.

The relationship between a candidate, candidate pair, transport address, transport address pair and component are shown in Figure 2. This figure shows the relationships as seen by the agent that owns the candidate with candidate ID "L". This candidate has two components with transport addresses A and B respectively. This candidate is called the native candidate, since it is the one owned by the agent in question. The candidate owned by its peer is called the remote candidate. As the figure shows, there is a single candidate pair, and two components in each candidate. The native candidate has a candidate-id of "L", and the remote candidate has a candidate-id of "R". Since the two component-ids are 1 and 2, candidate "L" has two transport addresses with transport address IDs of "L:1" and "L:2" respectively. Similarly, candidate "R" has two transport addresses with transport address IDs of "R:1" and "R:2" respectively.

Furthermore, each transport address pair is associated with an ID, the transport address pair ID. This ID is equal to the concatenation of the tid of the native transport address with the tid of the remote transport address, separated by a colon. This means that the identifiers are seen differenly for each agent. For the agent that owns candidate "L", there are two transport address pairs. One contains transport address "L:1" and "R:1", with a transport address pair ID of "L:1:R:1". The other contains transport address "L:2" and "R:2", with a transport address pair ID of "L:2:R:2". For the agent that owns candidate "R", the identifiers for these two transport address pairs are reversed; it would be "R:1:L:1" for the first one

and "R:2:L:2" for the second.

```
              ...............................................
              .                                             .
              .                                             .
              .  .............               .............  .
              .  .  tid=L:1  .               .  tid=R:1  .  .
              .  .    --     .               .    --     .  . component
     component.  .   | A|------------------------| C|    .  .   id=1
       id=1   .  .    --     .   Transport   .    --     .  .
              .  .           .    Address    .           .  .
              .  .           .     Pair      .           .  .
              .  .           .  id=L:1:R:1   .           .  .
              .  .           .               .           .  .
              .  .           .               .           .  .
              .  .  tid=L:2  .               .  tid=R:2  .  .
    component .  .    --     .               .    --     .  .
      id=2    .  .   | B|------------------------| D|         component
              .  .    --     .   Transport   .    --     .  .   id=2
              .  .           .    Address    .           .  .
              .  .           .     Pair      .           .  .
              .  .           .   id=L:2:R:2  .           .  .
              .  .           .               .           .  .
              .  .............               .............  .
              .     Native                      Remote      .
              .    Candidate                   Candidate    .
              .      id=L                        id=R       .
              .                                             .
              .                                             .
              ...............................................
```

Candidate Pair

Figure 2

If a candidate pair was created as a consequence of an offer generated by an agent, then that agent is said to be the offerer of that candidate pair and all of its transport address pairs. Similarly, the other agent is said to be the answerer of that candidate pair and all of its transport address pairs. As a consequence, each agent has a particular role, either offerer or answerer, for each transport address pair. This role is important; when a candidate pair is to be promoted to active, the offerer is the one which performs the updated offer.

7.5 Ordering the Candidate Pairs

For the same reason that the STUN and TURN allocations are paced at a rate of Ta transactions per second, so too are the connectivity checks paced, also at a rate of Ta transactions per second. However, in order to rapidly converge on a valid candidate pair that is mutually desirable, the candidate pairs are ordered, and the checks start with the candidate pair at the top of the list. Rapid convergence of ICE depends on both the offerer and answerer coming to the same conclusion on the ordering of candidate pairs.

Recall that when each candidate is encoded into SDP, it contains a qvalue between 1 and 0, with 1 being the highest priority. Peer-derived candidates, learned through the procedures described in Section 7.10 also have a priority between 0 and 1. For each media stream, the native candidates are ordered based on their qvalues, with higher q-values coming first. Amongst candidates with the same qvalue, they are ordered based on candidate ID, using lexicographic order where C1 is placed before C2, if C2 precedes C1. In other words, if the qvalues are the same, the candidates are sorted in reverse order. This is actually important; as discussed in Section 13, it allows peer-derived candidates to be preferred over native ones. The result of these two ordering rules will be an ordered list of candidates. The first candidate in this list is given a sequence number of 1, the next is given a sequence number of 2, and so on. This same procedure is done for the remote candidates. The result is that each candidate pair has two sequence numbers, one for the native candidate, and one for the remote candidate.

First, all of the candidate pairs for whom the smaller of the two sequence numbers equals 1 are taken first. Then, all of those for whom the smaller of the two sequence numbers equals 2 are taken next, and so on. Amongst those pairs that share the same value for their smaller sequence number, they are ordered by the larger of their two sequence numbers (smallest first). Amongst those pairs that share the same value for their smaller sequence number and the same value for their larger sequence number, the larger of the two candidate IDs in each pair are selected, and the pairs are lexicographically ordered in reverse by that candidate ID, largest first.

As an example, consider two agents, A and B. One offers two candidates for a media stream with candidate IDs of "g9" and "88", with q-values of 1.0 and 0.8 respectively. The other answers with three candidates with candidate IDs of "h8", "65" and "kl", with q-values of 0.3, 0.2 and 0.1 respectively. The following table shows the rank ordering of the six candidate pairs. The column labeled "Max SN" is the larger of the two sequence numbers in the candidate pair, and "Min SN" is the minimum. The column labeled "Max Cand.

ID" is the value of the larger of the two candidate IDs in the candidate pair.

Order	A Cand. ID	A Cand. q-value	A Cand. SN	B Cand. ID	B Cand. q-value	B Cand. SN	Max SN	Min SN	Max Cand. ID
1	g9	1.0	1	h8	0.3	1	1	1	h8
2	88	0.8	2	h8	0.3	1	2	1	h8
3	g9	1.0	1	65	0.2	2	2	1	g9
4	g9	1.0	1	k1	0.1	3	3	1	k1
5	88	0.8	2	65	0.2	2	2	2	88
6	88	0.8	2	k1	0.1	3	3	2	k1

This ordering is then modified slightly by taking the candidate pair corresponding to the active candidate, if there is one, and promoting it to the top of the list. This allows the current active candidate to be tested first. As discussed below, media is not sent until the corresponding candidate is verified, necessitating rapid verification of the active candidate. This modified ordering is called the candidate pair check ordering, since it reflects the order in which connectivity checks will be done. If there was no active candidate, the candidate pair check ordering and the candidate pair priority ordering will be identical.

Within each candidate pair there will be a set of transport address pairs, one for each component ID. Those pairs are ordered by component ID. The result is an absolute ordering of all transport address pairs for a media stream, sorted first by the order of their candidate pairs (with the exception of the active candidate), followed by the order of their component IDs. This ordering is called the transport address pair check ordering.

Ordering of candidates may involve transport protocol specific considerations. There are none for UDP. However, extensions that define usage of ICE with other transport protocols SHOULD specify any special ordering considerations.

7.6 Performing the Connectivity Checks

Connectivity checks are performed by sending peer-to-peer STUN Binding Requests. These checks result in a candidate progressing through a state machine that captures the progress of connectivity checks. The specific state machine and the procedures for the connectivity checks are specific to the transport protocol. This specification defines rules for UDP. Extensions to ICE that describe

other transport protocols SHOULD describe the state machine and the procedures for connectivity checks.

The set of states visited by the offerer and answerer are depicted graphically in Figure 4

```
                                         |
                                         |Start
                                         |
                                         |
                                         V
                                   +------------+
                                   |            |
                                   |            |
                                   |  Waiting   |----------------+
                                   |            |                |
                                   |            |                |
                                   +------------+                |
                                         |                       |
                                         | Timer Ta              | Get Req
                                         | --------.             | -------
                                         | Send Req              | Send Res,
                                         V                       | Send Req
                  Get Res          +------------+      Get Req   |
                  -------          |            |      -------   |
                     -             |            |      Send Res  |
                   +---------------| Testing    |-----------+    |
                   |               |            |           |    |
                   |               |            |           |    |
                   |               +------------+           |    |
                   |                     |                  |    |
                   |                     | Error            |    |
                   |                     | -----            |    |
    Timer Tr       |                     |   -              |    |
    --------       V                     V                  V    V
    Send Req +------------+        +------------+        +------------+
       +-----|            |        |            |        |            |
       |     |   Recv-    |        |            |        |   Send-    |
       |     |   Valid    |------->|  Invalid   |<-------|   Valid    |
       |     |            |        |            |        |            |
       +---->|            | Error  |            |  Error |            |
             +------------+ -----  +------------+  ----- +------------+
                   |          -          ^           -         |
                   |                     | Error               |
                   |                     | -----               |
                   |                     |   -                 |
                   |               +------------+              |
```

```
                   |               |            |              |
                   |               |            |              |
                   +-------------->|   Valid    |<-------------+
                      Get Req      |            |     Get Res
                      -------      |            |     -------
                      Send Res     +------------+        -
                                     |       ^
                                     |       |
                                     |       |
                                     +-------+
                                      Timer Tr
                                      --------
                                      Send Req
```

Figure 4

The state machine has six states - waiting, testing, Recv-Valid, Send-Valid, Valid and Invalid. Initially, all transport address pairs start in the waiting state. In this state, the agent waits for one of two events - a chance to send a Binding Request, or receipt of a Binding Request.

Since there is an instance of the state machine for each transport address pair, Binding Requests and responses need to be matched to the specific state machine for which they apply. This is done by computing the matching transport address pair for each Binding Request. This is done by examining the USERNAME of the incoming Binding Request. The USERNAME directly contains the transport address pair ID. Requests that are sent by an agent as part of the processing described here encode the transport address pair in the USERNAME. Binding Responses are matched to their requests using the STUN transaction ID, and then mapped to the transport address pair from that.

Every Ta seconds, the agent starts a new connectivity check for a transport address pair. The check is started for the first transport address pair in the transport address pair check ordered list (which will be the active candidate) that is in the Waiting state. The state machine for this transport address pair is moved to the Testing state, and the agent sends a connectivity check using a STUN Binding Request, as outlined in Section 7.7. Once a STUN connectivity check begins, the processing of the check follows the rules for STUN. Specifically, retransmits of STUN requests are done as specified in RFC 3489, and furthermore, if a transaction fails and needs to be retried, that retry can happen rapidly, as described below. It doesn't "count" against the rate limit of 1/Ta checks per second. In addition, the keepalives that are generated for a valid pair do not

count against the rate limit either. The rate limit applies strictly to the start of connectivity checks by the answerer for a transport address pair that has been newly signaled through an offer/answer exchange.

In addition, if, while in the Waiting state, an agent receives a Binding Request matching that transport address pair, and this Binding Request generates a successful response, the agent moves into the Send-Valid state, and sends a connectivity check of its own using a STUN Binding Request, as outlined in Section 7.7. If the Binding Request didn't generate a success response, there is no change in state or generation of a Binding Request.

If, while in the Testing state, the agent receives a successful response to its STUN request, it moves into the Recv-Valid state. In this state, the agent knows that packets can flow in both directions. However, its peer agent doesn't yet know that; all it knows is that it has been able to receive a packet. Thus, in this state, the agent awaits receipt of the Binding Request sent by its peer, as the response to that request is what informs its peer that packets can flow in both directions.

If, while in the Send-Valid state, the agent receives a successful response to its STUN request, it moves to the Valid state. In this state, the agent knows that packets can flow in each direction. It also knows that its peer has sent it the STUN Request whose response will demonstrate to the peer that packets can flow in each direction.

If, while in the Recv-Valid state, the agent receives a STUN Binding Request from its peer that results in a successful response, the agent moves into the Valid state. Receipt of a request whose response was not a successful one does not result in a change in state.

In any state, if the STUN transaction results in an error, the state machine moves into the invalid state.

If a transport address pair is in the Recv-Valid or Valid state, an agent MUST generate a new STUN Binding Request transaction every Tr seconds. This transaction ensures that NAT bindings for the transport address pair remain open while the candidate is under consideration. The transaction is performed as outlined in Section 7.7. These transactions can also be used to keep the bindings alive when the candidate is promoted to active, as described in Section 7.12. Tr SHOULD be configurable, and SHOULD default to 15 seconds. If the transaction results in an error, the state machine moves to the invalid state. This happens in cases where the NAT bindings expire (e.g., due to binding timeouts or NAT failures).

