MMUSIC
Internet-Draft
Expires: July 20, 2007

J. Rosenberg
Cisco Systems

January 16, 2007

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

draft-ietf-mmusic-ice-13

Status of this Memo

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 the Session Traversal Utilities for NAT (STUN) protocol, applying its binding discovery and relay usages, in addition to defining a new usage for checking connectivity between peers.

Table of Contents

1. Introduction 5
2. Overview of ICE 5
2.1. Gathering Candidate Addresses 7
2.2. Connectivity Checks 9
2.3. Sorting Candidates 10
2.4. Frozen Candidates 11
2.5. Security for Checks 11
2.6. Concluding ICE 12
2.7. Lite Implementations 13
3. Terminology 13
4. Sending the Initial Offer 16
4.1. Full Implementation Requirements 16
4.1.1. Gathering Candidates 16
4.1.2. Prioritizing Candidates 18
4.1.3. Choosing In-Use Candidates 20
4.2. Lite Implementation 20
4.3. Encoding the SDP 21
5. Receiving the Initial Offer 22
5.1. Verifying ICE Support 23
5.2. Determining Role 23
5.3. Gathering Candidates 24
5.4. Prioritizing Candidates 24
5.5. Choosing In Use Candidates 24
5.6. Encoding the SDP 24
5.7. Forming the Check Lists 24
5.8. Performing Periodic Checks 27
6. Receipt of the Initial Answer 28
6.1. Verifying ICE Support 28
6.2. Determining Role 28
6.3. Forming the Check List 28
6.4. Performing Periodic Checks 28
7. Connectivity Checks 28
7.1. Client Procedures 29
7.1.1. Sending the Request 29
7.1.2. Processing the Response 30
7.2. Server Procedures 31
7.2.1. Additional Procedures for Full Implementations 32
7.2.2. Additional Procedures for Lite Implementations 34
8. Concluding ICE 34
9. Subsequent Offer/Answer Exchanges 35
9.1. Generating the Offer 35
9.1.1. Additional Procedures for Full Implementations 36
9.1.2. Additional Procedures for Lite Implementations 37
9.2. Receiving the Offer and Generating an Answer 37
9.2.1. Additional Procedures for Full Implementations 38
9.3. Updating the Check and Valid Lists 38

9.3.1. Additional Procedures for Full Implementations 38
10. Keepalives 40
11. Media Handling 41
11.1. Sending Media 41
11.1.1. Procedures for Full Implementations 41
11.1.2. Procedures for Lite Implementations 42
11.2. Receiving Media 42
12. Usage with SIP 42
12.1. Latency Guidelines 42
12.2. SIP Option Tags and Media Feature Tags 44
12.3. Interactions with Forking 44
12.4. Interactions with Preconditions 45
12.5. Interactions with Third Party Call Control 45
13. Grammar 45
14. Extensibility Considerations 48
15. Example 49
16. Security Considerations 54
16.1. Attacks on Connectivity Checks 54
16.2. Attacks on Address Gathering 57
16.3. Attacks on the Offer/Answer Exchanges 57
16.4. Insider Attacks 57
16.4.1. The Voice Hammer Attack 58
16.4.2. STUN Amplification Attack 58
16.5. Interactions with Application Layer Gateways and SIP . . . 59
17. Definition of Connectivity Check Usage 59
17.1. Applicability 60
17.2. Client Discovery of Server 60
17.3. Server Determination of Usage 60
17.4. New Requests or Indications 60
17.5. New Attributes 60
17.6. New Error Response Codes 61
17.7. Client Procedures 61
17.8. Server Procedures 61
17.9. Security Considerations for Connectivity Check 61
18. IANA Considerations 61
18.1. SDP Attributes 61
18.1.1. candidate Attribute 61
18.1.2. remote-candidates Attribute 62
18.1.3. ice-lite Attribute 62
18.1.4. ice-mismatch Attribute 63
18.1.5. ice-pwd Attribute 63
18.1.6. ice-ufrag Attribute 63
18.1.7. ice-options Attribute 64
18.2. STUN Attributes 64
19. IAB Considerations 65
19.1. Problem Definition 65
19.2. Exit Strategy 65
19.3. Brittleness Introduced by ICE 66

19.4. Requirements for a Long Term Solution 67
19.5. Issues with Existing NAPT Boxes 67
20. Acknowledgements 68
21. References 68
21.1. Normative References 68
21.2. Informative References 69
Appendix A. Lite and Full Implementations 71
Appendix B. Design Motivations 71
B.1. Pacing of STUN Transactions 72
B.2. Candidates with Multiple Bases 72
B.3. Purpose of the Translation 74
B.4. Importance of the STUN Username 74
B.5. The Candidate Pair Sequence Number Formula 75
B.6. The Frozen State 76
B.7. The remote-candidates attribute 76
B.8. Why are Keepalives Needed? 77
B.9. Why Prefer Peer Reflexive Candidates? 78
B.10. Why Send an Updated Offer? 78
B.11. Why are Binding Indications Used for Keepalives? 78
Author's Address 80
Intellectual Property and Copyright Statements 81

1. Introduction

RFC 3264 [4] defines a two-phase exchange of Session Description Protocol (SDP) messages [10] for the purposes of establishment of multimedia sessions. This offer/answer mechanism is used by protocols such as the Session Initiation Protocol (SIP) [3].

Protocols using offer/answer are difficult to operate through Network Address Translators (NAT). Because their purpose is to establish a flow of media packets, they tend to carry IP addresses within their messages, which is known to be problematic through NAT [15]. 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 [16], Simple Traversal Underneath NAT (STUN) [14] and its revision, retitled Session Traversal Utilities for NAT [11], the STUN Relay Usage [12], and Realm Specific IP [18] [19] along with session description extensions needed to make them work, such as the Session Description Protocol (SDP) [10] 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. It is called Interactive Connectivity Establishment, or ICE. ICE makes use of STUN and its relay extension, commonly called TURN, but uses them in a specific methodology which avoids many of the pitfalls of using any one alone.

2. Overview of ICE

In a typical ICE deployment, we have two endpoints (known as agents in RFC 3264 terminology) which want to communicate. They are able to communicate indirectly via some signaling system such as SIP, by

which they can perform an offer/answer exchange of SDP [4] messages. Note that ICE is not intended for NAT traversal for SIP, which is assumed to be provided via some other mechanism [32]. At the beginning of the ICE process, the agents are ignorant of their own topologies. In particular, they might or might not be behind a NAT (or multiple tiers of NATs). ICE allows the agents to discover enough information about their topologies to find a path or paths by which they can communicate.

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

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

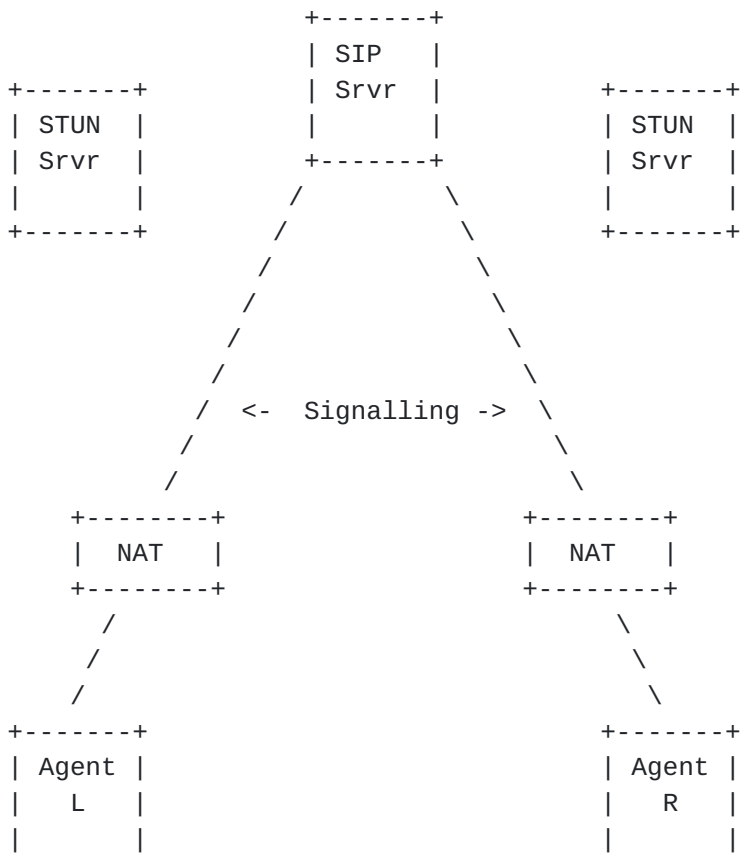


Figure 1

The basic idea behind ICE is as follows: each agent has a variety of candidate transport addresses it could use to communicate with the other agent. These might include:

- It's directly attached network interface (or interfaces in the case of a multihomed machine
- A translated address on the public side of a NAT (a "server reflexive" address)
- The address of a media relay the agent is using.

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

2.1. Gathering Candidate Addresses

In order to execute ICE, an agent has to identify all of its address candidates. Naturally, one viable candidate is one obtained directly from a local interface the client has towards the network. Such a candidate is called a HOST CANDIDATE. The local interface could be one on a local layer 2 network technology, such as ethernet or WiFi, or it could be one that is obtained through a tunnel mechanism, such as a Virtual Private Network (VPN) or Mobile IP (MIP). In all cases, these appear to the agent as a local interface from which ports (and thus a candidate) can be allocated.

If an agent is multihomed, it can obtain a candidate from each interface. Depending on the location of the peer on the IP network relative to the agent, the agent may be reachable by the peer through one of those interfaces, or through another. Consider, for example, an agent which has a local interface to a private net 10 network, and also to the public Internet. A candidate from the net10 interface will be directly reachable when communicating with a peer on the same private net 10 network, while a candidate from the public interface will be directly reachable when communicating with a peer on the public Internet. Rather than trying to guess which interface will

work prior to sending an offer, the offering agent includes both candidates in its offer.

Once the agent has obtained host candidates, it uses STUN to obtain additional candidates. These come in two flavors: translated addresses on the public side of a NAT (SERVER REFLEXIVE CANDIDATES) and addresses of media relays (RELAYED CANDIDATES). The relationship of these candidates to the host candidate is shown in Figure 2. Both types of candidates are discovered using STUN.

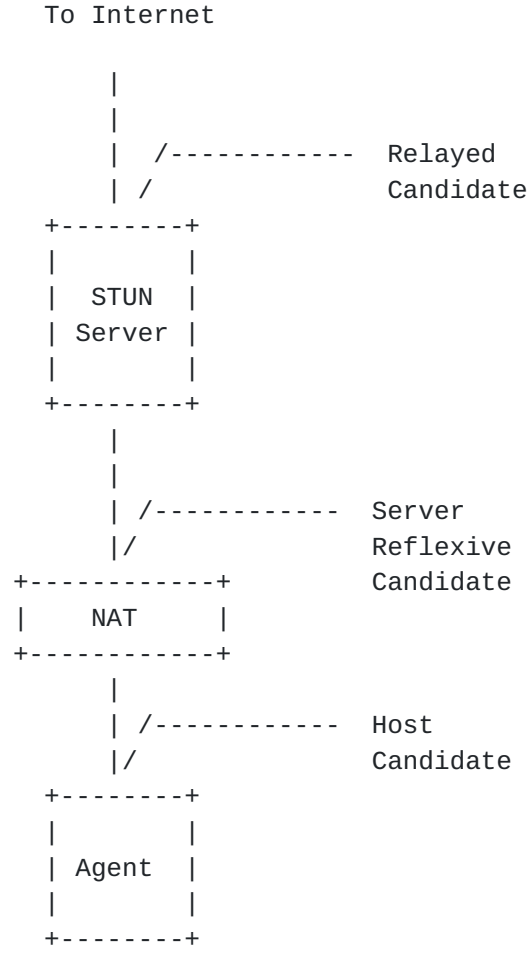


Figure 2

To find a server reflexive candidate, the agent sends a STUN Binding Request, using the Binding Discovery Usage [11] from each host candidate, to its STUN server. (It is assumed that the address of the STUN server is configured, or learned in some way.) When the agent sends the Binding Request, the NAT (assuming there is one) will allocate a binding, mapping this server reflexive candidate to the host candidate. Outgoing packets sent from the host candidate will

be translated by the NAT to the server reflexive candidate. Incoming packets sent to the server relexive candidate will be translated by the NAT to the host candidate and forwarded to the agent. We call the host candidate associated with a given server reflexive candidate the BASE.