The candidate pair itself has a state, which is derived from the states of its transport address pairs. If at least one of the transport address pairs in a candidate pair is in the invalid state, the state of the candidate pair is considered to be invalid. If the candidate pair enters this state, an agent SHOULD move the state machines for all of the other transport address pairs in this candidate pair into the invalid state as well. This will ensure that connectivity checks never start for those transport address pairs. Furthermore, if checks are already in progress for one of those transport address pairs, the agent SHOULD cease them.

If all of the transport address pairs making up the candidate pair are Valid, the candidate pair is considered valid. If all of the transport address pairs making up the candidate pair are either Valid or Recv-Valid, and at least one is Recv-Valid, the candidate pair is considered to be Recv-Valid. If all of the transport address pairs making up the candidate pair are either Valid or Send-Valid, and at least one is Send-Valid, the candidate pair is considered to be Send-Valid. If all of the transport address pairs in a candidate pair are in the Waiting state, the candidate pair is in the waiting state. If all of the transport address pairs in the candidate pair are either in the Waiting or Testing states, and at least one is in the Testing state, the state of the candidate pair is Testing. Otherwise, the state of the candidate pair is considered Indeterminate.

A candidate itself also has a state. If a candidate is present in at least one valid candidate pair, that candidate is said to be valid. If all of the candidate pairs containing that candidate are invalid, the candidate itself is invalid. Otherwise, the candidate's state is Indeterminate.

If a native candidate becomes valid, and is more preferred than the active one, the offerer sends an updated offer with this newly validated candidate promoted to the m/c-line. This process is discussed in more detail in Section 7.9.

7.7 Sending a Binding Request for Connectivity Checks

An agent performs a Binding Request transaction by sending a STUN Binding Request from its native transport address, and sending it to the remote transport address. The meaning of "sending from its native transport address" depends on the type of transport protocol and the type of transport address (local, STUN-derived, TURN-derived, or peer-derived). This specification defines the meaning for UDP. Specifications defining other transport protocols must define what this means for them.

For UDP-based local transport addresses, sending from the local

transport address has the meaning one would expect - the request is sent such that the source IP address and port For STUN derived UDP transport addresses, it is sent by sending from the local transport address used to derive that STUN address. For TURN derived UDP transport addresses, it is sent by using TURN mechanisms to send the request through the TURN server (using the SEND primitive). Sending the request through the TURN server neccesarily requires that the request be sent from the client, using the local transport address used to derive the TURN transport address.

The Binding Request sent by the agent MUST contain the USERNAME attribute. This attribute MUST be set to the transport address pair ID of the corresponding transport address pair as seen by its peer. Thus, for the first transport address pair in Figure 2, if the agent on the left sends the STUN Binding Request, the USERNAME will have the value R:1:L:1. If the agent on the right sends the STUN Binding Request, the USERNAME will have the value L:1:R:1. To be clear, the USERNAME that is used is NOT the one seen locally, but rather the one as seen by its peer. The request SHOULD contain the MESSAGE-INTEGRITY attribute, computed according to RFC 3489 procedures. The key used as input to the HMAC is the password provided by the peer for this remote transport address. The Binding Request MUST NOT contain the CHANGE-REQUEST or RESPONSE-ADDRESS attribute.

The STUN transaction will generate either a timeout, or a response. If the response is a 420, 500, or 401, the agent should try again as described in RFC 3489 (as mentioned above, it need not wait Ta seconds to try again). Either initially, or after such a retry, the STUN transaction might produce a non-recoverable failure response (error codes 400, 430, 431, or 600) or a failure result inapplicable to this usage of STUN and thus unrecoverable (432, 433). If this happens, an error event is generated into the state machine, and the transport address pair enters the invalid state.

If the STUN transaction times out, the client SHOULD NOT retry. The only reason a retry might succeed is if there was severe packet loss during the duration of the check, or the answer was significantly delayed, also due to packet loss. However, STUN Binding Request transactions run for 9.5 seconds, which is well beyond the typical tolerance for a session establishment. The retries come with a penalty of additional traffic, which can be used to launch DoS attacks Section 13.4.2. The only reason to not follow the SHOULD NOT is if the agent has adjusted the STUN transaction timers to be more aggressive.

If the Binding Response is a 200, the agent SHOULD check for the MESSAGE-INTEGRITY attribute and verify it, as discussed in RFC 3489. Indeed, this check SHOULD be done for all responses. This will

result in the response being discarded (eventually leading to a timeout), if the integrity check fails.

7.8 Receiving a Binding Request for Connectivity Checks

As a result of providing a list of candidates in its offer or answer, an agent will receive STUN Binding Request messages. An agent MUST be prepared to receive STUN Binding Requests on each local transport address from the moment it sends an offer or answer that contains a candidate with that local transport address. Similarly, it MUST be prepared to receive STUN Binding Requests on a local transport address the moment it sends an offer or answer that contains a STUN or TURN candidate derived from a local candidate containing that local transport address. It can cease listening for STUN messages on that local transport address after sending an updated offer or answer which does not include any candidates with transport addresses that are equal to or derived from that local transport address.

The agent does not need to provide STUN service on any other IP addresses or ports, unlike the STUN usage described in [1]. The need to run the service on multiple ports is to support receipt of Binding Requests with the CHANGE-REQUEST attribute. However, that attribute is not used when STUN is used for connectivity checks. A server SHOULD reject, with a 400 answer, any STUN requests with a CHANGE-REQUEST attribute whose value is non-zero. The CHANGED-ADDRESS attribute in a BindingAnswer is set to the transport address on which the server is running.

Furthermore, there is no need to support TLS or to be prepared to receive SharedSecret request messages. Those messages are used to obtain shared secrets to be used with BindingRequests. However, with ICE, these shared secrets are exchanged through the offer/answer exchange itself.

One of the candidates may be in use as the active candidate. For the transport addresses comprising that candidate, the agent will receive both STUN requests and media packets on its associated local transport addresses. The agent MUST be able to disambiguate them. In the case of RTP/RTCP, this disambiguation is easy. RTP and RTCP packets start with the bits 0b10 (v=2). The first two bits in STUN are always 0b00. This disambiguation also works for packets sent using Secure RTP [25], since the RTP header is in the clear. Disambiguating STUN with other media stream protocols may be more complicated. However, it can always be possible with arbitrarily high probabilities by selecting an appropriately random username (see below).

Processing of the Binding Request proceeds in two steps. The first

is generation of the response, and the second is side-effect processing. Generation of the response follows the general procedures of RFC 3489. The USERNAME is considered valid if its topmost portion (the part up to, but not including the second colon) corresponds to a transport address ID known to the agent. The password associated with that transport address ID is used to verify the MESSAGE-INTEGRITY attribute, if one was present in the request. If the USERNAME was not valid, the agent generates a 430. Otherwise, the success response will include the MAPPED-ADDRESS attribute, which is used for learning new candidates, as described in Section 7.10. The MAPPED-ADDRESS attribute is populated with the source IP address and port of the Binding Request. For Binding Requests received over TURN-derived transport addresses, this MUST be the source IP address and port of the Binding Request when it arrived at the TURN relay, prior to forwarding towards the agent. That source transport address will be present in the REMOTE-ADDRESS attribute of a TURN Data Indication message, if the Binding Request were delivered through a Data Indication. 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.

The side effect processing involves changes to the state machine for a transport address pair. This processing cannot be done until the initial offer/answer exchange has completed. As a consequence, if the answerer received a Binding Request that generated a success response, but had not yet received the answer to its offer, it waits for the answer, and when it arrives, then performs the side effect processing.

The agent takes the entire contents of the USERNAME, and compares them against the transport address pair identifiers as seen by that agent for each transport address pair. If there is no match, nothing is done - this should never happen for compliant implementations. If there is a match, the resulting transport address pair is called the matching transport address pair. The state machine for the matching transport address pair is then updated based on the receipt of a STUN Binding Request, and the resulting actions described in Section 7.6 are undertaken.

An agent will continue to receive periodic STUN transactions on a local transport address as long as it had listed that transport address, or one derived from it, in an a=candidate attribute in its most recent offer or answer, and the state machine indicates that Binding Requests are periodically sent (as is the case for UDP). It MUST process any such transactions according to this section. It is possible that a transport address pair that was previously valid may become invalidated as a result of a subsequent failed STUN transaction.

7.9 Promoting a Candidate to Active

As a consequence of the connectivity checks, each agent will change the states for each transport address pair, and consequently, for the candidate pairs. When a candidate pair becomes valid, and the agent is in the role of offerer for that candidate pair, the agent follows the logic in this section. The rules only apply to the offerer of a candidate pair in order to eliminate the possibility of both agents simultaneously offering an update to promote a candidate to active.

If this candidate pair is the first one in the candidate pair priority ordered list, the agent SHOULD send an updated offer as described in Section 7.11.1. If this candidate pair is not the first on that list, but it is the first on the candidate pair check ordered list, it means that this candidate pair is the active one, and its connectivity has been verified. This is good news; the currently active candidate is working. Media can now flow as described in Section 7.13 (media will never flow prior to validation). However, no updated offer is sent at this time.

If this candidate pair is not the first on the candidate pair priority ordered list or the candidate pair check ordered list, and the wait-state timer has not yet been set, the agent sets this timer to Tws seconds. Tws SHOULD be configurable, and SHOULD have a default of 100ms. This timer allows for a higher priority connectivity check to complete, in the event its STUN Binding Request was lost or delayed in the network. If, prior to the wait-state timer firing, another connectivity check completes and a candidate pair is validated, there is no need to reset or cancel the timer. Once the timer fires, the agent SHOULD issue an updated offer as described in Section 7.11.1.

7.10 Learning New Candidates from Connectivity Checks

ICE makes use of candidate addresses learned through protocols like STUN, as described in Section 7.1. These addresses are learned when STUN requests are sent to configured STUN servers. However, the peer-to-peer STUN connectivity checks can themselves provide additional candidates that ICE can make use of. This happens, for example, when two agents are separated by a symmetric NAT. When the agent behind the symmetric NAT sends a Binding Request to the other agent (which can have a public address or be behind any type of NAT except for symmetric), the symmetric NAT will create a new NAT binding for this Binding Request. Because of the properties of symmetric NAT, that binding can be used be the agent on the public side of the symmetric NAT to send packets back to the agent behind the symmetric NAT.

To do this, ICE agents perform additional processing on the receipt of STUN Binding Requests and responses, beyond the logic described in Section 7.7 and Section 7.8. This logic is described below.

7.10.1 On Receipt of a Binding Request

When a STUN Binding Request is received which generates a success response, that Binding Request would have been associated with a matching transport address pair and corresponding candidate pair. The source IP and port of this Binding Request are compared to the IP address and port of the remote transport address in the matching transport address pair. Note that, in this case, we are comparing actual IP addresses and ports - not tids. In addition, if the Binding Request arrived through a TURN derived transport address, the source IP and port of this binding request used for the comparison are those in the Binding Request when it arrived at the TURN relay, prior to forwarding towards the agent. That source transport address will be present in the REMOTE-ADDRESS attribute of a TURN Data Indication message, if the Binding Request were delivered through a Data Indication. 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.

The comparison of the source IP and port of the Binding Request and the IP address and port of the remote transport address in the matching transport address pair may not match. One reason this could happen is if there was a NAT between the two agents. If they do not match, the source IP and port of the Binding Request (and again, for TURN derived transport address, this refers to the source IP address and port of the packet when it arrived at the relay) are compared to the IP address and ports across the transport address pairs in *all* remote candidates. If there is still no match, it means that the source IP and port might represent another valid remote transport address. Such a transport address is called a peer-derived transport address.

To use it, that address needs to be associated with a candidate (called a peer-derived candidate). In this case, however, the candidate isn't signaled through an offer/answer exchange; it is constructed dynamically from information in the STUN request. Like all other candidates, the peer-derived candidate has a candidate ID. The candidate ID is derived from the candidate IDs of the matching candidate pair. In particular, the candidate ID is constructed by concatenating the remote candidate ID with the native candidate ID (without the colon).

On receipt of a STUN Binding Request whose source IP and port don't match the transport address in any remote candidate, the agent

constructs the candidate ID that represents the peer-derived candidate, and checks to see if that candidate exists. It may already exist if it had been constructed as a consequence of a previous application of this logic on receipt of a Binding Request for a different transport address pair of the same candidate pair. If there is not yet a peer derived candidate with that candidate ID, the agent creates it, and assigns it the newly computed candidate ID. The priority of the peer-derived candidate MUST be set to the priority of its generating candidate - the remote candidate in the matching transport address pair. Note that, at this time, the peer derived candidate has no transport addresses in it.

Newly created or not, the agent extracts the component ID from the matching transport address pair, and sees if a transport address with that same component ID exists in the peer derived candidate. If not (and it shouldn't), the agent adds a transport address to the peer-derived candidate. This transport address is equal to the source IP address and port from the incoming STUN Binding Request. It is assigned the component ID equal to the component ID in the matching transport address pair. This transport address will have a tid, equal to the concatenation of the candidate ID for this new candidate, and the component ID, separated by a colon.

The peer-derived candidate becomes usable once the number of transport addresses in it equals the transport address pair count of the candidate pair from which it is derived. Initially, the peer-derived candidate will start with a single transport address. More are added as the connectivity checks for the original candidate pair take place. Once the peer-derived candidate becomes usable, it has to be paired up with native candidates. However, unlike the procedures of Section 7.5, which pair up each remote candidate with each native candidate, this peer-derived candidate is only paired up with the native candidate from the candidate pair from which it was derived. This creates a new candidate pair, and a set of new transport address pairs.

Recall that, for each candidate pair, one agent plays the role of offerer, and the other of answerer. For peer-derived candidates, the agent that receives the STUN request and follows the processing in this section acts as the answerer.

Figure 5 provides a pictorial representation of the peer derived candidate (the one with id=RL) and its pairing with the native candidate with id L. The candidate with ID R is referred to as the generating candidate. The peer-derived candidate is effectively an alternate for that generating candidate, but is only paired with a specific native candidate. Note that, for a particular generating candidate, there can be many peer derived candidates, up to one for

each native candidate.

```
                 .............               .............
                 .  tid=L:1  .               .  tid=R:1  .
        component.    --     .    id=L:1:R:1 .    --     .component
          id=1   .   | A|------------------------| C|    .  id=1
                 .    -- -------+            .    --     .
                 .           .  |            .           .
                 .           .  |            .           .   Generating
                 .           .  |            .           .   Candidate
                 .           .  |            .           .
                 .           .  |            .           .
                 .  tid=L:2  .  |            .  tid=R:2  .
        component.    --     .  | id=L:2:R:2 .    --     .component
          id=2   .   | B|-------C----------------| D|    .  id=2
                 .    -- -----+ |            .    --     .
                 .           .| |            .           .
                 .           .| |            .           .
                 .           .| |            .           .
                 .           .| |            .           .
                 .............| |            .............
                    Native    | |               Remote
                   Candidate  | |              Candidate
                     id=L     | |                id=R
                              | |
                              | |
                             .| |
                              | |
                              | |
                              | |
                              | |             .............
                              | |             .  tid=RL:1 .
                              | | id=L:1:RL:1 .    --     .component
                              | +-----------------| C|    .  id=1
                              |               .    --     .
                              |               .           .
                              |               .           .   Peer Derived
                              |               .           .   Candidate
                              |               .           .
                              |               .           .
                              |               .  tid=RL:2 .
                              |   id=L:2:RL:2 .    --     .component
                              +-------------------| D|    .  id=2
                                              .    --     .
                                              .           .
                                              .           .
                                              .           .
```

```
                                              .           .
                                              .............
                                                 Remote
                                                Candidate
                                                  id=RL
```

Figure 5

The new transport address pairs have a state machine associated with them. The state that is entered, and actions to take as a consequence, are specific to the transport protocol. For UDP, the procedures are defined here. Extensions that define processing for other transport protocols SHOULD describe the behavior.

For UDP, the state machine enters the Send-Valid state. Effectively, the Binding Request just received "counts" as a validation in this direction, even though it was formally done for a different candidate pair. In addition, the agent SHOULD generate a Binding Request for each transport address in this new candidate pair, as described in Section 7.7. The transport address pairs are inserted into the ordered list of pairs based on the ordering described in Section 7.5 and processing follows the logic described in Section 7.6.

7.10.2 On Receipt of a Binding Response

The procedures on receipt of a Binding Response are nearly identical to those for receipt of a Binding Request as described above.

When a successful STUN Binding Response is received, it will be associated with a matching transport address pair and corresponding candidate pair. This matching is done based on comparison of candidate IDs. The value of the MAPPED-ADDRESS attribute of the Binding Response are compared to the IP address and port of the native transport address in the matching transport address pair. Note that, in this case, we are comparing actual IP addresses and ports - not tids. These may not match if there was a NAT between the two agents. If they do not match, the value of the MAPPED-ADDRESS attribute of the Binding Response are compared to the IP address and ports across the transport address pairs in *all* native candidates. If there is still no match, it means that the MAPPED-ADDRESS might represent another valid remote transport address.

To use it, that address needs to be associated with a candidate. In this case, however, the candidate isn't signaled through an offer/answer exchange; it is constructed dynamically from information in the STUN response. Such a candidate is called a peer-derived candidate. Like all other candidates, the peer-derived candidate has a candidate ID. The candidate ID is derived from the candidate IDs

of the matching candidate pair. In particular, the candidate ID is constructed by concatenating the native candidate ID with the remote candidate ID (without the colon).

On receipt of a STUN Binding Response whose MAPPED-ADDRESS didn't match the transport address in any native candidate, the agent constructs the candidate ID that represents the peer-derived candidate, and checks to see if that candidate exists. It may already exist if it had been constructed as a consequence of a previous application of this logic on receipt of a Binding Response for a different transport address pair of the same candidate pair. If there is not yet a peer derived candidate with that candidate ID, the agent creates it, and assigns it the newly computed candidate ID. The priority of the new candidate MUST be set to the priority of the generating candidate - the native candidate in the matching transport address pair. Note that, at this time, the peer derived candidate has no transport addresses in it.

Newly created or not, the agent extracts the component ID from the matching transport address pair, and sees if a transport address with that same component ID exists in the peer derived candidate. If not (and it shouldn't), the agent adds a transport address to the peer-derived candidate. This transport address is equal to the MAPPED-ADDRESS from the STUN Binding Response. It is assigned the component ID equal to the component ID in the matching transport address pair. This transport address will have a tid, equal to the concatenation of the candidate ID for this new candidate, and the component ID, separated by a colon.

The peer-derived candidate becomes usable once the number of transport addresses in it equals the transport address pair count of candidate pair from which it is derived. Initially, the peer-derived candidate will start with a single transport address. More are added as the connectivity checks for the original candidate pair take place. Once the peer-derived candidate becomes usable, it has to be paired up with remote candidates. However, unlike the procedures of Section 7.5, which pair up each remote candidate with each native candidate, the peer-derived candidate is only paired up with the remote candidate from the matching candidate pair . This creates a new candidate pair, and a set of new transport address pairs.

Recall that, for each candidate pair, one agent plays the role of offerer, and the other of answerer. For peer-derived candidates, the agent that receives the STUN request and follows the processing in this section acts as the answerer.

The new transport address pairs have a state machine associated with them. The state that is entered, and actions to take as a

consequence, are specific to the transport protocol. For UDP, the procedures are defined here. Extensions that define processing for other transport protocols SHOULD describe the behavior.

For UDP, the state machine enters the Recv-Valid state. Effectively, the Binding Response just received "counts" as a validation in this direction, even though it was formally done for a different candidate pair. The transport address pairs are inserted into the ordered list of pairs based on the ordering described in Section 7.5, and processing follows the logic described in Section 7.6.

7.11 Subsequent Offer/Answer Exchanges

An agent MAY issue an updated offer at any time. This updated offer may be sent for reasons having nothing to do with ICE processing (for example, the addition of a video stream in a multimedia session), or it may be due to a change in ICE-related parameters. For example, if an agent acquires a new candidate after the initial offer/answer exchange, it may seek to add it.

However, agents SHOULD follow the logic described in Section 7.9 to determine when to send an updated offer as a consequence of promoting a candidate to active.

If there are any aspects of this processing that are specific to the transport protocol, those SHOULD be called out in ICE extensions that define operation with other transport protocols. There are no additional considerations for UDP.

7.11.1 Sending of a Subsequent Offer

The offer MAY contain a new active candidate in the m/c line. This candidate SHOULD be the native candidate from the highest candidate pair in the candidate pair priority ordered list whose state is valid. If there are no candidate pairs in this state, the highest one whose state is partially valid SHOULD be used. If there are no candidate pairs in this state, the candidate pair that is most likely to work with this peer, as described in Section 7.2, SHOULD be used. The candidate is encoded into the m/c line in an updated offer as described in Section 7.3.

If the candidate pair whose native candidate was encoded into the m/c-line was valid or partially valid, the agent MUST include an a=remote-candidate attribute into the offer. This attribute MUST contain the candidate ID of the remote candidate in the candidate pair. It is used by the recipient of the offer in selecting its candidate for the answer.

The meaning of a=candidate attributes within a subsequent offer have the same meaning as they do in an initial offer. They are a request for the peer to attempt (or continue to attempt if the candidate was provided previously) a connectivity check using STUN from each of its own candidates. When an updated offer is sent, there are several dispositions regarding the candidates:

retained: A candidate is retained if the candidate ID for the candidate is included in the new offer, and matches the candidate ID for a candidate in the previous offer or answer. In this case, all of the information about the candidate - its qvalue and components, and the IP addresses, ports, STUN passwords and transport protocols of its components, MUST be the same as the previous offer or answer from the agent. If the agent wants to change them, this is accomplished by changing the candidate ID as well. That will have the effect of removing the old candidate and adding a new one with the updated information.

removed: A candidate is removed if its candidate ID appeared in a previous offer or answer, and that candidate ID is not present in the new offer.

added: A candidate is added if its candidate ID appeared in the new offer, but was not present in a previous offer or answer from that agent.

The following rules are used to determine the disposition of the each of the current native candidates in the new offer:

- If a candidate is invalid, and all peer-derived candidates generated from it are invalid as well, it SHOULD be removed.

- If the candidate in the m/c-line is valid, all other candidates SHOULD be removed. This has the effect of stopping connectivity checks of other candidates. This SHOULD would not be followed if an agent wanted to keep a candidate ready for usage should, for some reason, the active candidate later become invalid.

- If the candidate in the m/c-line is valid, and it is not peer-derived, that candidate MUST be retained. If the candidate in the m/c-line is peer-derived, its generating candidate MUST be retained, even if it is itself invalid.

- If the candidate in the m/c-line has not been validated, all other candidates that are not invalid, or candidates for whom their derived candidates are not invalid, SHOULD be retained.

- Peer derived candidates MUST NOT be added; they continue to be used as long as their generating candidate was retained. Peer derived candidates are learned exclusively through the STUN connectivity checks.

A new candidate MAY be added. This can happen when the candidate is a new one, learned since the previous offer/answer exchange, and it has a higher priority than the currently active candidate. It can also occur when an agent wishes to restart checks for a transport address it had tried previously. Effectively, changing the candidate ID value in an updated offer will "restart" connectivity checks for that candidate.

If a candidate is removed, the agent takes the following steps:

1. The agent eliminates any candidate pairs whose native candidate equalled the candidate that was removed. Equality is based on comparison of candidate IDs.

2. The agent eliminates any candidate pairs that had a native candidate that is a peer derived candidate generated from the candidate that was removed.

3. The candidate pairs that are eliminated are removed from the candidate pair priority ordered list and candidate pair check ordered list. As a consequence of this, if connectivity checks had not yet begun for the candidate pair, they won't.

4. If connectivity checks were already in progress for transport addresses in that candidate pair, the agent SHOULD immediately terminate them. No further retransmissions take place, and no further transactions from that candidate will be made.

5. If the removed candidate was a TURN-derived candidate, the agent SHOULD de-allocate its transport addresses from the TURN server. If a local candidate was removed, and all of its derived candidates were also removed (including any peer-derived candidates), local operating system resources for each of the transport addresses in the local candidate SHOULD be de-allocated.

7.11.2 Receiving the Offer and Sending an Answer

To generate the answer, the answerer has to decide which transport addresses to include in the m/c line, and which to include in candidate attributes.