Note

"Base" refers to the address you'd send from for a particular candidate. Thus, as a degenerate case host candidates also have a base, but it's the same as the host candidate.

When there are multiple NATs between the agent and the STUN server, the STUN request will create a binding on each NAT, but only the outermost server reflexive candidate will be discovered by the agent. If the agent is not behind a NAT, then the base candidate will be the same as the server reflexive candidate and the server reflexive candidate can be ignored.

The final type of candidate is a RELAYED candidate. The STUN Relay Usage [12] allows a STUN server to act as a media relay, forwarding traffic between L and R. In order to send traffic to L, R sends traffic to the media relay which forwards it to L and vice versa. The same thing happens in the other direction.

Traffic from L to R has its addresses rewritten twice: first by the NAT and second by the STUN relay server. Thus, the address that R knows about and the one that it wants to send to is the one on the STUN relay server. This address is the final kind of candidate, which we call a RELAYED CANDIDATE.

2.2. Connectivity Checks

Once L has gathered all of its candidates, it orders them highest to lowest priority and sends them to R over the signalling channel. The candidates are carried in attributes in the SDP offer. When R receives the offer, it performs the same gathering process and responds with its own list of candidates. At the end of this process, each agent has a complete list of both its candidates and its peer's candidates and is ready to perform connectivity checks by pairing up the candidates to see which pair works.

The basic principle of the connectivity checks is simple:

1. Sort the candidate pairs in priority order.

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

3. Acknowledge checks received from the other agent.

A complete connectivity check for a single candidate pair is a simple 4-message handshake:

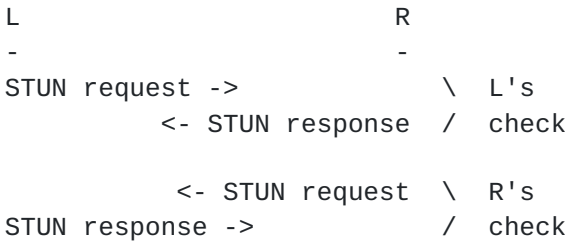


Figure 3

As an optimization, as soon as R gets L's check message he immediately sends his own check message to L on the same candidate pair. This accelerates the process of finding a valid candidate, and is called a triggered check.

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

2.3. Sorting Candidates

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

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

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

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

In general the priority algorithm is designed so that candidates of similar type get similar priorities and so that more direct routes

are preferred over indirect ones. Within those guidelines, however, agents have a fair amount of discretion about how to tune their algorithms.

2.4. Frozen Candidates

The previous description only addresses the case where the agents wish to establish a single media component--i.e., a single flow with a single host-port quartet. However, in many cases (in particular RTP and RTCP) the agents actually need to establish connectivity for more than one flow.

The naive way to attack this problem would be to simply do independent ICE exchanges for each media component. This is obviously inefficient because the network properties are likely to be very similar for each component (especially because RTP and RTCP are typically run on adjacent ports). Thus, it should be possible to leverage information from one media component in order to determine the best candidates for another. ICE does this with a mechanism called "frozen candidates."

The basic principle behind frozen candidates is that initially only the candidates for a single media component are tested. The other media components are marked "frozen". When the connectivity checks for the first component succeed, the corresponding candidates for the other components are unfrozen and checked immediately. This avoids repeated checking of components which are superficially more attractive but in fact are likely to fail.

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

2.5. Security for Checks

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

2.6. Concluding ICE

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

Consequently, ICE assigns one of the agents in the role of the controlling agent, and the other of the controlled agent. The controlling agent runs a selection algorithm, through which it can decide when to conclude ICE checks, and which pairs get selected. The one that is selected is called the favored candidate pair. When a controlling agent selects a pair for a particular component of a media stream, it generates a check for that pair and includes a flag in the check indicating that the pair has been selected. If the controlled agent has already performed in a check in the reverse direction that succeeded, the controlled agent considers ICE processing to be concluded for that component. Once there is a selected pair for each component of a media stream, the ICE checks for that media stream are considered to be completed. At this point, further checks stop for that media stream - ICE is considered to be done. Consequently, media can flow in each direction for that stream, as shown in Figure 4. Once all of the media streams are completed, the controlling endpoint sends an updated offer if the currently in-use candidates don't match the ones it selected.

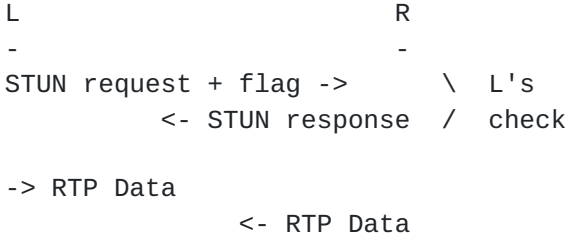


Figure 4

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

2.7. Lite Implementations

In order for ICE to be used in a call, both agents need to support it. However, certain agents, such as those in gateways to the PSTN, media servers, conferencing servers, and voicemail servers, are known to not be behind a NAT or firewall. To make it easier for these devices to support ICE, ICE defines a special type of implementation called "lite" (in contrast to the normal "full" implementation). A lite implementation doesn't gather candidates; it includes only its host candidate for any media stream. When a lite implementation connects with a full implementation, the full agent takes the role of the controlling agent, and the lite agent takes on the controlled role. In addition, lite agents do not need to generate connectivity checks, run the state machines, or compute candidate pairs. For an informational summary of ICE processing as seen by a lite agent, see [33].

3. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [1].

This specification makes use of the following terminology:

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

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

Transport Address: The combination of an IP address and port.

Candidate: A transport address that is to be tested by ICE procedures in order to determine its suitability for usage for receipt of media.

Component: A component is a single transport address that is used to support a media stream. For media streams based on RTP, there are two components per media stream - one for RTP, and one for RTCP.

Host Candidate: A candidate obtained by binding to a specific port from an interface on the host. This includes both physical interfaces and logical ones, such as ones obtained through Virtual Private Networks (VPNs) and Realm Specific IP (RSIP) [18] (which lives at the operating system level).

Server Reflexive Candidate: A candidate obtained by sending a STUN request from a host candidate to a STUN server, distinct from the peer, whose address is configured or learned by the client prior to an offer/answer exchange.

Peer Reflexive Candidate: A candidate obtained by sending a STUN request from a host candidate to the STUN server running on a peer's candidate.

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

Translation: The translation of a relayed candidate is the transport address that the relay will forward a packet to, when one is received at the relayed candidate. For relayed candidates learned through the STUN Allocate request, the translation of the relayed candidate is the server reflexive candidate returned by the Allocate response.

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

Foundation: Each candidate has a foundation, which is an identifier that is distinct for two candidates that have different types, different interface IP addresses for their base, and different IP addresses for their STUN servers. Two candidates have the same foundation when they are of the same type, their bases have the same IP address, and, for server reflexive or relayed candidates, they come from the same STUN server. Foundations are used to correlate candidates, so that when one candidate is found to be valid, candidates sharing the same foundation can be tested next, as they are likely to also be valid.

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

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

In-Use Candidate: A candidate is in-use when it appears in the m/c-line of an active media stream.

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

Check: A candidate pair where the local candidate is a transport address from which an agent can send a STUN connectivity check.

Check List: An ordered set of STUN checks that an agent is to generate towards a peer.

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

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

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

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

Lite: An ICE implementation that omits certain functions, implementing only as much as is necessary for a peer implementation that is full to gain the benefits of ICE. Lite implementations can only act as the controlled agent in a session, and do not gather candidates.

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

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

4. Sending the Initial Offer

In order to send the initial offer in an offer/answer exchange, an agent must gather candidates, priorize them, choose ones for inclusion in the m/c-line, and then formulate and send the SDP. The first of these three steps differ for full and lite implementations.

4.1. Full Implementation Requirements

4.1.1. Gathering Candidates

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

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

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

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

Agents SHOULD obtain relayed candidates and MUST obtain server reflexive candidates. The requirement to obtain relayed candidates is at SHOULD strength to allow for provider variation. If they are not 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.

The agent next pairs each host candidate with the STUN server with

which it is configured or has discovered by some means. This specification only considers usage of a single STUN server. Every Ta seconds, the agent chooses another such pair (the order is inconsequential), and sends a STUN request to the server from that host candidate. If the agent is using both relayed and server reflexive candidates, this request MUST be a STUN Allocate request from the relay usage [12]. If the agent is using only server reflexive candidates, the request MUST be a STUN Binding request using the binding discovery usage [11].

The value of Ta SHOULD be configurable, and SHOULD have a default of 20ms. Note that this pacing applies only to starting STUN transactions with source and destination transport addresses (i.e., the host candidate and STUN server respectively) for which a STUN transaction has not previously been sent. Consequently, retransmissions of a STUN request are governed entirely by the retransmission rules defined in [11]. Similarly, retries of a request due to recoverable errors (such as an authentication challenge) happen immediately and are not paced by timer Ta. Because of this pacing, it will take a certain amount of time to obtain all of the server reflexive and relayed candidates. Implementations should be aware of the time required to do this, and if the application requires a time budget, limit the amount of candidates which are gathered.

An Allocate Response will provide the agent with a server reflexive candidate (obtained from the mapped address) and a relayed candidate in the RELAY-ADDRESS attribute. A Binding Response will provide the agent with only a server reflexive candidate (also obtained from the mapped address). The base of the server reflexive candidate is the host candidate from which the Allocate or Binding request was sent. The base of a relayed candidate is that candidate itself. A server reflexive candidate obtained from an Allocate response is the called the "translation" of the relayed candidate obtained from the same response. The agent will need to remember the translation for the relayed candidate, since it is placed into the SDP. If a relayed candidate is identical to a host candidate (which can happen in rare cases), the relayed candidate MUST be discarded. Proper operation of ICE depends on each base being unique.

Next, the agent eliminates redundant candidates. A candidate is redundant if its transport address equals another candidate, and its base equals the base of that other candidate. Note that two candidates can have the same transport address yet have different bases, and these would not be considered redundant.

Finally, the agent assigns each candidate a foundation. The foundation is an identifier, scoped within a session. Two candidates

MUST have the same foundation ID when they are of the same type (host, relayed, server reflexive, peer reflexive or relayed), their bases have the same IP address (the ports can be different), and, for reflexive and relayed candidates, the STUN servers used to obtain them have the same IP address. Similarly, two candidates MUST have different foundations if their types are different, their bases have different IP addresses, or the STUN servers used to obtain them have different IP addresses.

4.1.2. Prioritizing Candidates

The prioritization process results in the assignment of a priority to each candidate. Each candidate for a media stream MUST have a unique priority. An agent SHOULD compute the priority by determining a preference for each type of candidate (server reflexive, peer reflexive, relayed and host), and, when the agent is multihomed, choosing a preference for its interfaces. These two preferences are then combined to compute the priority for a candidate. That priority SHOULD be computed using the following formula:

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

The type preference MUST be an integer from 0 to 126 inclusive, and represents the preference for the type of the candidate (where the types are local, server reflexive, peer reflexive and relayed). A 126 is the highest preference, and a 0 is the lowest. Setting the value to a 0 means that candidates of this type will only be used as a last resort. The type preference MUST be identical for all candidates of the same type and MUST be different for candidates of different types. The type preference for peer reflexive candidates MUST be higher than that of server reflexive candidates. Note that candidates gathered based on the procedures of Section 4.1.1 will never be peer reflexive candidates; candidates of these type are learned from the STUN connectivity checks performed by ICE. The component ID is the component ID for the candidate, and MUST be between 1 and 256 inclusive. The local preference MUST be an integer from 0 to 65535 inclusive. It represents a preference for the particular interface from which the candidate was obtained, in cases where an agent is multihomed. 65535 represents the highest preference, and a zero, the lowest. When there is only a single interface, this value SHOULD be set to 65535. Generally speaking, if there are multiple candidates for a particular component for a particular media stream which have the same type, the local preference MUST be unique for each one. In this specification, this

only happens for multi-homed hosts.

These rules guarantee that there is a unique priority for each candidate. This priority will be used by ICE to determine the order of the connectivity checks and the relative preference for candidates. Consequently, what follows are some guidelines for selection of these values.

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

Another criteria for selection of preferences is IP address family. ICE works with both IPv4 and IPv6. It therefore provides a transition mechanism that allows dual-stack hosts to prefer connectivity over IPv6, but to fall back to IPv4 in case the v6 networks are disconnected (due, for example, to a failure in a 6to4 relay) [23]. It can also help with hosts that have both a native IPv6 address and a 6to4 address. In such a case, lower local preferences could be assigned to the v6 interface, followed by the 6to4 interfaces, followed by the v4 interfaces. This allows a site to obtain and begin using native v6 addresses immediately, yet still fallback to 6to4 addresses when communicating with agents in other sites that do not yet have native v6 connectivity.