Rules for choosing transport addresses for the m/c-line are as follows. The agent examines the transport addresses in the m/c-line of the offer. It compares these with the transport addresses in the remote candidates of candidate pairs whose states are Valid. If there is matching candidate pair in that state, the agent MUST pick the native candidate from one of those pairs, and use that candidate as the active one. If none of the matching pairs are in the Valid state, the agent checks if there are any matching pairs in the Send-Valid state. If there are, the agent looks for the a=remote-candidate attribute in the offer. If present, and the candidate ID listed there is one of the native candidate IDs amongst the matching pairs, that candidate ID MUST be used as the active one. If the a=remote-candidate attribute was not present in the offer, or there were no matching candidate pairs in the Send-Valid state, the candidate that is most likely to work with this peer, as described in Section 7.2, SHOULD be used.

The a=remote-candidate 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 state. If the answer arrives at the agent prior to the Binding Response, the candidate pair that was validated by the offer will still be in the Send-Valid state. To eliminate this condition, the identity of the validated candidate is included in the offer itself.

Like the offerer, the answer can decide, for each of its candidates, whether they are retained or removed. The same rules defined in Section 7.11.1 for determining their disposition apply to the answerer. Similarly, if a candidate is removed, the same rules in Section 7.11.1 regarding removal of canididate pairs and freeing of resources apply.

Once the answer is sent, the answerer will have the set of native and remote candidates before this offer/answer exchange, and the set of native and remote candidates afterwards. The agent then pairs up the native and remote candidates which were added or retained. Furthermore, for candidate pairs containing a peer derived transport address, those pairs continue as long as both candidates are retained. A peer derived candidate continues to be used as long as its generating parent continues to be used. This leads to a set of current candidate pairs.

If a candidate pair existed previously, but as a consequence of the offer/answer exchange, either its native or remote candidate has been removed, the agent takes the following steps:

1. The candidate pair is removed from the candidate pair priority ordered list and candidate pair check ordered list. As a

consequence of this, if connectivity checks had not yet begun for the candidate pair, they won't.

2. If connectivity checks were already in progress for that candidate pair, the agent SHOULD immediately terminate any STUN transactions in progress from that candidate. No further retransmissions take place, and no further transactions from that candidate will be made.

3. If the agent receives a STUN Binding Request for that candidate pair, the agent SHOULD generate a 430 response.

If a candidate pair existed previously, and continues to exist, no changes are made; any STUN transactions in progress for that candidate pair continue, and it remains on the candidate pair priority ordered list and candidate pair check ordered list.

If a candidate pair is new (because either its native candidate is new, or its remote candidate is new, or both), the agent takes the role of answerer for this candidate pair. The new candidate pair is inserted into the candidate pair priority ordered list and candidate pair check ordered list. STUN connectivity checks will start for them based on the logic described in Section 7.6.

7.11.3 Receiving the Answer

Once the answer is received, the answerer will have the set of native and remote candidates before this offer/answer exchange, and the set of native and remote candidates afterwards. It then follows the same logic described in Section 7.11.2, pairing up the candidate pairs, removing ones that are no longer in use, and beginning of processing for ones that are new.

7.12 Binding Keepalives

Once a candidate is promoted to active, and media begins flowing, 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 [5]. 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 [34], 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.

To prevent these problems, ICE implementations MUST continue to list their active transport addresses in a=candidate lines for UDP-based media streams. As a consequence of this, STUN packets will be transmitted periodically independently of the transmission (or lack thereof) of media packets. This provides a media independent, RTP independent, and codec independent solution for keeping the NAT bindings alive. STUN Binding Requests cannot be used for TCP-based transports because the media protocol may not provide framing services to support this. As such, application layer keepalives MUST be used in this case.

If an ICE implementation is communciating with one that does not support ICE, keepalives MUST still be sent. Indeed, these keepalives are essential even if neither endpoint implements ICE. As such, this specification defines keepalive behavior generally, for endpoints that support ICE, and those that do not.

All endpoints MUST send keepalives for each media session. These keepalives MUST be sent regardless of whether the media stream is currently inactive, sendonly, recvonly or sendrecv. 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 communicating with a peer that supports ICE. An agent can determine that its peer supports ICE by the presence of the 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 [29] and RTP comfort noise [27].

STUN-based keepalives will be sent periodically every Tr seconds as a consequence of the rules in in Section 7.7. If STUN keepalives are not in use (because the peer does not support ICE or because of TCP), an agent SHOULD ensure that a media packet is sent every Tr seconds. If one is not sent as a consequence of normal media communications, a keepalive packet using one of the formats discussed above SHOULD be sent.

7.13 Sending Media

An agent MUST NOT send media packets until the active candidate has

entered either the Valid or Recv-Valid state. This is to prevent a particularly destructive denial-of-service attack described in Section 13.4.1.

It is important to note that an agent always sends media to the address in the m/c-line, not to a validated candidate. To use a candidate, it must be promoted to the m/c-line through an updated offer/answer exchange.

When an agent sends media packets, it MUST send them from the same IP address and port it has advertised in the m/c-line. This provides a property known as symmetry, which is an essential facet of NAT traversal.

In the case of a STUN-derived transport address, this means that the RTP packets are sent from the local transport address used to obtain the STUN address. In the case of a TURN-derived transport address, this means that media packets are sent through the TURN server (using the TURN SEND primitive). For local transport addresses, media is sent from that local transport address.

This symmetric behavior MUST be followed by an agent even if its peer in the session doesn't support ICE.

8. Guidelines for Usage with SIP

SIP [3] makes use of the offer/answer model, and is one of the primary targets for usage of ICE. SIP allows for offer/answer exchanges to occur in many different combinations of messages, including INVITE/200 OK and 200 OK/ACK. When support for reliable provisional responses (RFC 3262 [13]) and UPDATE (RFC 3311 [28]) are added, additional combinations of messages that can be used for offer/answer exchanges are added. As such, this section provides some guidance on good ways to make use of SIP with ICE.

ICE requires a series of STUN-based connectivity checks to take place between endpoints, along with an updated offer/answer exchange to use a validated candidate. These exchanges require time to complete. If the initial offer/answer exchange were to take place in the INVITE and 200 OK response respectively, the connectivity checks and updated offer would all occur after the called party answered. This will result in a potential increase in the post-pickup delay. This delay refers to the time between when a user "answers the phone" and when any speech they utter can be delivered to the caller.

To eliminate any increase in post-pickup delay due to ICE, it is RECOMMENDED that the initial offer/answer exchange take place in an INVITE and a 18x provisional response. As a consequence, support for

RFC 3262 is RECOMMENDED with ICE. The STUN connectivity checks will then take place while the called party is being "rung". To deliver the updated offer prior to the user answering the call, it is RECOMMENDED that it be delivered with an UPDATE request. This will allow ICE to have completed prior to the called party even answering the session invitation.

If RFC 3262 and RFC 3311 are not supported by both agents, tuning can still take place to reduce post-pickup delays. In particular, the answerer SHOULD include its answer in an unreliable 18x response. RFC 3261 requires that the same answer also be placed in a 200 OK, which is delivered reliably. However, placing it in a 18x gives the offerer an early preview of the answer, and allows the connectivity checks to all occur prior to the user answering the call. However, the updated offer with the highest priority valid candidate promoted to the m/c-line cannot occur until after the 200 OK, in which case it SHOULD be done with a re-INVITE. Fortunately, if the active candidates in the initial offer/answer exchange end up being valid anyway, media can flow as soon as the user answers the call (or even before hand, if early media is needed). The additional offer/answer exchange in the re-INVITE would merely improve the situation by using a higher priority candidate pair.

One of the difficulties in including the answer in the 18x, and then using it for connectivity checks, is that the 18x might be lost. In such a case, the STUN connectivity check from the answerer to the offerer (UAS to UAC) will pend indefinitely. To prevent this, it is RECOMMENDED that a SIP UA retransmit its 18x periodically, using the same exponential backoff defined in RFC 3262, until such time as a Binding Response is received for any of the Binding Requests it sent.

As discussed in Section 13, offer/answer exchanges SHOULD be secured against eavesdropping and man-in-the-middle attacks. To do that, the usage of SIPS is RECOMMENDED when used in concert with ICE.

9. Interactions with Forking

SIP allows INVITE requests carrying offers to fork, which means that they are delivered to multiple user agents. Each of those user agents then provides an answer to the offer in the INVITE. The result is that a single offer generated by the UAC produces multiple answers.

ICE interacts very well with forking. Indeed, ICE fixes some of the problems associated with forking. Once the offer/answer exchange has completed, the UAC will have an answer from each UAS that received the INVITE. The ICE connectivity checks that ensue will carry transport address pair IDs that correlate each of those checks (and

thus their corresponding IP addresses and ports) with a specific remote user agent. As these checks happen before any media is transmitted, ICE allows a UAC to disambiguate subsequent media traffic by looking at the source IP address and port, and then correlate that traffic with a particular remote UA. When SIP is used without ICE, the incoming media traffic cannot be disambiguated without an additional offer/answer exchange.

10. Interactions with Preconditions

Because ICE involves multiple addresses and pre-session activities, its interactions with preconditions merits further discussion.

Quality of Service (QoS) preconditions, which are defined in RFC 3312 [9] and RFC 4032 [10], apply only to the IP addresses and ports listed in the m/c lines in an offer/answer. If ICE changes the address and port where media is received, this change is reflected in the m/c lines of a new offer/answer. As such, it appears like any other re-INVITE would, and is fully treated in RFC 3312 and 4032, which applies without regard to the fact that the m/c lines are changing due to ICE negotiations ocurring "in the background".

ICE also has (purposeful) interactions with connectivity preconditions [14]. Those interactions are described there.

11. Example

This section provides an example ICE call flow. Two agents, L and R, are using ICE. Both agents have a single IPv4 interface, and are configured with a single TURN and single STUN server each (indeed, the same one for each). As a consequence, each agent will end up with three candidates - a local candidate, a TURN-derived candidate, and a STUN-derived candidate. The agents are seeking to communicate using a single RTP-based voice stream. As a consequence, each candidate has two components - one for RTP and one for RTCP. Agent L is behind a symmetric NAT, and agent R is on the public Internet.

To facilitate understanding, transport addresses are listed in a mnemonic form. This form is <entity&rt;-<type&rt;-<seq-no&rt;, where <entity&rt; refers to the entity whose interface the transport address is on, and is one of "L", "R", "STUN", "TURN", or "NAT". The <type&rt; is either "PUB" for transport addresses that are public, and "PRIV" for transport addresses that are private. Finally, <seq-no&rt; is a sequence number that is different for each transport address of the same type on a particular entity.

The STUN server has advertised transport address STUN-PUB-1 for STUN requests, and the TURN server has advertised transport address TURN-

PUB-1 for TURN allocations.

In addition, candidate IDs are also listed in mnemonic form. Agent L uses candidate ID L1 for its local candidate, L2 for its STUN derived candidate, and L3 for its TURN derived candidate. Agent R uses R1 for its local candidate and R2 for its TURN derived candidate. The passwords for each transport address are LPASS1 through LPASS6 for agent L, and RPASS1 through RPASS4 for agent R.

In example SDP messages, $<token&rt;.IP is used to refer to the value of the IP address of the transport address with mnemonic name "token". Similarly, $<token&rt;.PORT is used to refer to the value of the port of the transport address with mnemonic name "token".

In the call flow example in Figure 6, STUN and TURN 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, or STUN Binding Response messages carried in a TURN SEND or TURN DATA message, and refers to the value of the MAPPED-ADDRESS attribute in the STUN Binding Response. The "RA=" attribute is used in TURN DATA messages, and refers to the value of the REMOTE-ADDRESS attribute. The "U=" attribute is used in STUN Binding Requests, and corresponds to the STUN USERNAME. Finally, the "DA=" attribute is used in TURN SEND messages, and refers to the value of the DESTINATION-ADDRESS attribute.

The call flow example omits STUN authentication operations.

```
             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  |              |              |
             |<-------------|              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |RTCP STUN alloc.             |              |
             |Ta secs. later|              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |(5) STUN Req  |              |              |
             |S=L-PRIV-2    |              |              |
             |D=STUN-PUB-1  |              |              |
             |------------->|              |              |
             |              |              |              |
             |              |              |              |
             |              |(6) STUN Req  |              |
             |              |S=NAT-PUB-2   |              |
             |              |D=STUN-PUB-1  |              |
             |              |------------->|              |
             |              |              |              |
             |              |(7) STUN Res  |              |
             |              |S=STUN-PUB-1  |              |
             |              |D=NAT-PUB-2   |              |
             |              |MA=NAT-PUB-2  |              |
             |              |<-------------|              |
             |              |              |              |
             |(8) STUN Res  |              |              |
             |S=STUN-PUB-1  |              |              |
             |D=L-PRIV-2    |              |              |
             |MA=NAT-PUB-2  |              |              |
             |<-------------|              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |RTP TURN alloc.              |              |
             |Ta secs. later|              |              |
```

```
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |(9) TURN Req  |              |              |
             |S=L-PRIV-1    |              |              |
             |D=TURN-PUB-1  |              |              |
             |------------->|              |              |
             |              |              |              |
             |              |              |              |
             |              |(10) TURN Req |              |
             |              |S=NAT-PUB-3   |              |
             |              |D=TURN-PUB-1  |              |
             |              |------------->|              |
             |              |              |              |
             |              |(11) TURN Res |              |
             |              |S=TURN-PUB-1  |              |
             |              |D=NAT-PUB-3   |              |
             |              |MA=TURN-PUB-2 |              |
             |              |<-------------|              |
             |              |              |              |
             |(12) TURN Res |              |              |
             |S=TURN-PUB-1  |              |              |
             |D=L-PRIV-1    |              |              |
             |MA=TURN-PUB-2 |              |              |
             |<-------------|              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |RTCP TURN alloc.             |              |
             |Ta secs. later|              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |(13) TURN Req |              |              |
             |S=L-PRIV-2    |              |              |
             |D=TURN-PUB-1  |              |              |
             |------------->|              |              |
             |              |              |              |
             |              |              |              |
             |              |(14) TURN Req |              |
             |              |S=NAT-PUB-4   |              |
             |              |D=TURN-PUB-1  |              |
             |              |------------->|              |
             |              |              |              |
             |              |(15) TURN Res |              |
             |              |S=TURN-PUB-1  |              |
             |              |D=NAT-PUB-4   |              |
             |              |MA=TURN-PUB-3 |              |
```

```
             |              |<-------------|              |
             |              |              |              |
             |(16) TURN Res |              |              |
             |S=TURN-PUB-1  |              |              |
             |D=L-PRIV-2    |              |              |
             |MA=TURN-PUB-3 |              |              |
             |<-------------|              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |(17) Offer    |              |              |
             |------------------------------------------->|
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |RTP STUN alloc.
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |              |              |(18) STUN Req |
             |              |              |S=R-PUB-1     |
             |              |              |D=STUN-PUB-1  |
             |              |              |<-------------|
             |              |              |              |
             |              |              |(19) STUN Res |
             |              |              |S=STUN-PUB-1  |
             |              |              |D=R-PUB-1     |
             |              |              |MA=R-PUB-1    |
             |              |              |------------->|
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |RTCP STUN alloc.
             |              |              |              |Ta secs. later
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |              |              |(20) STUN Req |
             |              |              |S=R-PUB-2     |
             |              |              |D=STUN-PUB-1  |
             |              |              |<-------------|
             |              |              |              |
             |              |              |(21) STUN Res |
             |              |              |S=STUN-PUB-1  |
             |              |              |D=R-PUB-2     |
             |              |              |MA=R-PUB-2    |
```

```
             |              |              |------------->|
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |RTP TURN alloc.
             |              |              |              |Ta secs. later
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |              |              |(22) TURN Req |
             |              |              |S=R-PUB-1     |
             |              |              |D=TURN-PUB-1  |
             |              |              |<-------------|
             |              |              |              |
             |              |              |(23) TURN Res |
             |              |              |S=TURN-PUB-1  |
             |              |              |D=R-PUB-1     |
             |              |              |MA=TURN-PUB-4 |
             |              |              |------------->|
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |RTCP TURN alloc.
             |              |              |              |Ta secs. later
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |              |              |(24) TURN Req |
             |              |              |S=R-PUB-2     |
             |              |              |D=TURN-PUB-1  |
             |              |              |<-------------|
             |              |              |              |
             |              |              |(25) TURN Res |
             |              |              |S=TURN-PUB-1  |
             |              |              |D=R-PUB-2     |
             |              |              |MA=TURN-PUB-5 |
             |              |              |------------->|
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |(26) answer   |              |              |
             |<-------------------------------------------|
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |Validate
             |              |              |              |TURN-PUB-4 to TURN-
```