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

a VPN interface would have a higher local preference than any other interface.

Another criteria for selecting preferences is topological awareness. This is most useful for candidates that make use of relays. 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 assign higher local preferences to candidates obtained from closer relays.

4.1.3. Choosing In-Use Candidates

A candidate is said to be "in-use" if it appears in the m/c-line of an offer or answer. When communicating with an ICE peer, being in-use implies that, should these candidates be selected by the ICE algorithm, a re-INVITE will not be required after ICE processing completes. When communicating with a peer that is not ICE-aware, the in-use candidates will be used exclusively for the exchange of media, as defined in normal offer/answer procedures.

An agent MUST choose a set of candidates, one for each component of each active media stream, to be in-use. A media stream is active if it does not contain the a=inactive SDP attribute.

It is RECOMMENDED that in-use candidates be chosen based on the likelihood of those candidates to work with the peer that is being contacted. Unfortunately, it is difficult to ascertain which candidates that might be. As an example, consider a user within an enterprise. To reach non-ICE capable agents within the enterprise, host candidates have 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, relayed candidates from a publically accessible STUN server are needed.

Indeed, the difficulty in picking just one transport 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 agents select relayed candidates to be in-use.

4.2. Lite Implementation

For each media stream, the agent allocates a single candidate for each component of the media stream from one of its interfaces. If an agent is multi-homed, it MUST choose one of its interfaces for a particular media stream; ICE cannot be used to dynamically choose one. Each component has an ID assigned to it, called the component ID. For RTP-based media streams, the RTP itself has a component ID

of 1, and RTCP a component ID of 2. If an agent is using RTCP it MUST obtain a candidate for it.

Each candidate is assigned a foundation. The foundation MUST be different for two candidates from different interfaces (which can occur if media streams are on different interfaces), and MUST be the same otherwise. A simple integer that increments for each interface will suffice. In addition, each candidate MUST be assigned a unique priority amongst all candidates for the same media stream. This priority SHOULD be equal to 2^24*(126) + 2^8*(65535) + 256 minus the component ID, which is 2130706432 minus the component ID. Each of these candidates is also considered to be "in-use", since they will be included in the m/c-line of an offer or answer.

4.3. Encoding the SDP

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

The agent includes a single a=candidate media level attribute in the SDP for each candidate for that media stream. The a=candidate attribute contains the IP address, port and transport protocol for that candidate. 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 candidate attribute also includes the component ID for that candidate. For media streams based on RTP, candidates for the actual RTP media MUST have a component ID of 1, and candidates for RTCP MUST have a component ID of 2. Other types of media streams which require multiple components MUST develop specifications which define the mapping of components to component IDs, and these component IDs MUST be between 1 and 256.

The candidate attribute also includes the priority and the foundation. The agent SHOULD include a type for each candidate by populating the candidate-types production with the appropriate value - "host" for host candidates, "srflx" for server reflexive candidates, "prflx" for peer reflexive candidates (though these never appear in an initial offer/answer exchange), and "relay" for relayed candidates. The related address MUST NOT be included if a type was not included. If a type was included, the related address SHOULD be present for server reflexive, peer reflexive and relayed candidates. If a candidate is server or peer reflexive, the related address is equal to the base for that server or peer reflexive candidate. If the candidate is relayed, the related address is equal to the translation of the relayed address. If the candidiate is a host

candidate, there is no related address and the rel-addr production MUST be omitted.

STUN connectivity checks between agents make use of a short term credential that is exchanged in the offer/answer process. The username part of this credential is formed by concatenating a username fragment from each agent, separated by a colon. Each agent also provides a password, used to compute the message integrity for requests it receives. As such, an SDP MUST contain the ice-ufrag and ice-pwd attributes, containing the username fragment and password respectively. These can be either session or media level attributes, and thus common across all candidates for all media streams, or all candidates for a particular media stream, respectively. However, if two media streams have identical ice-ufrag's, they MUST have identical ice-pwd's. The ice-ufrag and ice-pwd attributes MUST be chosen randomly at the beginning of a session. The ice-ufrag attribute MUST contain at least 24 bits of randomness, and the ice-pwd attribute MUST contain at least 128 bits of randomness. This means that the ice-ufrag attribute will be at least 4 characters long, and the ice-pwd at least 22 characters long, since the grammar for these attributes allows for 6 bits of randomness per character. The attributes MAY be longer than 4 and 22 characters respectively, of course.

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

The m/c-line is populated with the candidates that are in-use. For streams based on RTP, this is done by placing the RTP candidate into the m and c lines respectively. If the agent is utilizing RTCP, it MUST encode the RTCP candidate into the m/c-line 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 [5].

There MUST be a candidate attribute for each component of the media stream in the m/c-line.

Once an offer or answer are sent, an agent MUST be prepared to receive both STUN and media packets on each candidate. As discussed in Section 11.1, media packets can be sent to a candidate prior to its appearence in the m/c-line.

5. Receiving the Initial Offer

When an agent receives an initial offer, it will check if the offeror

supports ICE, determine its role, gather candidates, prioritize them, choose one for in-use, encode and send an answer, and for full implementations, form the check lists and begin connectivity checks.

5.1. Verifying ICE Support

The answerer will proceed with the ICE procedures defined in this specification if the following are true:

- There is at least one a=candidate attribute for each media stream in the offer it just received.

- For each media stream, at least one of the candidates is a match for its respective in-use component in the m/c-line.

If both of these conditions are not met, the agent MUST process the SDP based on normal RFC 3264 procedures, without using any of the ICE mechanisms described in the remainder of this specification with two exceptions. First, in all cases, the agent MUST follow the rules of Section 10, which describe keepalive procedures for all agents. Secondly, if the agent is not proceeding with ICE because there were a=candidate attributes, but none that matched the m/c-line of the media stream, the agent MUST include an a=ice-mismatch attribute in its answer. This mismatch occurs in cases where intermediary elements modify the m/c-line, but don't modify candidate attributes. By including this attribute in the response, diagnostic information on the ICE failure is provided to the offeror and any intermediate signaling entities.

In addition, if the offer contains the "a=ice-lite" attribute, and the answerer is also lite, the agent MUST process the SDP based on normal RFC 3264 procedures, as if it didn't support ICE, with the exception of Section 10, which describes keepalive procedures.

5.2. Determining Role

For each session, each agent takes on a role. There are two roles - controlling, and controlled. The controlling agent is responsible for selecting the candidate pairs to be used for each media stream, and for generating the updated offer based on that selection, when needed. The controlled agent is told which candidate pairs to use for each media stream, and does not generate an updated offer to signal this information in SIP.

If one of the agents is a lite implementation, it MUST assume the controlled role, and its peer (which will be full) MUST assume the controlling role. If the agent and its peer are both full implementations, the agent which generated the offer which started

the ICE processing takes on the controlling role, and the other takes the controlled role.

Based on this definition, once roles are determined for a session, they persist unless ICE is restarted, as discussed below. A restart causes a new selection of roles.

5.3. Gathering Candidates

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

5.4. Prioritizing Candidates

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

5.5. Choosing In Use Candidates

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

5.6. Encoding the SDP

The process for encoding the SDP at the answerer is identical to the process followed by the offerer, as described in Section 4.3.

5.7. Forming the Check Lists

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

There is one check list per in-use media stream resulting from the offer/answer exchange. A media stream is in-use as long as its port is non-zero (which is used in RFC 3264 to reject a media stream). Consequently, a media stream is in-use even if it is marked as a=inactive or has a bandwidth value of zero. Each check list is a sequence of STUN connectivity checks that are performed by the agent. To form the check list for a media stream, the agent forms candidate pairs, computes a candidate pair priority, orders the pairs by priority, prunes them, and sets their states. These steps are

described in this section.

First, the agent takes each of its candidates for a media stream (called local candidates) and pairs them with the candidates it received from its peer (called remote candidates) for that media stream. A local candidate is paired with a remote candidate if and only if the two candidates have the same component ID and have the same IP address version. It is possible that some of the local candidates don't get paired with a remote candidate, and some of the remote candidates don't get paired with local candidates. This can happen if one agent didn't include candidates for the all of the components for a media stream. In the case of RTP, for example, this would happen when one agent provided candidates for RTCP, and the other did not. If this happens, the number of components for that media stream is effectively reduced, and considered to be equal to the minimum across both agents of the maximum component ID provided by each agent across all components for the media stream.

Once the pairs are formed, a candidate pair priority is computed. Let O-P be the priority for the candidate provided by the offerer. Let A-P be the priority for the candidate provided by the answerer. The priority for a pair is computed as:

pair priority = 2^32*MIN(O-P,A-P) + 2*MAX(O-P,A-P) + (O-P>A-P?1:0)

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

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

Each check is also said to have a foundation, which is merely the combination of the foundations of the local and remote candidates in

the check.

Each check in the check list is associated with a state. This state is assigned once the check list for each media stream has been computed. There are five potential values that the state can have:

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

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

Succeeded: This check was already done and produced a successful result.

Failed: This check was already done and failed, either never producing any response or producing an unrecoverable failure response.

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

First, the agent sets all of the checks in each check list to the Frozen state. Then, it takes the first check in the check list for the first media stream (a media stream is the first media stream when it is described by the first m-line in the SDP offer and answer), and sets its state to Waiting. It then finds all of the other checks in that check list with the same component ID, but different foundations, and sets all of their states to Waiting as well. Once this is done, one of the check lists will have some number of checks in the Waiting state, and the other check lists will have all of their checks in the Frozen state. A check list with at least one check that is not Frozen is called an active check list.

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

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

Completed: In this state, the controlling agent has signaled that a candidate pair has been selected for each component. Consequently, no further ICE checks are performed.

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

ICE processing across all media streams also has a state associated with it. This state is equal to Running while checks are in progress. The state is Completed when all checks have been completed. Rules for transitioning between states are described below.

5.8. Performing Periodic Checks

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

An agent performs two types of checks. The first type are periodic checks. These checks occur periodically for each media stream, and involve choosing the highest priority check in the Waiting state from each check list, and performing it. The other type of check is called a triggered check. This is a check that is performed on receipt of a connectivity check from the peer. This section describes how periodic checks are performed.

Once the agent has computed the check lists as described in Section 5.7, it sets a timer for each active check list. The timer fires every Ta/N seconds, where N is the number of active check lists (initially, there is only one active check list). Implementations MAY set the timer to fire less frequently than this. Ta is the same value used to pace the gathering of candidates, as described in Section 4.1.1. The first timer for each active check list fires immediately, so that the agent performs a connectivity check the moment the offer/answer exchange has been done, followed by the next periodic check Ta seconds later.

When the timer fires, the agent MUST find the highest priority check in that check list that is in the Waiting state. The agent then sends a STUN check from the local candidate of that check to the remote candidate of that check. The procedures for forming the STUN request for this purpose are described in Section 7.1.1. If none of the checks in that check list are in the Waiting state, but there are checks in the Frozen state, the highest priority check in the Frozen state is moved into the Waiting state, and that check is performed. When a check is performed, its state is set to In-Progress. If there are no checks in either the Waiting or Frozen state, then the timer for that check list is stopped.

Performing the connectivity check requires the agent to know the username fragment for the local and remote candidates, and the password for the remote candidate. For periodic checks, the remote username fragment and password are learned directly from the SDP received from the peer, and the local username fragment is known by the agent.

6. Receipt of the Initial Answer

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

6.1. Verifying ICE Support

The answerer will proceed with the ICE procedures defined in this specification if there is at least one a=candidate attribute for each media stream in the answer it just received. If this condition is not met, the agent MUST process the SDP based on normal RFC 3264 procedures, without using any of the ICE mechanisms described in the remainder of this specification, with the exception of Section 10, which describes keepalive procedures.

In some cases, the answer may omit a=candidate attributes for the media streams, and instead include an a=ice-mismatch attribute for one or more of the media streams in the SDP. This signals to the offerer that the answerer supports ICE, but that ICE processing was not used for the session because an intermediary modified the m/c-lines without modifying the candidate attributes. See Section 16 for a discussion of cases where this can happen. This specification provides no guidance on how an agent should proceed in such a failure case.