PUB-2

```
             |              |              |              |
             |              |              |              |
             |              |              |(27) TURN SEND|
             |              |              |S=R-PUB-1     |
             |              |              |D=TURN-PUB-1  |
             |              |              |DA=TURN-PUB-2 |
             |              |              |<-------------|
             |              |              |              |
             |              |              |STUN Req.     |
             |              |              |S=TURN-PUB-4  |
             |              |              |D=TURN-PUB-2  |
             |              |              |U=L3:1:R2:1   |
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |              |              |Discard       |
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |Validate      |              |              |
             |TURN-PUB-2 to TURN-PUB-4     |              |
             |              |              |              |
             |              |              |              |
             |(28) TURN SEND|              |              |
             |S=L-PRIV-1    |              |              |
             |D=TURN-PUB-1  |              |              |
             |DA=TURN-PUB-4 |              |              |
             |------------->|              |              |
             |              |              |              |
             |              |(29) TURN SEND|              |
             |              |S=NAT-PUB-3   |              |
             |              |D=TURN-PUB-1  |              |
             |              |DA=TURN-PUB-4 |              |
             |              |------------->|              |
             |              |              |              |
             |              |              |STUN Req.     |
             |              |              |S=TURN-PUB-2  |
             |              |              |D=TURN-PUB-4  |
             |              |              |U=R2:1:L3:1   |
             |              |              |              |
             |              |              |              |
             |              |              |(30) TURN DATA|
             |              |              |S=TURN-PUB-1  |
             |              |              |D=R-PUB-1     |
             |              |              |RA=TURN-PUB-2 |
```

```
             |              |              |------------->|
             |              |              |(31) TURN SEND|
             |              |              |S=R-PUB-1     |
             |              |              |D=TURN-PUB-1  |
             |              |              |DA=TURN-PUB-2 |
             |              |              |MA=TURN-PUB-2 |
             |              |              |<-------------|
             |              |              |              |
             |              |              |STUN Res.     |
             |              |              |S=TURN-PUB-4  |
             |              |              |D=TURN-PUB-2  |
             |              |              |MA=TURN-PUB-2 |
             |              |              |              |
             |              |(32) TURN DATA|              |
             |              |S=TURN-PUB-1  |              |
             |              |D=NAT-PUB-3   |              |
             |              |RA=TURN-PUB-4 |              |
             |              |MA=TURN-PUB-2 |              |
             |              |<-------------|              |
             |(33) TURN DATA|              |              |
             |S=TURN-PUB-1  |              |              |
             |D=L-PRIV-1    |              |              |
             |RA=TURN-PUB-4 |              |              |
             |MA=TURN-PUB-2 |              |              |
             |<-------------|              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |Validate
             |              |              |              |TURN-PUB-4 to TURN-
PUB-2
             |              |              |              |
             |              |              |              |
             |              |              |(34) TURN SEND|
             |              |              |S=R-PUB-1     |
             |              |              |D=TURN-PUB-1  |
             |              |              |DA=TURN-PUB-2 |
             |              |              |<-------------|
             |              |              |              |
             |              |              |STUN Req.     |
             |              |              |S=TURN-PUB-4  |
             |              |              |D=TURN-PUB-2  |
             |              |              |U=L3:1:R2:1   |
             |              |              |              |
             |              |              |              |
             |              |(35) TURN DATA|              |
             |              |S=TURN-PUB-1  |              |
             |              |D=NAT-PUB-3   |              |
           |               |RA=TURN-PUB-4 |               |
           |              |<-------------|              |
           |              |              |              |
           |(36) TURN DATA|              |              |
           |S=TURN-PUB-1  |              |              |
           |D=L-PRIV-1    |              |              |
           |RA=TURN-PUB-4 |              |              |
           |<-------------|              |              |
           |(37) TURN SEND|              |              |
           |S=L-PRIV-1    |              |              |
           |D=TURN-PUB-1  |              |              |
           |DA=TURN-PUB-4 |              |              |
           |MA=TURN-PUB-4 |              |              |
           |------------->|              |              |
           |              |(38) TURN SEND|              |
           |              |S=NAT-PUB-3   |              |
           |              |D=TURN-PUB-1  |              |
           |              |DA=TURN-PUB-4 |              |
           |              |MA=TURN-PUB-4 |              |
           |              |------------->|              |
           |              |              |              |
           |              |              |STUN Res.     |
           |              |              |S=TURN-PUB-2  |
           |              |              |D=TURN-PUB-4  |
           |              |              |MA=TURN-PUB-4 |
           |              |              |              |
           |              |              |(39) TURN DATA|
           |              |              |S=TURN-PUB-1  |
           |              |              |D=R-PUB-1     |
           |              |              |RA=TURN-PUB-2 |
           |              |              |MA=TURN-PUB-4 |
           |              |              |------------->|
           |              |              |              |
           |              |              |              |
           |              |              |              |
           |              |              |              |Validate
           |              |              |              |TURN-PUB-5 to TURN-
PUB-3
           |              |              |              |
           |              |              |              |
           |              |              |(40) TURN SEND|
           |              |              |S=R-PUB-2     |
           |              |              |D=TURN-PUB-1  |
           |              |              |DA=TURN-PUB-3 |
           |              |              |<-------------|
           |              |              |              |
           |              |              |STUN Req.     |
           |              |              |S=TURN-PUB-5  |
           |              |              |D=TURN-PUB-3  |
           |              |              |U=L3:2:R2:2   |
           |              |              |              |
           |              |              |              |
           |              |              |              |
           |              |              |              |
           |              |              |              |
           |              |              |Discard       |
           |              |              |              |
           |              |              |              |
           |              |              |              |
           |              |              |              |
           |Validate      |              |              |
           |TURN-PUB-3 to TURN-PUB-5     |              |
           |              |              |              |
           |              |              |              |
           |(41) TURN SEND|              |              |
           |S=L-PRIV-2    |              |              |
           |D=TURN-PUB-1  |              |              |
           |DA=TURN-PUB-5 |              |              |
           |------------->|              |              |
           |              |              |              |
           |              |(42) TURN SEND|              |
           |              |S=NAT-PUB-4   |              |
           |              |D=TURN-PUB-1  |              |
           |              |DA=TURN-PUB-5 |              |
           |              |------------->|              |
           |              |              |              |
           |              |              |STUN Req.     |
           |              |              |S=TURN-PUB-3  |
           |              |              |D=TURN-PUB-5  |
           |              |              |U=R2:2:L3:2   |
           |              |              |              |
           |              |              |              |
           |              |              |(43) TURN DATA|
           |              |              |S=TURN-PUB-1  |
           |              |              |D=R-PUB-2     |
           |              |              |RA=TURN-PUB-3 |
           |              |              |------------->|
           |              |              |(44) TURN SEND|
           |              |              |S=R-PUB-2     |
           |              |              |D=TURN-PUB-1  |
           |              |              |DA=TURN-PUB-3 |
           |              |              |MA=TURN-PUB-3 |
           |              |              |<-------------|
           |              |              |              |
           |              |              |STUN Res.     |
           |              |              |S=TURN-PUB-5  |
           |              |              |D=TURN-PUB-3  |
           |              |              |MA=TURN-PUB-3 |
```

```
             |              |              |              |
             |              |(45) TURN DATA|              |
             |              |S=TURN-PUB-1  |              |
             |              |D=NAT-PUB-4   |              |
             |              |RA=TURN-PUB-5 |              |
             |              |MA=TURN-PUB-3 |              |
             |              |<-------------|              |
             |(46) TURN DATA|              |              |
             |S=TURN-PUB-1  |              |              |
             |D=L-PRIV-2    |              |              |
             |RA=TURN-PUB-5 |              |              |
             |MA=TURN-PUB-3 |              |              |
             |<-------------|              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |Validate
             |              |              |              |TURN-PUB-5 to TURN-
PUB-3
             |              |              |              |
             |              |              |              |
             |              |              |(47) TURN SEND|
             |              |              |S=R-PUB-2     |
             |              |              |D=TURN-PUB-1  |
             |              |              |DA=TURN-PUB-3 |
             |              |              |<-------------|
             |              |              |              |
             |              |              |STUN Req.     |
             |              |              |S=TURN-PUB-5  |
             |              |              |D=TURN-PUB-3  |
             |              |              |U=L3:2:R2:2   |
             |              |              |              |
             |              |              |              |
             |              |(48) TURN DATA|              |
             |              |S=TURN-PUB-1  |              |
             |              |D=NAT-PUB-4   |              |
             |              |RA=TURN-PUB-5 |              |
             |              |<-------------|              |
             |              |              |              |
             |(49) TURN DATA|              |              |
             |S=TURN-PUB-1  |              |              |
             |D=L-PRIV-2    |              |              |
             |RA=TURN-PUB-5 |              |              |
             |<-------------|              |              |
             |(50) TURN SEND|              |              |
             |S=L-PRIV-2    |              |              |
             |D=TURN-PUB-1  |              |              |
             |DA=TURN-PUB-5 |              |              |
              |MA=TURN-PUB-5 |              |              |
              |------------->|              |              |
              |              |(51) TURN SEND|              |
              |              |S=NAT-PUB-4   |              |
              |              |D=TURN-PUB-1  |              |
              |              |DA=TURN-PUB-5 |              |
              |              |MA=TURN-PUB-5 |              |
              |              |------------->|              |
              |              |              |              |
              |              |              |STUN Res.     |
              |              |              |S=TURN-PUB-3  |
              |              |              |D=TURN-PUB-5  |
              |              |              |MA=TURN-PUB-5 |
              |              |              |              |
              |              |              |(52) TURN DATA|
              |              |              |S=TURN-PUB-1  |
              |              |              |D=R-PUB-2     |
              |              |              |RA=TURN-PUB-3 |
              |              |              |MA=TURN-PUB-5 |
              |              |              |------------->|
              |              |              |              |
              |              |              |              |
              |              |              |              |
              |              |              |              |
              |RTP flows     |              |              |
              |              |              |              |
              |              |              |              |
              |(53) TURN SEND|              |              |
              |S=L-PRIV-1    |              |              |
              |D=TURN-PUB-1  |              |              |
              |DA=TURN-PUB-4 |              |              |
              |------------->|              |              |
              |              |              |              |
              |              |(54) TURN SEND|              |
              |              |S=NAT-PUB-3   |              |
              |              |D=TURN-PUB-1  |              |
              |              |DA=TURN-PUB-4 |              |
              |              |------------->|              |
              |              |              |              |
              |              |              |              |
              |              |              |RTP           |
              |              |              |S=TURN-PUB-2  |
              |              |              |D=TURN-PUB-4  |
              |              |              |              |
              |              |              |              |
              |              |              |(55) TURN DATA|
              |              |              |S=TURN-PUB-1  |
              |              |              |D=R-PUB-1     |
              |              |              |RA=TURN-PUB-2 |
```

```
             |              |              |------------->|
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |RTP flows
             |              |              |              |
             |              |              |              |
             |              |              |(56) TURN SEND|
             |              |              |S=R-PUB-1     |
             |              |              |D=TURN-PUB-1  |
             |              |              |DA=TURN-PUB-2 |
             |              |              |<-------------|
             |              |              |              |
             |              |              |              |
             |              |              |RTP           |
             |              |              |S=TURN-PUB-4  |
             |              |              |D=TURN-PUB-2  |
             |              |              |              |
             |              |              |              |
             |              |(57) TURN DATA|              |
             |              |S=TURN-PUB-1  |              |
             |              |D=NAT-PUB-3   |              |
             |              |RA=TURN-PUB-4 |              |
             |              |<-------------|              |
             |              |              |              |
             |(58) TURN DATA|              |              |
             |S=TURN-PUB-1  |              |              |
             |D=L-PRIV-1    |              |              |
             |RA=TURN-PUB-4 |              |              |
             |<-------------|              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |Validate      |              |              |
             |L-PRIV-1 to R-PUB-1          |              |
             |              |              |              |
             |              |              |              |
             |(59) STUN Req.|              |              |
             |S=L-PRIV-1    |              |              |
             |D=R-PUB-1     |              |              |
             |U=R1:1:L1:1   |              |              |
             |------------->|              |              |
             |              |              |              |
             |              |(60) STUN Req.|              |
             |              |S=NAT-PUB-5   |              |
             |              |D=R-PUB-1     |              |
             |              |U=R1:1:L1:1   |              |
```

```
             |              |----------------------------->|
             |              |              |              |
             |              |(61) STUN Res.|              |
             |              |S=R-PUB-1     |              |
             |              |D=NAT-PUB-5   |              |
             |              |MA=NAT-PUB-5  |              |
             |              |<-----------------------------|
             |              |              |              |
             |(62) STUN Res.|              |              |
             |S=R-PUB-1     |              |              |
             |D=L-PRIV-1    |              |              |
             |MA-NAT-PUB-5  |              |              |
             |<-------------|              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |Validate
             |              |              |              |R-PUB-1 to L-PRIV-1
             |              |              |              |
             |              |              |              |
             |              |(63) STUN Req.|              |
             |              |S=R-PUB-1     |              |
             |              |D=L-PRIV-1    |              |
             |              |U=L1:1:R1:1   |              |
             |              |<-----------------------------|
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |              |Discard       |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |Validate
             |              |              |              |R-PUB-2 to L-PRIV-2
             |              |              |              |
             |              |              |              |
             |              |(64) STUN Req.|              |
             |              |S=R-PUB-2     |              |
             |              |D=L-PRIV-2    |              |
             |              |U=L1:2:R1:2   |              |
             |              |<-----------------------------|
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |              |Discard       |              |
```

```
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |Validate      |              |              |
             |L-PRIV-2 to R-PUB-2          |              |
             |              |              |              |
             |              |              |              |
             |(65) STUN Req.|              |              |
             |S=L-PRIV-2    |              |              |
             |D=R-PUB-2     |              |              |
             |U=R1:2:L1:2   |              |              |
             |------------->|              |              |
             |              |              |              |
             |              |(66) STUN Req.|              |
             |              |S=NAT-PUB-6   |              |
             |              |D=R-PUB-2     |              |
             |              |U=R1:2:L1:2   |              |
             |              |---------------------------->|
             |              |              |              |
             |              |(67) STUN Res.|              |
             |              |S=R-PUB-2     |              |
             |              |D=NAT-PUB-6   |              |
             |              |MA=NAT-PUB-6  |              |
             |              |<----------------------------|
             |              |              |              |
             |(68) STUN Res.|              |              |
             |S=R-PUB-2     |              |              |
             |D=L-PRIV-2    |              |              |
             |MA=NAT-PUB-6  |              |              |
             |<-------------|              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |Validate
             |              |              |              |R-PUB-1 to NAT-PUB-5
             |              |              |              |
             |              |              |              |
             |              |(69) STUN Req.|              |
             |              |S=R-PUB-1     |              |
             |              |D=NAT-PUB-5   |              |
             |              |U=L1R1:1:R1:1 |              |
             |              |<----------------------------|
             |              |              |              |
             |(70) STUN Req.|              |              |
             |S=R-PUB-1     |              |              |
             |D=L-PRIV-1    |              |              |
             |U=L1R1:1:R1:1 |              |              |
```

```
             |<-------------|              |              |
             |              |              |              |
             |(71) STUN Res.|              |              |
             |S=L-PRIV-1    |              |              |
             |D=R-PUB-1     |              |              |
             |MA=R-PUB-1    |              |              |
             |------------->|              |              |
             |              |              |              |
             |              |(72) STUN Res.|              |
             |              |S=NAT-PUB-5   |              |
             |              |D=R-PUB-1     |              |
             |              |MA=R-PUB-1    |              |
             |              |---------------------------->|
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |Validate
             |              |              |              |R-PUB-2 to NAT-PUB-6
             |              |              |              |
             |              |              |              |
             |              |(73) STUN Req.|              |
             |              |S=R-PUB-2     |              |
             |              |D=NAT-PUB-6   |              |
             |              |U=L1R1:2:R1:2 |              |
             |              |<----------------------------|
             |              |              |              |
             |(74) STUN Req.|              |              |
             |S=R-PUB-2     |              |              |
             |D=L-PRIV-2    |              |              |
             |U=L1R1:2:R1:2 |              |              |
             |<-------------|              |              |
             |              |              |              |
             |(75) STUN Res.|              |              |
             |S=L-PRIV-2    |              |              |
             |D=R-PUB-2     |              |              |
             |MA=R-PUB-2    |              |              |
             |------------->|              |              |
             |              |              |              |
             |              |(76) STUN Res.|              |
             |              |S=NAT-PUB-6   |              |
             |              |D=R-PUB-2     |              |
             |              |MA=R-PUB-2    |              |
             |              |---------------------------->|
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |(77) Offer    |              |              |
```

```
             |-------------------------------------------->|
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |(78) Answer   |              |              |
             |<--------------------------------------------|
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |
             |              |              |              |
```

Figure 6

First, agent L obtains a STUN derived transport address for its RTP packets (messages 1-4). Recall that the NAT is symmetric. Here, it creates a binding of NAT-PUB-1 for this UDP request, and this becomes the STUN derived transport address for RTP. Agent L repeats this process for RTCP (messages 5-8) Ta seconds later, and obtains NAT-PUB-2 as its STUN derived transport address for RTCP. The two transport addresses are the two components of the STUN derived candidate that agent L has just obtained.

Next, agent L will allocate a TURN derived transport address for RTP (messages 9-12) and RTCP (messages 13-16). This produces TURN-PUB-2 and TURN-PUB-3 for RTP and RTCP, respectively.

With its three candidates, agent L prioritizes them, choosing the local candidate as highest priority, followed by the STUN derived candidate, followed by the TURN-derived candidate. It chooses its TURN derived candidate as the active candidate, and encodes it into the m/c-line. The resulting offer (message 17) looks like:

```
v=0
o=jdoe 2890844526 2890842807 IN IP4 $L-PRIV-1.IP
s=
c=IN IP4 $TURN-PUB-2.IP
t=0 0
m=audio $TURN-PUB-2.PORT RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=rtcp:$TURN-PUB-3.PORT
a=candidate $L1 1 $L-PASS1 UDP 1.0 $L-PRIV-1.IP $L-PRIV-1.PORT
a=candidate $L1 2 $L-PASS2 UDP 1.0 $L-PRIV-2.IP $L-PRIV-2.PORT
a=candidate $L2 1 $L-PASS3 UDP 0.7 $NAT-PUB-1.IP $NAT-PUB-1.PORT
a=candidate $L2 2 $L-PASS4 UDP 0.7 $NAT-PUB-2.IP $NAT-PUB-2.PORT
```

```
       a=candidate $L3 1 $L-PASS5 UDP 0.3 $TURN-PUB-2.IP $TURN-PUB-2.PORT
       a=candidate $L3 2 $L-PASS6 UDP 0.3 $TURN-PUB-3.IP $TURN-PUB-3.PORT
```

This offer is received at agent R. Agent R will gather its STUN derived RTP transport address (messages 18-19) and RTCP address (messages 20-21). Since the result of the STUN allocations did not provide a new set of transport addresses, there will not be a separate candidate for them. Agent R then gathers its TURN derived RTP transport address (messages 22-23) and TURN derived RTCP transport addresses (messages 24-25). Agent R now has two candidates. It prioritizes the local candidate with higher priority than the TURN candidate, and selects the TURN candidate as the active candidate. Its resulting answer looks like:

```
       v=0
       o=bob 2808844564 2808844564 IN IP4 $R-PUB-1.IP
       s=
       c=IN IP4 $TURN-PUB-4.IP
       t=0 0
       m=audio $TURN-PUB-4.PORT RTP/AVP 0
       a=rtpmap:0 PCMU/8000
       a=rtcp:$TURN-PUB-5.PORT
       a=candidate $R1 1 $R-PASS1 UDP 1.0 $R-PUB-1.IP $R-PUB-1.PORT
       a=candidate $R1 2 $R-PASS2 UDP 1.0 $R-PUB-2.IP $R-PUB-2.PORT
       a=candidate $R2 1 $R-PASS3 UDP 0.3 $TURN-PUB-4.IP $TURN-PUB-4.PORT
       a=candidate $R2 2 $R-PASS4 UDP 0.3 $TURN-PUB-5.IP $TURN-PUB-5.PORT
```

Next, agents L and R form candidate pairs and the transport address check ordered list. This list will start with the two components in the currently active candidate pair - TURN. Agent R begins its checks (message 27). It will check connectivity between the active candidate pair, starting with the first component, which is TURN-PUB-4 for agent R and TURN-PUB-2 for agent L. The state machine for that transport address pair moves to the Testing state. Since this is a TURN-derived transport address for agent R, it utilizes the TURN SEND mechanism to deliver the Binding Request. The DESTINATION-ADDRESS is TURN-PUB-2.