6.2. Determining Role

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

6.3. Forming the Check List

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

6.4. Performing Periodic Checks

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

7. Connectivity Checks

This section describes how connectivity checks are performed. All

ICE implementations are required to be compliant to [11], as opposed to the older [14]. However, whereas a full implementation will both generate checks (acting as a STUN client) and receive them (acting as a STUN server), a lite implementation will only ever receive checks, and thus will only act as a STUN server.

7.1. Client Procedures

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

7.1.1. Sending the Request

The agent acting as the client generates a connectivity check either periodically, or triggered. In either case, the check is generated by sending a Binding Request from a local candidate, to a remote candidate. The agent must know the username fragment for both candidates and the password for the remote candidate.

A Binding Request serving as a connectivity check MUST utilize a STUN short term credential. Rather than being learned from a Shared Secret request, the short term credential is exchanged in the offer/answer procedures. In particular, the username is formed by concatenating the username fragment provided by the peer with the username fragment of the agent sending the request, separated by a colon (":"). The password is equal to the password provided by the peer. For example, consider the case where agent A is the offerer, and agent B is the answerer. Agent A included a username fragment of AFRAG for its candidates, and a password of APASS. Agent B provided a username fragment of BFRAG and a password of BPASS. A connectivity check from A to B (and its response of course) utilize the username BFRAG:AFRAG and a password of BPASS. A connectivity check from B to A (and its response) utilize the username AFRAG:BFRAG and a password of APASS.

An agent MUST include the PRIORITY attribute in its Binding Request. The attribute MUST be set equal to the priority that would be assigned, based on the algorithm in Section 4.1.2, to a peer reflexive candidate learned from this check. Such a peer reflexive candidate has a stream ID, component ID and local preference that are equal to the host candidate from which the check is being sent, but a type preference equal to the value associated with peer reflexive candidates.

The Binding Request sent by an agent MUST include the USERNAME and MESSAGE-INTEGRITY attributes. That is, an agent MUST NOT wait to be challenged for short term credentials. Rather, it MUST provide them

in the Binding Request right away.

The controlling agent MAY include the USE-CANDIDATE attribute in the Binding Request. The controlled agent MUST NOT include it in its Binding Request. This attribute signals that the controlling agent wishes to cease checks for this component, and use the candidate pair resulting from the check for this component. Section 8 provides guidance on determining when to include it.

If the agent is using Diffserv Codepoint markings [26] in its media packets, it SHOULD apply those same markings to its connectivity checks.

7.1.2. Processing the Response

If the STUN transaction generates an unrecoverable failure response or times out, the agent sets the state of the check to Failed. The remainder of this section applies to processing of successful responses (any response from 200 to 299).

The agent MUST check that the source IP address and port of the response equals the destination IP address and port that the Binding Request was sent to, and that the destination IP address and port of the response match the source IP address and port that the Binding Request was sent from. If these do not match, the processing described in the remainder of this section MUST NOT be performed. In addition, an agent sets the state of the check to Failed.

If the check succeeds, processing continues. The agent creates a candidate pair whose local candidate equals the mapped address of the response, and whose remote candidate equals the destination address to which the request was sent. This is called a validated pair, since it has been validated by a STUN connectivity check. It is very important to note that this validated pair will often not be identical to the check itself; in many cases, the local candidate (learned through the mapped address in the response) will be different than the local candidate the request was sent from.

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

attribute in the Binding Request which triggered the check that just completed.

Once the priority of the candidate pair has been computed, the pair is added to the valid list for that media stream. If the agent was a controlling agent, and the check had included a USE-CANDIDATE attribute, the candidate pair is marked as "favored". If the agent was a controlled agent, and the check was a triggered check, and the request which caused the triggered check included the USE-CANDIDATE attribute, the candidate pair is marked as "favored".

Next, the agent updates its ICE states. The agent checks the mapped address from the STUN response. If the transport address does not match any of the local candidates that the agent knows about, the mapped address represents a new peer reflexive candidate. Its type is equal to peer reflexive. Its base is set equal to the candidate from which the STUN check was sent. Its username fragment and password are identical to the candidate from which the check was sent. It is assigned the priority value that was placed in the PRIORITY attribute of the request. Its foundation is selected as described in Section 4.1.1. The peer reflexive candidate is then added to the list of local candidates known by the agent (though it is not paired with other remote candidates at this time).

Next, the agent changes the state for this check to Succeeded. The agent sees if the success of this check can cause other checks to be unfrozen. If the check had a component ID of one, the agent MUST change the states for all other Frozen checks for the same media stream and same foundation, but different component IDs, to Waiting. If the component ID for the check was equal to the number of components for the media stream (where this is the actual number of components being used, in cases where the number of components signaled in the SDP differs from offerer to answerer), the agent MUST change the state for all other Frozen checks for the first component of different media streams (and thus in different check lists) but the same foundation, to Waiting.

7.2. Server Procedures

An agent MUST be prepared to receive a Binding Request on the base of each candidate it included in its most recent offer or answer. Receipt of a Binding Request on a transport address that the agent had included in a candidate attribute is an indication that the connectivity check usage applies to the request.

The agent MUST use a short term credential to authenticate the request and perform a message integrity check. The agent MUST accept a credential if the username consists of two values separated by a

colon, where the first value is equal to the username fragment generated by the agent in an offer or answer for a session in-progress, and the password is equal to the password for that username fragment. It is possible (and in fact very likely) that an offeror will receive a Binding Request prior to receiving the answer from its peer. However, the request can be processed without receiving this answer, and a response generated.

If the agent is using Diffserv Codepoint markings [26] in its media packets, it SHOULD apply those same markings to its responses to Binding Requests.

7.2.1. Additional Procedures for Full Implementations

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

For requests being received on a relayed candidate, the source transport address used for STUN processing (namely, generation of the XOR-MAPPED-ADDRESS attribute) is the transport address as seen by the relay. That source transport address will be present in the REMOTE-ADDRESS attribute of a STUN Data Indication message, if the Binding Request was 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 STUN relay session.

If the STUN request resulted in an error response, no further processing is performed.

Assuming a success response, if the source transport address of the request does not match any existing remote candidates, it represents a new peer reflexive remote candidate. The full-mode agent gives the candidate a priority equal to the PRIORITY attribute from the request. The type of the candidate is equal to peer reflexive. Its foundation is set to an arbitrary value, different from the foundation for all other remote candidates. Note that any subsequent offer/answer exchanges will contain this new peer reflexive candidate in the SDP, and will signal the actual foundation for the candidate. This candidate is then added to the list of remote candidates. However, the agent does not pair this candidate with any local candidates.

Next, the agent constructs a tentative check in the reverse direction, called a triggered check. The triggered check has a local candidate equal to the candidate on which the STUN request was received, and a remote candidate equal to the source transport address where the request came from (which may be a new peer-

reflexive remote candidate). Since both candidates are known to the agent, it can obtain their priorities and compute the candidate pair priority. This tentative check is then looked up in the check list. There can be one of several outcomes:

- If there is already a check on the check list with this same local and remote candidates, and the state of that check is Waiting or Frozen, its state is changed to In-Progress and the tentative check is performed.

- If there is already a check on the check list with this same local and remote candidates, and its state was In-Progress, the agent SHOULD abandon the new tentative check and instead generate an immediate retransmit of the Binding Request for the check in progress. This is to facilitate rapid completion of ICE when both agents are behind NAT.

- If there is already a check on the check list with this same local and remote candidates, and its state was Succeeded, the new tentative check is abandoned. If the Binding Request just received contained the USE-CANDIDATE attribute, it means that the pair resulting from that previous check is favored by the peer controlling agent. The agent MUST take the candidate pair in the valid list that was learned from that previous successful check, and mark it as favored.

- If there is already a check on the check list with this same local and remote candidates, and its state was Failed, the new tentative check is abandoned.

- If there is no matching check on the check list, the new tentative check is inserted into the check list based on its priority, and its state is set to In-Progress.

If the tentative check is to be performed, it is constructed and processed as described in Section 7.1.1. These procedures require the agent to know the username fragment and password for the peer. They are readily determined from the SDP and from the check that was just received. The username fragment for the remote candidate is equal to the bottom half (the part after the colon) of the USERNAME in the Binding Request that was just received. Using that username fragment, the agent can check the SDP messages received from its peer (there may be more than one in cases of forking), and find this username fragment. The corresponding password is then selected. If agent has not yet received this SDP (a likely case for the offerer in the initial offer/answer exchange), it MUST wait for the SDP to be received, and then proceed with the triggered check.

7.2.2. Additional Procedures for Lite Implementations

If the check that was just received contained a USE-CANDIDATE attribute, the agent constructs a candidate pair whose local candidate is equal to the transport address on which the request was received, and whose remote candidate is equal to the source transport address of the request that was received. This candidate pair is assigned an arbitrary priority, and placed into a list of valid candidates for that component of that media stream called the valid list. In addition, it is marked as favored, since the peer agent has indicated that it is to be used. ICE processing is considered complete for a media stream if the valid list contains a candidate pair for each component.

8. Concluding ICE

The processing rules in this section apply only to full implementations.

Concluding ICE involves selection of pairs by the controlling agent, updating of state machinery, and possibly the generation of an updated offer by the controlling agent.

The controlling agent can use any algorithm it likes for deciding when to select a candidate pair, called the favored pair, as the one that will be used for media. However, it MUST eventually include a USE-CANDIDATE attribute in at least one successful check for each component of each media stream.

The most apparent way to utilize the USE-CANDIDATE attribute is to run through a series of checks, each of which omit the flag. Once one or more checks complete successfully for a component of a media stream, the agent evaluates the choices based on some criteria, and picks a candidate pair. The criteria for evaluation is a matter of implementation and it allows for localized optimizations. The check that yielded this pair is then repeated, this time with the USE-CANDIDATE flag. This approach provides the most flexibility in terms of algorithms, and also improves ICE's resilience to variations in implementation (see Section 14. This approach is called "introspective selection". The drawback of introspective selection is that it is guaranteed to increase latencies because it requires an additional check to be done.

An alternative is called "proactive selection". In this approach, the controlling agent includes the USE-CANDIDATE attribute in every check it sends. Once the first check for a component succeeds, it is used by ICE. In this mode, the agent will end up using the candidate

pair which is highest priority based on ICE's prioritization algorithm, instead of some other local optimization. It is possible with proactive selection that multiple checks might succeed with the flag set; this is why ICE still applies its prioritization algorithm to pick amongst those pairs that have been favored.

If an agent is controlling and its peer has a lite implementation, an agent MUST use an introspective selection algorithm. Of course, it MAY select a favored pair based on ICE's prioritization. The key requirement is that the agent must complete a successful check before redoing it with the USE-CANDIDATE attribute.

For both controlling and controlled agents, once a candidate pair in the Valid list is marked as favored, an agent MUST NOT generate any further periodic checks for that component of that media stream, and SHOULD cease any retransmissions in progress for checks for that component of that media stream. Once there is at least one candidate pair for each component of a media stream that is favored, a full-mode agent MUST change the state of processing for its check list to Completed. Once all of the check lists for the media streams enter the Completed state, the controlling agent takes the highest priority favored candidate pair for each component of each media stream. If any of those candidate pairs differ from the in-use candidates in m/c-lines of the most recent offer/answer exchange, the controlling agent MUST generate an updated offer as described in Section 9.

9. Subsequent Offer/Answer Exchanges

An agent MAY generate a subsequent offer at any time. However, the rules in Section 8 will cause the controlling agent to send an updated offer at the conclusion of ICE processing when ICE has selected different candidate pairs from the in-use pairs. This section defines rules for construction of subsequent offers and answers.

9.1. Generating the Offer

An agent MAY change the ice-pwd and/or ice-ufrag for a media stream in an offer. Doing so is a signal to restart ICE processing for that media stream. When an agent restarts ICE for a media stream, it MUST NOT include the a=remote-candidates attribute, since the state of the media stream would not be Completed at this point. Note that it is permissible to use a session-level attribute in one offer, but to provide the same password as a media-level attribute in a subsequent offer. This is not a change in password, just a change in its representation.

An agent MUST restart ICE processing if the offer is being generated for the purposes of changing the target of the media stream. In other words, if an agent wants to generated an updated offer which, had ICE not been in use, would result in a new value for the transport address in the m/c-line, the agent MUST restart ICE for that media stream. This implies that setting the IP address in the c line to 0.0.0.0 will cause an ICE restart. Consequently, ICE implementations SHOULD NOT utilize this mechanism for call hold, and instead use a=inactive as described in [4]

If an agent removes a media stream by setting its port to zero, it MUST NOT include any candidate attributes for that media stream.

An agent MUST NOT signal a change in its implementation level (full or lite) by adding or removing the a=ice-lite attribute from an updated offer, unless ICE processing is being restarted for all media streams in the offer. Of course, in normal cases the implementation level is not dynamic and there would be no need to signal a change. However, in applications like third party call control, which involve a mid-session change in remote correspondent, this can happen and it is permitted by ICE with a restart.

Note that an agent can add a new media stream at any time, even if ICE has long finished for the existing media streams. Based on the rules described here, checks will begin for this new stream as if it was in an initial offer.

9.1.1. Additional Procedures for Full Implementations

This section describes additional procedures for full implementations.

When an agent generates an updated offer, the set of candidate attributes to include for each media stream depend on the state of ICE processing for that media stream. If the processing for that media stream is in the Completed state, a full-mode agent MUST include a candidate attribute for the local candidate of each pair that has been chosen for use by ICE for that media stream. A pair is chosen if it is the highest priority favored pair in the valid list for a component of that media stream. An agent SHOULD NOT include any other candidate attributes for that media stream. If ICE processing for a media stream is in the Running state, the agent MUST include all current candidates (including peer reflexive candidates learned through ICE processing) for that media stream. It MAY include candidates it did not offer previously, but which it has gathered since the last offer/answer exchange. If a media stream is new or ICE checks are restarting for that stream, an agent includes the set of candidates it wishes to utilize. This MAY include some,

none, or all of the previous candidates for that stream in the case of a restart, and MAY include a totally new set of candidates gathered as described in Section 4.1.1.

If a candidate was sent in a previous offer/answer exchange, it SHOULD have the same priority. For a peer reflexive candidate, the priority SHOULD be the same as determined by the processing in Section 7.1.2. The foundation SHOULD be the same. The username fragments and passwords for a media stream SHOULD remain the same as the previous offer or answer.

Population of the m/c-lines also depends on the state of ICE processing. If ICE processing for a media stream is in the Completed state, the m/c-line MUST use the local candidate from the highest priority favored pair in the valid list for each component of that media stream. If ICE processing is in the Running state, a full-mode agent SHOULD populate the m/c-line for that media stream based on the considerations in Section 4.1.3.

In addition, if the agent is controlling, it MUST include the a=remote-candidates attribute for each media stream that is in the Completed state. The attribute contains the remote candidates from the highest priority favored pair in the valid list for each component of that media stream.

9.1.2. Additional Procedures for Lite Implementations

A passive-only agent includes its one and only candidate for each component of each media stream in an a=candidate attribute in any subsequent offer. This candidate is formed identically to the procedures for initial offers, as described in Section 4.2.

9.2. Receiving the Offer and Generating an Answer

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

If the offer contained a change in the a=ice-ufrag or a=ice-pwd attributes compared to the previous SDP from the peer, it is a signal that ICE is restarting for this media stream. If all media streams are restarting, than ICE is restarting overall. Procedures for ICE restarts are discussed below. Unless ICE is restarting for that media stream, an agent MUST NOT change the a=ice-ufrag or a=ice-pwd attributes in an answer relative to the last SDP it provided. Such a

change can only take place in an offer. If ICE is restarting, the a=ice-ufrag and a=ice-pwd attributes MUST be changed.

An agent MUST NOT change its implementation level from its previous SDP unless, based on the offer, ICE procedures are being restarted for all media streams in the offer. In that case, it MAY change its level.

An agent MUST NOT include the a=remote-candidates attribute in an answer.

When the answerer generates its answer, it must decide what candidates to include in the answer, how to populate the m/c-line, and how to adjust the states of ICE processing. The rules for inclusion of candidate attributes in an answer are identical to the rules followed by the offerer as described in Section 9.1 for both full and lite implementations. For lite implementations, those rules also apply for setting the m/c-line. However, additional considerations apply to full implementations.

9.2.1. Additional Procedures for Full Implementations

The computation of the m/c-line additionally depends on the presence or absence of the a=remote-candidates attribute in a media stream. If present, it means that the offerer (acting as the controlling agent) believed that ICE processing has completed for that media stream. In this case, the remote-candidates attribute contains the candidates that the answerer is supposed to use. It is possible that the agent doesn't even know of these candidates yet; they will be discovered shortly through a response to an in-progress check. The full-mode agent MUST populate the m/c-line with the candidates from the a=remote-candidates attribute.

If the offer did not contain the a=remote-candidates attribute, the agent follows the same procedures for populating the m/c-line as described for the offerer in Section 9.1.

9.3. Updating the Check and Valid Lists

If ICE is restarting for a media stream, the agent MUST start a new Valid list for that media stream. However, it retains the old Valid list for the purposes of sending media until ICE processing completes, at which point the old Valid list is discarded and the new one is utilized to determine media and keepalive targets.

9.3.1. Additional Procedures for Full Implementations

The procedures in this section are applicable only to full

implementations.

Once the subsequent offer/answer exchange has completed, each agent needs to determine the impact, if any, on the Check and Valid lists. Unless there is an ICE restart, an offer/answer exchange has no impact on the state of ICE processing for each media stream; that is determined entirely by the checks themselves.

When ICE restarts, an agent MUST flush the check list for the affected media streams, and then recompute the check list and its states as described in Section 5.7.

The remainder of this section describes processing when ICE is not restarting.

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

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

If a media stream existed previously, and remains after the offer/answer exchange, the agent MUST NOT modify the Valid list for that media stream. However, if an agent is in the Running state for that media stream, the check list is updated. To do that, the agent recomputes the check lists using the procedures described in Section 5.7. If a check on the new check lists was also on the previous check lists, and its state was Waiting, In-Progress, Succeeded or Failed, its state is copied over. If a check on the new check lists does not have a state (because it's a new check on an existing check list, or a check on a new check list, or the check was on an old check list but its state was not copied over) its state is set to Frozen.

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

Next, the agent goes through each check list, starting with the highest priority check. If a check has a state of Succeeded, and it has a component ID of 1, then all Frozen checks in the same check list with the same foundation whose component IDs are not one, have

their state set to Waiting. If, for a particular check list, there are checks for each component of that media stream in the Succeeded state, the agent moves the state of all Frozen checks for the first component of all other media streams (and thus in different check lists) with the same foundation to Waiting.

10. Keepalives

All endpoints MUST send keepalives for each media session. These keepalives serve the purpose of keeping NAT bindings active for the media session. These keepalives MUST be sent regardless of whether the media stream is currently inactive, sendonly, recvonly or sendrecv, and regardless of the presence or value of the bandwidth attribute. These keepalives MUST be sent even if ICE is not being utilized for the session at all. The keepalive SHOULD be sent using a format which is supported by its peer. ICE endpoints allow for STUN-based keepalives for UDP streams, and as such, STUN keepalives MUST be used when an agent is communicating with a peer that supports ICE. An agent can determine that its peer supports ICE by the presence of a=candidate attributes for each media session. If the peer does not support ICE, the choice of a packet format for keepalives is a matter of local implementation. A format which allows packets to easily be sent in the absence of actual media content is RECOMMENDED. Examples of formats which readily meet this goal are RTP No-Op [28] and RTP comfort noise [24]. If the peer doesn't support any formats that are particularly well suited for keepalives, an agent SHOULD send RTP packets with an incorrect version number, or some other form of error which would cause them to be discarded by the peer.

If there has been no packet sent on a candidate pair being used for media for Tr seconds (where packets include media and previous keepalives), an agent MUST generate a keepalive on that pair. Tr SHOULD be configurable and SHOULD have a default of 15 seconds.

If STUN is being used for keepalives, a STUN Binding Indication is used [11]. The Binding Indication SHOULD NOT contain integrity checks; since the messages are simply discarded on receipt regardless of contents. The Indication SHOULD NOT contain the PRIORITY or USE-CANDIDATE attributes defined here. The Binding Indication is sent using the same local and remote candidates that are being used for media. An agent receipt a Binding Indication MUST discard it silently. Though Binding Indications are used for keepalives, an agent MUST be prepared to receive Binding Requests as well. If a Binding Request is received, a response is generated as discussed in [11], but there is no impact on ICE processing otherwise.

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

11. Media Handling

11.1. Sending Media

Procedures for sending media differ for full and lite implementations.

11.1.1. Procedures for Full Implementations

Agents always send media using a candidate pair. An agent will send media to the remote candidate in the pair (setting the destination address and port of the packet equal to that remote candidate), and will send it from the local candidate. When the local candidate is server or peer reflexive, media is originated from the base. Media sent from a relayed candidate is sent through that relay, using procedures defined in [12].

If the state of a media stream is Running, there is no old Valid list for that media stream (which would be due to an ICE restart), an agent MUST NOT send media.

When an agent sends media, it MUST send it using the highest priority selected pair for each component in either the old Valid list for a media stream (if it exists), else the new Valid list for that media stream. In several cases, this will not be the same candidate pairs present in the m/c-line. When ICE first completes, if the selected pairs aren't a match for the m/c-line, an updated offer/answer exchange will take place to remedy this disparity. However, until that update offer arrives, there will not be a match. Furthermore, in very unusual cases, the m/c-lines in the updated offer/answer will not be a match.

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

switches transmission of media from one candidate pair to another.

11.1.2. Procedures for Lite Implementations

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

In cases where there has been an ICE restart, there will be an old and a new Valid list. The old Valid list MUST be used by the agent for sending media until the new one is complete, at which point the new one MUST be used, and the old one discarded.

11.2. Receiving Media

ICE implementations MUST be prepared to receive media on any candidates provided in the most recent offer/answer exchange.

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

RFC 3550 [21] describes an algorithm in Section 8.2 for detecting SSRC collisions and loops. These algorithms are based, in part, on seeing different source transport addresses with the same SSRC. However, when ICE is used, such changes will sometimes occur as the media streams switch between candidates. An agent will be able to determine that a media stream is from the same peer as a consequence of the STUN exchange that proceeds media transmission. Thus, if there is a change in source transport address, but the media packets come from the same peer agent, this SHOULD NOT be treated as an SSRC collision.

12. Usage with SIP

12.1. Latency Guidelines

ICE requires a series of STUN-based connectivity checks to take place between endpoints. These checks start from the answerer on generation of its answer, and start from the offerer when it receives the answer. These checks can take time to complete, and as such, the selection of messages to use with offers and answers can effect perceived user latency. Two latency figures are of particular interest. These are the post-pickup delay and the post-dial delay.

The post-pickup delay refers to the time between when a user "answers the phone" and when any speech they utter can be delivered to the caller. The post-dial delay refers to the time between when a user enters the destination address for the user, and ringback begins as a consequence of having succesfully started ringing the phone of the called party.

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

If an offer is received in an INVITE request, the callee SHOULD immediately gather its candidates and then generate an answer in a provisional response. ICE requires that a provisional response with an SDP be transmitted reliably. This can be done through the existing PRACK mechanism [9], or through an optimization that is specific to ICE. With this optimization, provisional responses containing an SDP answer that begins ICE processing for one or more media streams can be sent reliably without RFC 3264. To do this, the agent retransmits the provisional response with th exponential backoff timers described in RFC 3262. Retransmits MUST cease on receipt of a STUN Binding Request for one of the media streams signaled in that SDP or on transmission of a 2xx response. If no Binding Request is received prior to the last retransmit, the agent does not consider the session terminated. Despite the fact that the provisional response will be delivered reliably, the rules for when an agent can send an updated offer or answer do not change from those specified in RFC 3262. Specifically, if the INVITE contained an offer, the same answer appears in all of the 1xx and in the 2xx response to the INVITE. Only after that 2xx has been sent can an updated offer/answer exchange occur. This optimization SHOULD NOT be used if both agents support PRACK. Note that the optimization is very specific to provisional response carrying answers that start ICE processing; it is not a general technique for 1xx reliability.

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

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

However, prior to this point, any media that needs to be sent towards the caller (such as SIP early media [25] cannot be transmitted. For this reason, implementations SHOULD delay alerting the called party

until candidates for each component of each media stream have entered the valid list. In the case of a PSTN gateway, this would mean that the setup message into the PSTN is delayed until this point. Doing this increases the post-dial delay, but has the effect of eliminating 'ghost rings'. Ghost rings are cases where the called party hears the phone ring, picks up, but hears nothing and cannot be heard. This technique works without requiring support for, or usage of, preconditions [6], since its a localized decision. It also has the benefit of guaranteeing that not a single packet of media will get clipped, so that post-pickup delay is zero. If an agent chooses to delay local alerting in this way, it SHOULD generate a 180 response once alerting begins.