The TURN server will extract the content of the TURN message, which is a STUN Binding Request, and deliver it to the destination, TURN-PUB-4. This request will be sent from the TURN address allocated to R, which is TURN-PUB-4. As both interfaces are on the TURN server, this message is sent to itself (and thus the lack of a message number in the sequence diagram above). Note that the USERNAME in the Binding Request is L3:1:R2:1, which represents the transport address pair ID. This message gets discarded by the TURN server since, as of yet, there are no permissions established for the TURN-PUB-2

allocation. However, it did have the side effect of establishing a permission on the TURN-PUB-4 binding, allowing incoming packets from TURN-PUB-2.

Once L gets the offer, it will attempt to validate the first transport address pair in the transport address pair check ordered list, which will be the active candidate. The state machine for this transport address pair moves into the Testing state. Like agent R did, it will use the TURN SEND primitive to send a STUN Binding Request from its TURN derived transport address, TURN-PUB-2, to TURN-PUB-4 (message 28). This packet traverses the NAT (message 29) and arrives at the TURN server. The TURN server will unwrap the contents of the packet and send them from TURN-PUB-2 to TURN-PUB-4. It will also, as a consequence, add a permission for TURN-PUB-4. The contents of the packet are a STUN Binding Request with USERNAME R2:1:L3:1 (note how this is the flip of the USERNAME in the Binding Request sent by agent R). This is also a packet from the TURN server to itself. However, now, the packet is not discarded, as a permission had been installed as a consequence of the "suicide packet" from agent R (a suicide packet is a packet that has no hope of traversing a far end NAT, but serves the purpose of enabling a permission in a near end NAT so that a packet from the peer can be returned). Thus, the TURN server will relay the received STUN request towards agent R (message 30). This is delivered as a TURN DATA Indication. Notice how the REMOTE-ADDRESS is TURN-PUB-2; this is important as it will be used to construct the STUN Binding Response.

Agent R will receive the DATA Indication, and unwrap its contents to find the Binding Request. The state machine for this transport address pair is currently in the Testing state. It therefore moves into the Send-Valid state, and it generates a Binding Response. However, the MAPPED-ADDRESS in the Binding Response is constructed using the source IP address and port that were seen by the TURN server when the Binding Request arrived at TURN-PUB-4, which is the looped message between messages 29 and 30. This source address is TURN-PUB-2, which is the value of the REMOTE-ADDRESS attribute in message 30. Thus, the STUN Binding Response will contain TURN-PUB-2 in the MAPPED-ADDRESS, and is to be sent to TURN-PUB-2. To send the response, agent R takes the STUN Binding Response and encapsulates it in a TURN SEND primitive, setting the DESTINATION-ADDRESS to TURN-PUB-2. This is shown in message 31.

The TURN server will receive this SEND request, and unwrap its contents to find the STUN Binding Response. It sends it to the value of the DESTINATION-ADDRESS attribute, and sends it from the TURN address allocated to R, which is TURN-PUB-4. This, once again, results in a looped message to itself, and it arrives at TURN-PUB-2.

Now, however, there is a permission installed for TURN-PUB-4. The TURN server will therefore forward the packet to agent L. To do so, it constructs a TURN DATA Indication containing the contents of the packet. It sets the REMOTE-ADDRESS to the source transport address of the request it received (TURN-PUB-4), and forwards it to agent L (message 32). This traverses the NAT (message 33) and arrives at agent L. As a consequence of the receipt of a Binding Response, the state machine for this transport address pair moves to the Recv-Valid state. The agent also examines the MAPPED-ADDRESS of the STUN response. It is TURN-PUB-2. This is the same as the native transport address of this transport address pair, and thus doesn't represent a new transport address that might have been learned.

A short while later, agent R will attempt a retransmission of its STUN Binding Request that was lost (the contents of message 27 that were discarded by the TURN server due to lack of permission). This time, however, a permission has been installed and the retransmission will work. So, it sends the Binding Request again (message 34, identical to message 27). This is looped by the TURN server to itself again, but this time there is a permission in place when it arrives at TURN-PUB-2. As such, the request is forwarded towards agent L this time, in a TURN DATA Indication (message 35). This traverses the NAT (message 36) and arrives at agent L. Agent L extracts the contents of the request, which are a STUN Binding Request. This causes the state machine to move from Recv-Valid to Valid. It generates a STUN Binding Response, and sets the MAPPED-ADDRESS to the value of the REMOTE-ADDRESS in message 36 (TURN-PUB-4). This Binding Response is sent to TURN-PUB-4, which is accomplished through a TURN SEND primitive (message 37). This SEND Request traverses the NAT (message 38) and is received by the TURN server. Its contents are decapsulated, and sent to TURN-PUB-4, which is again a loop on the same host. This packet is then sent towards agent R in a DATA Indication (message 39). The contents of the DATA Indication are extracted, and the agent sees a successful Binding Response. It therefore moves the state machine from the Send-Valid state to the Valid state. At this point, the transport address pair is in the Valid state for both agents.

Approximately Ta seconds after agent R sent message 27, agent R will start checks for the next transport address pair in its transport address pair check ordered list. This is the second component of the same candidate pair, used for RTCP. This sequence, messages 40 through 52, are identical to the ones for RTP, but differ only in the specific transport addresses.

Once that validation happens, the second transport address pair has been validated. The candidate pair moves into the valid state, and both candidates are considered valid. The active candidate has now

been validated, and media can begin to flow. It will do so through the TURN server; indeed, it is relayed "twice" through the TURN server. Even though there is a single TURN server, it is logically acting as two separate TURN servers. Indeed, had L and R used two separate TURN servers, media would be relayed through both TURN servers.

The actual media flows are shown as well. It is important to note that, since the ICE checks have not yet concluded on the candidate that will ultimately be used, no TURN Set Active Destinations have been sent. As a consequence, media that is sent through the TURN servers has to be sent using TURN Send requests. This introduces some overhead, but is a transient condition. In message 53, agent L sends an RTP packet to agent R using a SEND request. It is sent to TURN-PUB-4. This traverses the NAT (message 54), and arrives at the TURN server. It is decapsulated, looped to itself, and arrives at TURN-PUB-4. From there, it is encapsulated in a DATA Indication and sent to agent R (message 55). In the reverse direction, agent R will send an RTP packet using a TURN SEND primitive (message 56), and send it to TURN-PUB-2. This is received by the TURN server, decapsulated, and sent to TURN-PUB-2 from TURN-PUB-4. This is again a loop within the same host, arriving at TURN-PUB-4. The contents of the packet are sent to agent L through a TURN DATA Indication (message 57), which traverses the NAT (message 58) to arrive at agent R. Since this call flow is already long enough, RTCP packet transmission is not shown.

Approximately Ta seconds after it sends message 41, agent L goes to the next transport address pair in its transport address pair check ordered list that is in the Waiting state. This will be the RTP candidate for the top priority candidate pair, which is L-PRIV-1 on agent L and R-PUB-1 on agent R. This is a local candidate for each agent. To perform the check, agent R sends a STUN Binding Request from L-PRIV-1 to R-PUB-1 (message 59). Note the USERNAME of R1:1:L1:1, which identifies this transport address pair. This traverses the NAT (message 60). Since the NAT is symmetric and this is a new destination IP address, the NAT allocates a new transport address on its public side, NAT-PUB-5, and places this in the source IP address and port. This packet arrives at agent R. Agent R finds a matching transport address pair in the Waiting state. The state machine transitions to the Send-Valid state. It sends the Binding response, with a MAPPED-ADDRESS equal to NAT-PUB-5 (message 61), which traverses the NAT and arrives at agent L (message 62). Agent R, in addition to sending the response, will also send a Binding Request. It is important to remember that this Binding Request is sent to the remote address in the transport address pair (L-PRIV-1), and NOT to the source IP address and port of the Binding Request (NAT-PUB-5); that will happen later. This attempt is shown in

message 63. However, since the L-PRIV-1 is private, the packet is discarded in the network.

Now, as a consequence of receiving message 60, agent R will have constructed a peer-derived candidate. The candidate ID for this candidate is L1R1, and it initially contains a single transport address pair, NAT-PUB-5 and R-PUB-1. However, the candidate isn't yet usable until the other component gets added. Similarly, agent L will have constructed the same peer-derived candidate, with the same candidate ID and the same transport address pair.

Some Ta seconds after sending message 40, agent R will move to the next transport address pair in the transport address pair check ordered list whose state is Waiting. This is the RTCP component of the highest priority candidate pair. It will attempt a connectivity check, from R-PUB-2 to L-PRIV-2 (message 64). Since L-PRIV-1 is private, this message is discarded.

Some Ta seconds after sending message 59, agent L will move to the next transport address pair in the transport address pair check ordered list whose state is Waiting. This is the RTCP component of the highest priority candidate pair. It will attempt a connectivity check, from L-PRIV-2 to R-PUB-2 (message 65), which operates nearly identically to messages 59-62, with the exception of the specific addresses. Here, the NAT will create a new binding for the RTCP, NAT-PUB-6, and this transport address is new for both participants. On receipt of this Binding Request at agent R (message 66), agent R constructs the candidate ID for the peer-derived candidate, L1R1, and finds it already exists. As such, this new transport address is added, and the peer-derived candidate becomes complete and usable. Agent L does the same thing on receipt of message 68. This candidate will have the same priority as its generating candidate L1 (1.0), and is paired up with R1 (also at priority 1.0). Since L1R1 has the same priority as L1 itself, the ordering algorithm in Section 7.5 will use the reverse lexicographic order of the candidate ID iself to determine order. L1R1 is larger than L1, so that the peer-derived candidate will come before its generating candidate. As a consequence, the peer-derived candidate pair will have a higher priority than its generating candidate, and appear just before it in the candidate pair priority ordered list.

As a consequence, after agent R sends message 67 and completes the peer-derived candidate, it will move the two transport addresses in the peer derived candidate into the Send-Valid state, and send a Binding Request for each in rapid succession (agent L will have moved both into the Recv-Valid state upon receipt of message 68). The first of these connectivity checks are for the RTP component, from R-PUB-1 to NAT-PUB-5 (message 69). Note the USERNAME in the STUN

Binding Request, L1R1:1:R1:1, which identifies the peer-derived transport address pair. This will succesfully traverse the NAT and be delivered to agent L (message 70). The receipt of this request moves the state machine for this transport address pair from Recv-Valid to Valid, and a Binding Response is sent (message 71). This passes through the NAT and arrives at agent R (message 72). This causes its state machine to enter the Valid state as well. The MAPPED-ADDRESS, R-PUB-1, is not new to agent R and thus does not result in the creation of a new peer-derived candidate.