In addition to uses where the offer is in an INVITE, and the answer is in the provisional and/or 200 OK, ICE works with cases where the offer appears in the response. In such cases, which are common in third party call control, ICE agents SHOULD generate their offers in a reliable provisional response (which MUST utilize RFC 3262). In that case, the answer will arrive in a PRACK. This allows for ICE processing to take place prior to alerting. Once ICE completes, the agent can alert the user and then generate a 200 OK. The 200 OK would contain no SDP, since the offer/answer exchange has completed. Agents MAY place the offer in a 2xx instead (in which case the answer comes in the ACK). This flow is simpler but results in a poorer user experience.

As discussed in Section 16, offer/answer exchanges SHOULD be secured against eavesdropping and man-in-the-middle attacks. To do that, the usage of SIPS [3] is RECOMMENDED when used in concert with ICE.

12.2. SIP Option Tags and Media Feature Tags

[13] specifies a SIP option tag and media feature tag for usage with ICE. ICE implementations using SIP SHOULD support this specification, which uses a feature tag in registrations to facilitate interoperability through gateways.

12.3. Interactions with Forking

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

With ICE, this problem is resolved. The connectivity checks which occur prior to transmission of media carry username fragments, which in turn are correlated to a specific callee. Subsequent media packets which arrive on the same 5-tuple as the connectivity check

will be associated with that same callee. Thus, the caller can perform this correlation as long as it has received an answer.

12.4. Interactions with Preconditions

Quality of Service (QoS) preconditions, which are defined in RFC 3312 [6] and RFC 4032 [7], apply only to the transport addresses listed in the m/c lines in an offer/answer. If ICE changes the transport address 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 apply without regard to the fact that the m/c lines are changing due to ICE negotiations ocurring "in the background".

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

ICE also has (purposeful) interactions with connectivity preconditions [27]. Those interactions are described there. Note that the procedures described in Section 12.1 describe their own type of "preconditions", albeit with less functionality than those provided by the explicit preconditions in [27].

12.5. Interactions with Third Party Call Control

ICE works with Flows I, III and IV as described in [17]. Flow I works without the controller supporting or being aware of ICE. Flow IV will work as long as the controller passes along the ICE attributes without alteration. Flow II is fundamentally incompatible with ICE; each agent will believe itself to be the answerer and thus never generate a re-INVITE.

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

13. Grammar

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

The candidate attribute is a media-level attribute only. It contains

a transport address for a candidate that can be used for connectivity checks.

The syntax of this attribute is defined using Augmented BNF as defined in RFC 4234 [8]:

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

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

The foundation is composed of one or more ice-char. The component-id is a positive integer, which identifies the specific component for which the transport address is a candidate. It MUST start at 1 and MUST increment by 1 for each component of a particular candidate. The connect-address production is taken from RFC 4566 [10], allowing for IPv4 addresses, IPv6 addresses and FQDNs. The port production is also taken from RFC 4566 [10]. 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) [29].

The cand-type production encodes the type of candidate. This specification defines the values "host", "srflx", "prflx" and "relay" for host, server reflexive, peer reflexive and relayed candidates,

respectively. The set of candidate types is extensible for the future. Inclusion of the candidate type is optional. The rel-addr and rel-port productions convey information the related transport addresses. Rules for inclusion of these values is described in Section 4.3.

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-candidates" attribute is defined using Augmented BNF as defined in RFC 4234 [8]. The remote-candidates attribute is a media level attribute only.

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

The attribute contains a connection-address and port for each component. The ordering of components is irrelevant. However, a value MUST be present for each component of a media stream.

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

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

"ice-lite" is a session level attribute only, and "ice-mismatch" is a media level attribute only. The syntax of the "ice-pwd" and "ice-ufrag" attributes are defined as:

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

The "ice-pwd" and "ice-ufrag" attributes can appear at either the session-level or media-level. When present in both, the value in the media-level takes precedence. Thus, the value at the session level is effectively a default that applies to all media streams, unless overriden by a media-level value.

The "ice-options" attribute is a session level attribute. It

contains a series of tokens which identify the options supported by the agent. Its grammar is:

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

14. Extensibility Considerations

This specification makes very specific choices about how both agents in a session coordinate to arrive at the set of candidate pairs that are selected for media. It is anticipated that future specifications will want to alter these algorithms, whether they are simple changes like timer tweaks, or larger changes like a revamp of the priority algorithm. When such a change is made, providing interoperability between the two agents in a session is critical.

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

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

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

15. Example

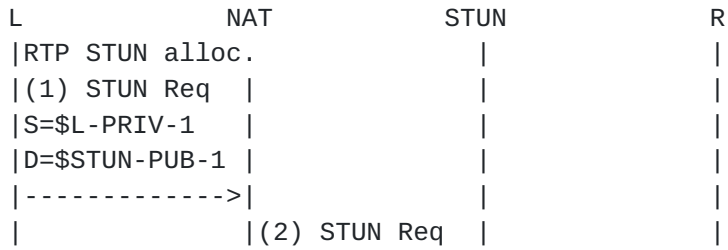
Two agents, L and R, are using ICE. Both are full-mode ICE implementations. Both agents have a single IPv4 interface. For agent L, it is 10.0.1.1, and for agent R, 192.0.2.1. Both are configured with a single STUN server each (indeed, the same one for each), which is listening for STUN requests at an IP address of 192.0.2.2 and port 3478. This STUN server supports only the Binding Discovery usage; relays are not used in this example. Agent L is behind a NAT, and agent R is on the public Internet. The NAT has an endpoint independent mapping property and an address dependent filtering property. The public side of the NAT has an IP address of 192.0.2.3.

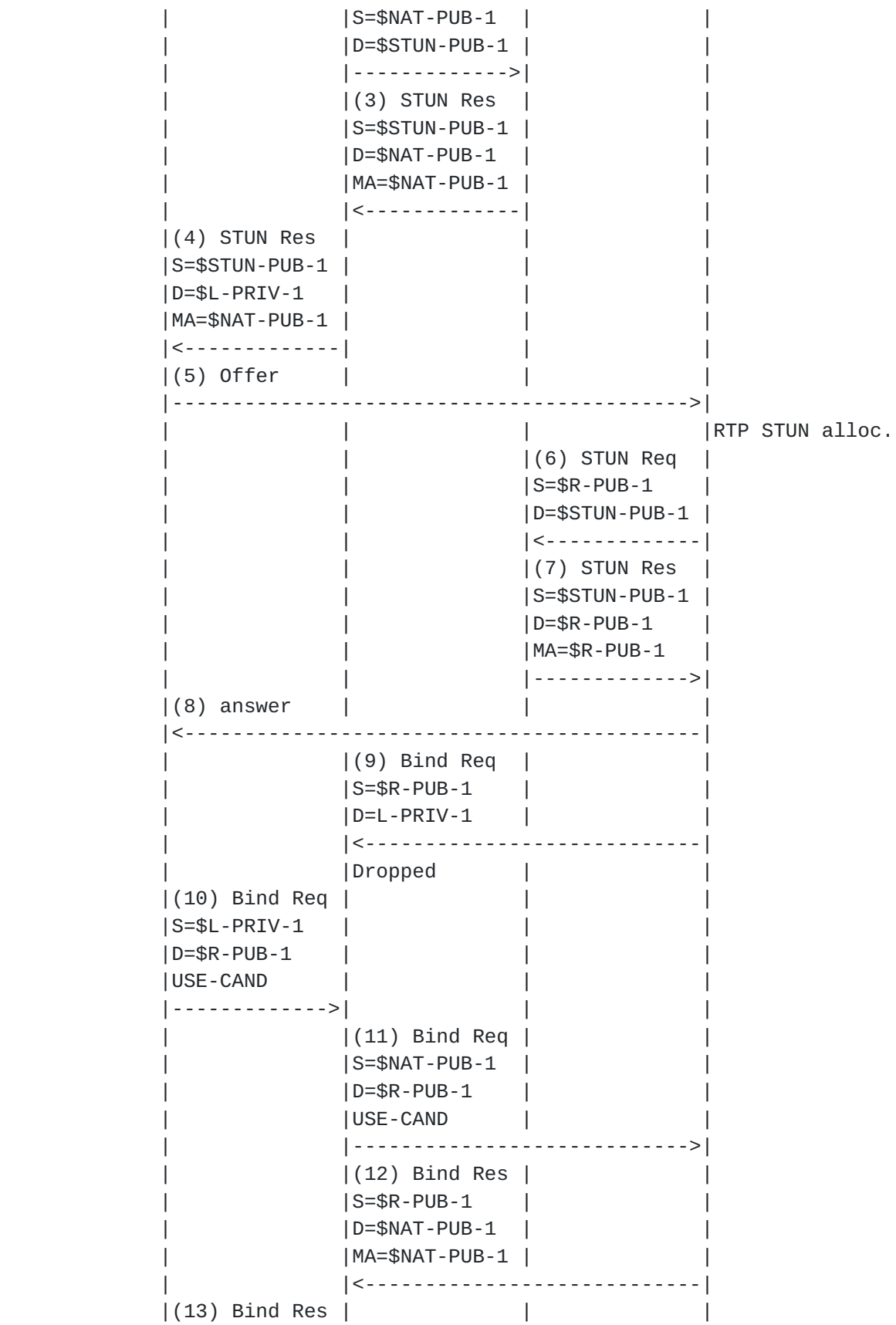
To facilitate understanding, transport addresses are listed using variables that have mnemonic names. The format of the name is entity-type-seqno, where entity refers to the entity whose interface the transport address is on, and is one of "L", "R", "STUN", or "NAT". The type is either "PUB" for transport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

The STUN server has advertised transport address STUN-PUB-1 (which is 192.0.2.2:3478) for the binding discovery usage.

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

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




|S=$NAT-PUB-1 |
|D=$STUN-PUB-1 |
|(3) STUN Res |
|S=$STUN-PUB-1 |
|D=$NAT-PUB-1 |
|MA=$NAT-PUB-1 |
|(4) STUN Res |
|S=$STUN-PUB-1 |
|D=$L-PRIV-1 |
|MA=$NAT-PUB-1 |
|(5) Offer
|RTP STUN alloc.
|(6) STUN Req |
|S=$R-PUB-1 |
|D=$STUN-PUB-1 |
|(7) STUN Res |
|S=$STUN-PUB-1 |
|D=$R-PUB-1 |
|MA=$R-PUB-1 |
|(8) answer |
|(9) Bind Req |
|S=$R-PUB-1 |
|D=L-PRIV-1 |
|Dropped |
|(10) Bind Req |
|S=$L-PRIV-1 |
|D=$R-PUB-1 |
|USE-CAND |
|(11) Bind Req |
|S=$NAT-PUB-1 |
|D=$R-PUB-1 |
|USE-CAND |
|(12) Bind Res |
|S=$R-PUB-1 |
|D=$NAT-PUB-1 |
|MA=$NAT-PUB-1 |
|(13) Bind Res |

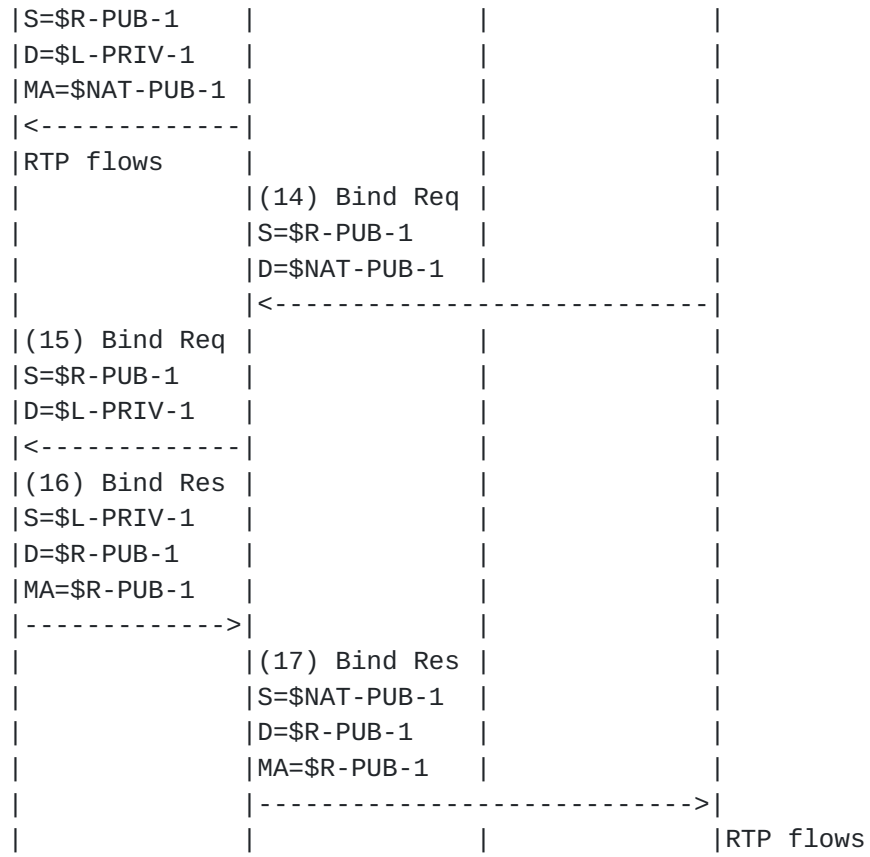


Figure 11

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

Agent L sets a type preference of 126 for the host candidate and 100 for the server reflexive. The local preference is 65535. Based on this, the priority of the host candidate is 2130706178 and for the server reflexive candidate is 1694498562. The host candidate is assigned a foundation of 1, and the server reflexive, a foundation of 2. It chooses its server reflexive candidate as the in-use candidate, and encodes it into the m/c-line. The resulting offer (message 5) looks like (lines folded for clarity):

```
       v=0
       o=jdoe 2890844526 2890842807 IN IP4 $L-PRIV-1.IP
       s=
       c=IN IP4 $NAT-PUB-1.IP
       t=0 0
       a=ice-pwd:asd88fgpdd777uzjYhagZg
       a=ice-ufrag:8hhY
       m=audio $NAT-PUB-1.PORT RTP/AVP 0
       a=rtpmap:0 PCMU/8000
       a=candidate:1 1 UDP 2130706178 $L-PRIV-1.IP $L-PRIV-1.PORT typ local
       a=candidate:2 1 UDP 1694498562 $NAT-PUB-1.IP $NAT-PUB-1.PORT typ srflx
raddr
   $L-PRIV-1.IP rport $L-PRIV-1.PORT
```

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

```
       v=0
       o=jdoe 2890844526 2890842807 IN IP4 10.0.1.1
       s=
       c=IN IP4 192.0.2.3
       t=0 0
       a=ice-pwd:asd88fgpdd777uzjYhagZg
       a=ice-ufrag:8hhY
       m=audio 45664 RTP/AVP 0
       a=rtpmap:0 PCMU/8000
       a=candidate:1 1 UDP 2130706178 10.0.1.1 8998 typ local
       a=candidate:2 1 UDP 1694498562 192.0.2.3 45664 typ srflx raddr
   10.0.1.1 rport 8998
```

This offer is received at agent R. Agent R will obtain a host candidate, and from it, obtain a server reflexive candidate (messages 6-7). Since R is not behind a NAT, this candidate is identical to its host candidate, and they share the same base. It therefore discards this candidate and ends up with a single host candidate. With identical type and local preferences as L, the priority for this candidate is 2130706178. It chooses a foundation of 1 for its single candidate. Its resulting answer looks like:

```
       v=0
       o=bob 2808844564 2808844564 IN IP4 $R-PUB-1.IP
       s=
       c=IN IP4 $R-PUB-1.IP
       t=0 0
       a=ice-pwd:YH75Fviy6338Vbrhrlp8Yh
       a=ice-ufrag:9uB6
```

```
m=audio $R-PUB-1.PORT RTP/AVP 0
```

```
    a=rtpmap:0 PCMU/8000
    a=candidate:1 1 UDP 2130706178 $R-PUB-1.IP $R-PUB-1.PORT typ local
```

With the variables filled in:

```
    v=0
    o=bob 2808844564 2808844564 IN IP4 192.0.2.1
    s=
    c=IN IP4 192.0.2.1
    t=0 0
    a=ice-pwd:YH75Fviy6338Vbrhrlp8Yh
    a=ice-ufrag:9uB6
    m=audio 3478 RTP/AVP 0
    a=rtpmap:0 PCMU/8000
    a=candidate:1 1 UDP 2130706178 192.0.2.1 3478 typ local
```

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

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

Agent R begins its connectivity check (message 9) for the first pair (between the two host candidates). Since R is the passive agent for this session, the check omits the USE-CANDIDATE attribute. The host candidate from agent L is private and behind a different NAT, and thus this check is discarded.

When agent L gets the answer, it performs its one and only connectivity check (messages 10-13). It implements the default algorithm for candidate selection, and thus includes a USE-CANDIDATE attribute in this check. Since the check succeeds, agent L creates a new pair, whose local candidate is from the mapped address in the binding response (NAT-PUB-1 from message 13) and whose remote candidate is the destination of the request (R-PUB-1 from message 10). This is added to the valid list. In addition, it is marked as selected since the Binding Request contained the USE-CANDIDATE

attribute. Since there is a selected candidate in the Valid list for the one component of this media stream, ICE processing for this stream moves into the Completed state. Agent L can now send media if it so chooses.

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

16. Security Considerations

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

16.1. Attacks on Connectivity Checks

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

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

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

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

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

Of the various techniques for creating faked STUN messages described in [11], 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 usually irrelevant since STUN servers are not typically discovered via DNS, they are signaled via IP addresses embedded in SDP. 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 from one of the agents to be sent. When it is, the attacker needs to inject a fake response with an unrecoverable error response, such as a 600. However, since the candidate is, in fact, valid, the original request may reach the peer agent, and result in a success response. The attacker needs to force this packet or its response to be dropped, through a DoS attack, layer 2 network disruption, or other technique. If it doesn't do this, the success response will also reach the originator, alerting it to a possible attack. Fortunately, this attack is mitigated completely through the STUN 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 reflexive candidate result can be done either with fake requests or responses, or with replays. We consider the fake requests and responses case first. It requires the attacker to send a Binding Request to one agent with a source IP address and port for the false candidate. In addition, the attacker must wait for a Binding Request from the other agent, and generate a fake response

with a XOR-MAPPED-ADDRESS attribute containing the false candidate. Like the other attacks described here, this attack is mitigated by the STUN message integrity mechanisms and secure offer/answer exchanges.

Forcing the false peer reflexive candidate result with packet replays is different. The attacker waits until one of the agents sends a check. It intercepts this request, and replays it towards the other agent with a faked source IP address. It must also prevent the original request from reaching the remote agent, either by launching a DoS attack to cause the packet to be dropped, or forcing it to be dropped using layer 2 mechanisms. The replayed packet is received at the other agent, and accepted, since the integrity check passes (the integrity check cannot and does not cover the source IP address and port). It is then responded to. This response will contain a XOR-MAPPED-ADDRESS with the false candidate, and will be sent to that false candidate. The attacker must then intercept it and relay it towards the originator.

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

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

If the attacker themself is identified by the fake candidate the attack is easier to coordinate. However, if SRTP is used [22], 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.

16.2. Attacks on Address Gathering

ICE endpoints make use of STUN for gathering candidates rom a STUN server in the network. This is corresponds to the Binding Discovery usage of STUN described in [11]. As a consequence, the attacks against STUN itself that are described in that specification can still be used against the binding discovery usage when utilized with ICE.

However, the additional mechanisms provided by ICE actually counteract such attacks, making binding discovery 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 [11]. This address will be used as a server reflexive candidate in the ICE exchange. For this candidate to actually be used for media, the attacker must also attack the connectivity checks, and in particular, force a false valid on a false candidate. This attack is very hard to launch if the false address identifies a 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 server reflexive candidate from being used. However, if the peer agent has at least one candidate that is reachable by the agent under attack, the STUN connectivity checks themselves will provide a peer reflexive candidate that can be used for the exchange of media. Peer reflexive candidates are generally preferred over server reflexive candidates. As such, an attack solely on the STUN address gathering will normally have no impact on a session at all.

16.3. 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 [4] apply. These require techniques for message integrity and encryption for offers and answers, which are satisfied by the SIPS mechanism [3] when SIP is used. As such, the usage of SIPS with ICE is RECOMMENDED.

16.4. 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.

16.4.1. The Voice Hammer Attack

The voice hammer attack is an amplification attack. 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 specific 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.

16.4.2. 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 20ms, and each check is a STUN transaction consisting of 7 transmissions of a message 65 bytes in length (plus 28 bytes for the IP/UDP header) that runs for 7.9 seconds, for a total of 58 bytes/second per transaction on average. In the worst case, there can be 395 transactions in progress at once (7.9 seconds divided by 20ms), for a total of 182 kbps, just for STUN requests.

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

16.5. Interactions with Application Layer Gateways and SIP

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

If an ALG is SIP aware but not ICE aware, ICE will work through it as long as the ALG correctly modifies the m/c-lines of SDP. In this case, correctly means that the ALG does not modify m/c-lines with external addresses. If the m/c-line contains internal addresses, but ones for which a public binding exists, the ALG replaces the internal address in the m/c-line with the public binding. Unfortunately, many ALG are known to work poorly in these corner cases. ICE does not try to work around broken ALGs, as this is outside the scope of its functionality. ICE can help diagnose these conditions, which often show up as a mismatch between the set of candidates and the m/c-line. The a=ice-mismatch parameter is used for this purpose.

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

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

17. Definition of Connectivity Check Usage

STUN [11] requires that new usages provide a specific set of information as part of their formal definition. This section meets the requirements spelled out there.

17.1. Applicability

This STUN usage provides a connectivity check between two peers participating in an offer/answer exchange. This check serves to validate a pair of candidates for usage of exchange of media. Connectivity checks also allow agents to discover reflexive candidates towards their peers, called peer reflexive candidates. Finally, connectivity checks serve to keep NAT bindings alive.

It is fundamental to this STUN usage that the addresses and ports used for media are the same ones used for the Binding Requests and responses. Consequently, it will be necessary to demultiplex STUN traffic from whatever the media traffic is. This demultiplexing is done using the techniques described in [11].

17.2. Client Discovery of Server

The client does not follow the DNS-based procedures defined in [11]. Rather, the remote candidate of the check to be performed is used as the transport address of the STUN server. Note that the STUN server is a logical entity, and is not a physically distinct server in this usage.

17.3. Server Determination of Usage

The server is aware of this usage because it signaled this port through the offer/answer exchange. Any STUN packets received on this port will be for the connectivity check usage.

17.4. New Requests or Indications

This usage does not define any new message types.

17.5. New Attributes

This usage defines two new attributes, PRIORITY and USE-CANDIDATE.

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

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

17.6. New Error Response Codes

This usage does not define any new error response codes.

17.7. Client Procedures

Client procedures are defined in Section 7.1.

17.8. Server Procedures

Server procedures are defined in Section 7.2.

17.9. Security Considerations for Connectivity Check

Security considerations for the connectivity check are discussed in Section 16.

18. IANA Considerations

This specification registers new SDP attributes and new STUN attributes.

18.1. SDP Attributes

This specification defines seven new SDP attributes per the procedures of Section 8.2.4 of [10]. The required information for the registrations are included here.

18.1.1. candidate Attribute

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

Attribute Name: candidate

Long Form: candidate

Type of Attribute: media level

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

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

Appropriate Values: See Section 13 of RFC XXXX [Note to RFC-ed: please replace XXXX with the RFC number of this specification].

18.1.2. remote-candidates Attribute

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

Attribute Name: remote-candidates

Long Form: remote-candidates

Type of Attribute: media level

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

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

Appropriate Values: See Section 13 of RFC XXXX [Note to RFC-ed: please replace XXXX with the RFC number of this specification].

18.1.3. ice-lite Attribute

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

Attribute Name: ice-lite

Long Form: ice-lite

Type of Attribute: session level

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

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

Appropriate Values: See Section 13 of RFC XXXX [Note to RFC-ed: please replace XXXX with the RFC number of this specification].

18.1.4. ice-mismatch Attribute

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

Attribute Name: ice-mismatch

Long Form: ice-mismatch

Type of Attribute: session level

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

Purpose: This attribute is used with Interactive Connectivity Establishment (ICE), and indicates that an agent is ICE capable, but did not proceed with ICE due to a mismatch of candidates with the values in the m/c-line.

Appropriate Values: See Section 13 of RFC XXXX [Note to RFC-ed: please replace XXXX with the RFC number of this specification].