Messages 73 through 76 show the same basic flow for RTCP. Upon receipt of message 76, both transport address pairs are Valid at both agents, causing the peer derived candidate to become valid. Timer Tws is set at agent L, and fires without any higher priority candidate pairs becoming validated. As such, it now decides to send an updated offer to promote the peer-derived candidate to active. This offer (message 77) looks like:

```
v=0
o=jdoe 2890844526 2890842808 IN IP4 $L-PRIV-1.IP
s=
c=IN IP4 $NAT-PUB-5.IP
t=0 0
m=audio $NAT-PUB-5.PORT RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=rtcp:$NAT-PUB-6.PORT
a=remote-candidate:R1
a=candidate $L1 1 $L-PASS1 UDP 1.0 $L-PRIV-1.IP $L-PRIV-1.PORT
a=candidate $L1 2 $L-PASS2 UDP 1.0 $L-PRIV-2.IP $L-PRIV-2.PORT
```

There are several important things to note in this offer. Firstly, note how the m/c-line now contains NAT-PUB-5 and NAT-PUB-6, the peer derived transport addresses it learned through the ICE processing. Secondly, note how there remains a candidate encoded into the a=candidate attributes. This is candidate L1, NOT candidate L1R1. Recall that the peer-derived candidates are never encoded into the SDP. Rather, their generating candidate is encoded. This will cause keepalives to take place for the genreating candidate if valid (though its not) and any of its derived candidates, which is what we want. Finally, notice the inclusion of the a=remote-candidate attribute. Since agent L doesn't know whether agent R received messages 72 or 76, it doesnt know whether the state of the candidate is Recv-Valid or Valid at agent R. So, it has to tell agent R that, in case its Recv-Valid, to please use it anyway.

The answer generated by agent R looks like:

```
v=0
o=bob 2808844564 2808844565 IN IP4 $R-PUB-1.IP
s=
c=IN IP4 $R-PUB-1.IP
t=0 0
m=audio $R-PUB-1.PORT RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=rtcp:$R-PUB-2.PORT
a=candidate $R1 1 $R-PASS1 UDP 1.0 $R-PUB-1.IP $R-PUB-1.PORT
a=candidate $R1 2 $R-PASS2 UDP 1.0 $R-PUB-2.IP $R-PUB-2.PORT
```

With this, media can now flow directly between endpoints. The removal of the TURN-based candidates from the offer/answer exchange will cause the TURN allocations to be removed.

12. Grammar

This specification defines two new SDP attributes - the "candidate" and "remote-candidate" attributes.

The candidate attribute MUST be present within a media block of the SDP. It contains a transport address for a candidate that can be used for connectivity checks. There may be multiple candidate attributes in a media block.

The syntax of this attribute is defined using Augmented BNF as defined in RFC 2234 [11]:

```
candidate-attribute   = "candidate" ":" candidate-id SP component-id SP
                        password SP
                        transport SP
                        qvalue SP   ;qvalue from RFC 3261
                        addr SP     ;addr from RFC 3266
                        port        ;port from RFC 2327
                        *(SP extension-att-name SP
                             extension-att-value)

transport             = "UDP" / transport-extension
transport-extension   = token
candidate-id          = 1*base64-char
password              = 1*base64-char
base64-char           = ALPHANUM / DIGIT / "+" / "/"
component-id          = 1*DIGIT
extension-att-name    = token
extension-att-value   = token
```

The candidate-id is used to group together the transport addresses for a particular candidate. It MUST be constructed with at least 128 bits of randomness. It MUST have the same value for all transport addresses within the same candidate. It MUST have a different value for transport addresses within different candidates for the same media stream. The password MUST also be constructed with at least 128 bits of randomness, and MUST differ for each transport address. Both of these use a syntax that is defined to be equal to the base64 alphabet [4], which allows the candidate-id and password to be generated by performing a base64 encoding of a randomly generated 128 bit value (note, however, that this does not mean that the candidate-id or password is base64 decoded when use in STUN messages). The component-id is a positive integer, which identifies the specific component of the candidate. It MUST start at 1 and MUST increment by 1 for each component of a particular candidate.

The addr production is taken from [12], allowing for IPv4 addresses, IPv6 addresses and FQDNs. The port production is taken from RFC 2327 [7]. The token production is taken from RFC 3261 [3]. The transport production 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) [32].

The a=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.

The syntax of the "remote-candidate" attribute is defined using Augmented BNF as defined in RFC 2234 [11]:

```
remote-candidate-att = "remote-candidate" ":" candidate-id
```

This attribute MUST be present in an offer when the candidate in the m/c-line is part of a candidate pair that is in the valid or partially valid state.

13. Security Considerations

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

13.1 Attacks on Connectivity Checks

An attacker might also attempt to disrupt the STUN-based 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-Derived Candidate An attacker can cause an agent to discover a new peer-derived 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-derived candidate or false STUN-derived candidate). It must then launch an attack that forces the agents to believe that this candidate is valid.

Of the various techniques for creating faked STUN messages described in RFC 3489, many are not applicable for the connectivity checks. Compromises of STUN servers are not much of a concern, since the STUN servers are embedded in endpoints and distributed throughout the network. Thus, compromising the STUN server is equivalent to comprimising the endpoint, and if that happens, far more problematic attacks are possible than those against ICE. Similarly, DNS attacks are irrelevant since STUN servers are not discovered via DNS, they are signaled via SIP. Injection of fake responses and relaying modified requests all can be handled in ICE with the countermeasures discussed below.

To force the false invalid result, the attacker has to wait for the connectivity check for 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 600. This attack only needs to be launched against one of the agents in order to invalidate the candidate pair. 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. This will cause the agent to abandon the candidate, which is the desired result in any case. Fortunately, this attack is mitigated completely through the STUN message integrity 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.

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 completely through the STUN message integrity and offer/answer security techniques.

Forcing the false peer-derived 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 transport address. In addition, the attacker must wait for a Binding Request from the other agent, and generate a fake response with a MAPPED-ADDRESS attribute. This attack is best launched against a candidate pair that is likely to be invalid, so the attacker doesnt need to contend with the actual responses to the real connectivity checks. 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-derived candidate result with packet replays is different. The attacker waits until one of the agents sends a Binding Request for one of the transport address pairs. It then 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 MAPPED-ADDRESS with the false transport address. It is passed to the this false address. The attacker must then intercept it and relay it towards the originator.

The other agent will then initiate a connectivity check towards that transport address. 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 Binding Request towards this false transport address, 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 themself is identified by the fake transport address. 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 transport address, the attack is easier to coordinate. However, if SRTP is used [25], 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.

13.2 Attacks on Address Gathering

ICE endpoints make use of STUN for gathering addresses from a STUN server in the network. This is corresponds to the binding acquisition use case discussed in Section 10.1 of RFC 3489. As a consequence, the attacks against STUN itself that are described in Section 12 of RFC 3489 can still be used against the STUN address gathering operations that occur in ICE.

However, the additional mechanisms provided by ICE actually counteract such attacks, making binding acquisition with STUN more secure when combined with ICE than without ICE.

Consider an attacker which is able to provide an agent with a faked MAPPED-ADDRESS in a STUN Binding Request that is used for address gathering. This is the primary attack primitive described in Section 12 of RFC 3489. This address will be used as a STUN derived 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 third party, 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 STUN-derived address from being used. However, if the peer agent has at least one address that is reachable by the agent under attack, the STUN connectivity checks themselves will provide a STUN-derived address that can be used for the exchange of media. Peer derived candidates are preferred over the candidate they are generated from for this reason. As such, an attack solely on the STUN address gathering will normally have no impact on a call at all.

13.3 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 [5] apply. These require techniques for message integrity and encryption for offers and answers, which are satisfied by the SIPS mechanism when SIP is used. As such, the usage of SIPS with ICE is RECOMMENDED.

13.4 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.

13.4.1 The Voice Hammer Attack

The voice hammer attack is an amplification attack, of the variety discussed in Section 3 of [30]. In this attack, the attacker initiates sessions to other agents, and includes the IP address and port of a DoS target in the m/c-line of their 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 speific to ICE, but ICE can help provide remediation.

Specifically, if ICE is used, the agent receiving the malicious SDP will first peform connectivity checks to the target of media before sending it 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.

13.4.2 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. This attack is accomplished by having the offerer send 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 50ms, and each check is a STUN transaction consisting of 9 retransmits of a message 64 bytes in length. This produces a fairly substantial 92 kbps, just in STUN requests.

It is impossible to eliminate the amplification, but the volume can be reduced through a variety of heuristics. For example, agents can limit the number of candidates they'll accept in an offer or answer, they can increase the value of Ta, or exponentially increase Ta as time goes on. All of these ultimately trade off the time for the ICE exchanges to complete, with the amount of traffic that gets sent.

14. IANA Considerations

This specification defines two new SDP attribute per the procedures of Appendix B of RFC 2327. The required information for the registrations are included here.

14.1 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 of UDP with NAT (STUN).

Appropriate Values: See Section 12 of RFC XXXX [Note to RFC-ed: please replace XXXX with the RFC number of this specification].

14.2 remote-candidate Attribute

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

Attribute Name: remote-candidate

Long Form: remote-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 the identity of the remote candidate that the offerer wishes the answerer to use in its answer.

Appropriate Values: See Section 12 of RFC XXXX [Note to RFC-ed: please replace XXXX with the RFC number of this specification].

15. 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 [23]. 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. The IAB has mandated that any protocols developed for this purpose document a specific set of considerations. This section meets those requirements.

15.1 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 resolving many of the limitations of other UNSAF mechanisms by wrapping them in an additional layer of processing (the ICE methodology).

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

15.2 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. 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, derived transport addresses from other UNSAF mechanisms simply never get used, because higher priority connectivity exists. 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.

15.3 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, traditional STUN (the usage described in RFC 3489) 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. The only potential problem is that bilaterally fixed addresses through STUN can expire if traffic does not keep them alive. However, that is substantially less brittleness than the STUN discovery mechanisms.

Another point of brittleness in STUN, TURN, 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 or TURN server would still allow for a call to progress when ICE is used.

Another point of brittleness in traditional 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 traditional 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 can be lifted.

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

15.4 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 STUN remain unchanged. However, we feel ICE actually helps because we believe it can be part of the long term solution.

15.5 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 will interfere with proper operation of any UNSAF mechanism, including ICE.

16. Acknowledgements

The authors would like to thank Flemming Andreasen, Dan Wing, Douglas Otis, Francois Audet and Magnus Westerlund for their comments and input.

17. References

17.1 Normative References

[1] 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.

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

[3] 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.

[4] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 3548, July 2003.

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

[6] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[7] Handley, M. and V. Jacobson, "SDP: Session Description Protocol", RFC 2327, April 1998.

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

July 2003.

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

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

[11] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[12] Olson, S., Camarillo, G., and A. Roach, "Support for IPv6 in Session Description Protocol (SDP)", RFC 3266, June 2002.

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

[14] Andreasen, F., "Connectivity Preconditions for Session Description Protocol Media Streams", draft-ietf-mmusic-connectivity-precon-00 (work in progress), May 2005.

[15] Yon, D., "Connection-Oriented Media Transport in the Session Description Protocol (SDP)", draft-ietf-mmusic-sdp-comedia-10 (work in progress), November 2004.

[16] Rosenberg, J., "Traversal Using Relay NAT (TURN)", draft-rosenberg-midcom-turn-08 (work in progress), September 2005.

17.2 Informative References

[17] Schulzrinne, H., Rao, A., and R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998.

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

[19] Rosenberg, J. and H. Schulzrinne, "An RTP Payload Format for Generic Forward Error Correction", RFC 2733, December 1999.

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

[21] Borella, M., Lo, J., Grabelsky, D., and G. Montenegro, "Realm

Specific IP: Framework", RFC 3102, October 2001.

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

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

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

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

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

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

[28] Rosenberg, J., "The Session Initiation Protocol (SIP) UPDATE Method", RFC 3311, October 2002.

[29] Andreasen, F., "A No-Op Payload Format for RTP", draft-ietf-avt-rtp-no-op-00 (work in progress), May 2005.

[30] Rescorla, E. and M. Handley, "Internet Denial of Service Considerations", draft-iab-dos-03 (work in progress), September 2005.

[31] Huitema, C., "Teredo: Tunneling IPv6 over UDP through NATs", draft-huitema-v6ops-teredo-05 (work in progress), April 2005.

[32] Kohler, E., "Datagram Congestion Control Protocol (DCCP)", draft-ietf-dccp-spec-11 (work in progress), March 2005.

[33] Lazzaro, J., "Framing RTP and RTCP Packets over Connection-Oriented Transport", draft-ietf-avt-rtp-framing-contrans-06 (work in progress), September 2005.

[34] Hellstrom, G., "RTP Payload for Text Conversation", draft-ietf-avt-rfc2793bis-09 (work in progress), August 2004.

Author's Address

Jonathan Rosenberg
Cisco Systems
600 Lanidex Plaza
Parsippany, NJ 07054
US

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

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.