18.1.5. ice-pwd Attribute

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

Attribute Name: ice-pwd

Long Form: ice-pwd

Type of Attribute: session or media level

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

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

Appropriate Values: See Section 13 of RFC XXXX [Note to RFC-ed: please replace XXXX with the RFC number of this specification].

18.1.6. ice-ufrag Attribute

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

Attribute Name: ice-ufrag

Long Form: ice-ufrag

Type of Attribute: session or media level

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

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

Appropriate Values: See Section 13 of RFC XXXX [Note to RFC-ed: please replace XXXX with the RFC number of this specification].

18.1.7. ice-options Attribute

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

Attribute Name: ice-options

Long Form: ice-options

Type of Attribute: session level

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

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

Appropriate Values: See Section 13 of RFC XXXX [Note to RFC-ed: please replace XXXX with the RFC number of this specification].

18.2. STUN Attributes

This section registers two new STUN attributes per the procedures in [11].

0x0024 PRIORITY
0x0025 USE-CANDIDATE

19. 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 [20]. ICE is an example of a protocol that performs this type of function. Interestingly, the process for ICE is not unilateral, but bilateral, and the difference has a signficant impact on the issues raised by IAB. Indeed, ICE can be considered a B-SAF (Bilateral Self-Address Fixing) protocol, rather than an UNSAF protocol. Regardless, the IAB has mandated that any protocols developed for this purpose document a specific set of considerations. This section meets those requirements.

19.1. 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.

19.2. Exit Strategy

From RFC 3424, any UNSAF proposal must provide:

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

ICE itself doesn't easily get phased out. However, it is useful even in a globally connected Internet, to serve as a means for detecting whether a router failure has temporarily disrupted connectivity, for example. ICE also helps prevent certain security attacks which have

nothing to do with NAT. However, what ICE does is help phase out other UNSAF mechanisms. ICE effectively selects amongst those mechanisms, prioritizing ones that are better, and deprioritizing ones that are worse. Local IPv6 addresses can be preferred. As NATs begin to dissipate as IPv6 is introduced, server reflexive and relayed candidates (both forms of UNSAF mechanisms) simply never get used, because higher priority connectivity exists to the native host candidates. Therefore, the servers get used less and less, and can eventually be remove when their usage goes to zero.

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

19.3. 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 (as described in RFC 3489 [14]) has several points of brittleness. One of them is the discovery process which requires a agent to try and classify the type of NAT it is behind. This process is error-prone. With ICE, that discovery process is simply not used. Rather than unilaterally assessing the validity of the address, its validity is dynamically determined by measuring connectivity to a peer. The process of determining connectivity is very robust.

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

Another point of brittleness in 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 is removed.

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

Consequently, ICE serves to repair the brittleness introduced in other UNSAF mechanisms, and does not introduce any additional brittleness into the system.

19.4. 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.

19.5. Issues with Existing NAPT Boxes

From RFC 3424, any UNSAF proposal must provide:

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

A number of NAT boxes are now being deployed into the market which try and provide "generic" ALG functionality. These generic ALGs hunt for IP addresses, either in text or binary form within a packet, and rewrite them if they match a binding. This interferes with traditional STUN. However, the update to STUN [11] uses an encoding which hides these binary addresses from generic ALGs. Since [11] is required for all ICE implementations, this NAPT problem does not impact ICE.

Existing NAPT boxes have non-deterministic and typically short expiration times for UDP-based bindings. This requires implementations to send periodic keepalives to maintain those bindings. ICE uses a default of 15s, which is a very conservative estimate. Eventually, over time, as NAT boxes become compliant to behave [31], this minimum keepalive will become deterministic and well-known, and the ICE timers can be adjusted. Having a way to discover and control the minimum keepalive interval would be far

better still.

20. Acknowledgements

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

21. References

21.1. Normative References

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

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

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

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

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

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

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

June 2002.

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

[11] Rosenberg, J., "Simple Traversal Underneath Network Address Translators (NAT) (STUN)", draft-ietf-behave-rfc3489bis-05 (work in progress), October 2006.

[12] Rosenberg, J., "Obtaining Relay Addresses from Simple Traversal Underneath NAT (STUN)", draft-ietf-behave-turn-02 (work in progress), October 2006.

[13] Rosenberg, J., "Indicating Support for Interactive Connectivity Establishment (ICE) in the Session Initiation Protocol (SIP)", draft-ietf-sip-ice-option-tag-00 (work in progress), January 2007.

21.2. Informative References

[14] 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.

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

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

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

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

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

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

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

RFC 3550, July 2003.

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

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

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

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

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

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

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

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

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

[31] Audet, F. and C. Jennings, "NAT Behavioral Requirements for Unicast UDP", draft-ietf-behave-nat-udp-08 (work in progress), October 2006.

[32] Jennings, C. and R. Mahy, "Managing Client Initiated Connections in the Session Initiation Protocol (SIP)", draft-ietf-sip-outbound-07 (work in progress), January 2007.

[33] Rescorla, E., "Overview of the Lite Implementation of Interactive Connectivity Establishment (ICE)", draft-ietf-mmusic-ice-lite-00.txt (work in progress), January 2007.

Appendix A. Lite and Full Implementations

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

Because ICE requires both endpoints to support it in order to bring benefits to either endpoint, incremental deployment of ICE in a network is more complicated. Many sessions involve an endpoint which is, by itself, not behind a NAT and not one that would worry about NAT traversal. Examples include gateways to the PSTN, media servers, conference bridges, and application servers. A very common case is to have one endpoint that requires NAT traversal (such as a VoIP hard phone or soft phone) make a call through one of these devices. Even if the phone supports a full ICE implementation, ICE won't be used at all if the other device doesn't support it. The lite implementation allows for a low-cost entry point for these devices. Once they support the lite implementation, full implementations can connect to them and get the full benefits of ICE.

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

It is important to note that the lite implementation was added to this specification to provide a stepping stone to full implementation. Even for devices that are always connected to the public Internet, a full implementation is preferable if achievable. A full implementation will reduce call setup times. Full implementations also obtain the security benefits of ICE unrelated to NAT traversal; in particular, the voice hammer attack described in Section 16 is prevented only for full implementations, not lite. Finally, it is often the case that a device which finds itself with a public address today will be placed in a network tomorrow where it will be behind a NAT. It is difficult to definitively know, over the lifetime of a device or product, that it will always be used on the public Internet. Full implementation provides assurance that communications will always work.

Appendix B. Design Motivations

ICE contains a number of normative behaviors which may themselves be simple, but derive from complicated or non-obvious thinking or use

cases which merit further discussion. Since these design motivations are not neccesary to understand for purposes of implementation, they are discussed here in an appendix to the specification. This section is non-normative.

B.1. Pacing of STUN Transactions

STUN transactions used to gather candidates and to verify connectivity are paced out at an approximate rate of one new transaction every Ta seconds, where Ta has a default of 20ms. Why are these transactions paced, and why was 20ms chosen as default?

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

B.2. Candidates with Multiple Bases

Section 4.1.1 talks about merging together candidates that are identical but have different bases. When can an agent have two candidates that have the same IP address and port, but different bases? Consider the topology of Figure 17:

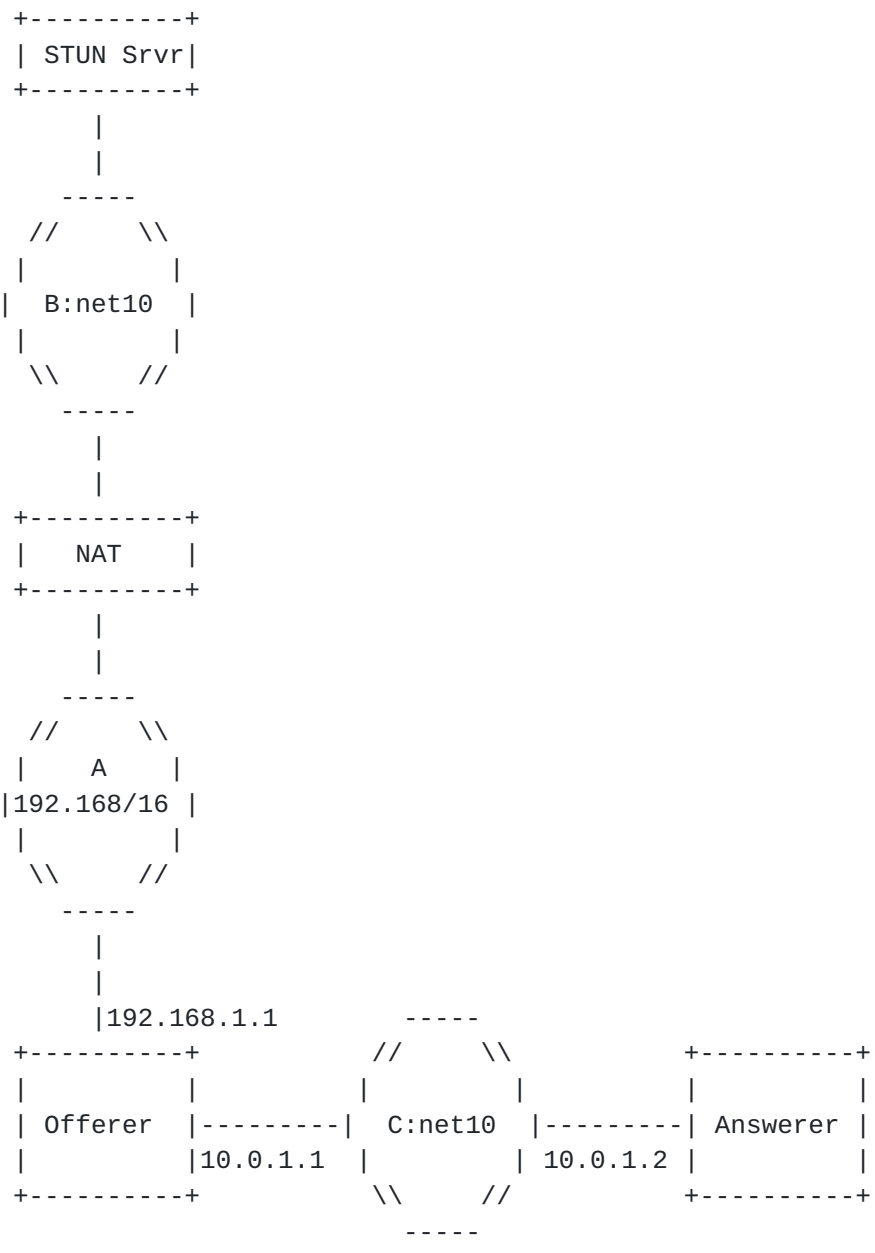


Figure 17

In this case, the offerer is multi-homed. It has one interface, 10.0.1.1, on network C, which is a net 10 private network. The Answerer is on this same network. The offerer is also connected to network A, which is 192.168/16. The offerer has an interface of 192.168.1.1 on this network. There is a NAT on this network, natting into network B, which is another net10 private network, but not connected to network C. There is a STUN server on network B.

The offerer obtains a host candidate on its interface on network C

(10.0.1.1:2498) and a host candidate on its interface on network A (192.168.1.1:3344). It performs a STUN query to its configured STUN server from 192.168.1.1:3344. This query passes through the NAT, which happens to assign the binding 10.0.1.1:2498. The STUN server reflects this in the STUN Binding Response. Now, the offerer has obtained a server reflexive candidate with a transport address that is identical to a host candidate (10.0.1.1:2498). However, the server reflexive candidate has a base of 192.168.1.1:3344, and the host candidate has a base of 10.0.1.1:2498.

B.3. Purpose of the Translation

When a candidate is relayed, the SDP offer or answer contain both the relayed candidate and its translation. However, the translation is never used by ICE itself. Why is it present in the message?

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

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

B.4. Importance of the STUN Username

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

Consider agents 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 host candidates. In this case, those candidates are 10.0.1.1:8866 and 10.0.1.1:8877. As it turns out, B is in a session at that same time, and is also using 10.0.1.1:8866 and 10.0.1.1:8877 as host candidates. This means that 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 and another to 10.0.1.1:8877. 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, the STUN short term credential mechanisms are used. The username fragments are sufficiently random that it is highly unlikely that B would be using the same values as A. Consequently, B would reject the STUN request since the credentials were invalid. In essence, the STUN username fragments provide a form of transient host identifiers, bound to a particular offer/answer session.

An unfortunate consequence of the non-uniqueness of IP addresses is that, in the above example, 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 usually 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.

B.5. The Candidate Pair Sequence Number Formula

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

```
pair priority = 2^32*MIN(O-P,A-P) + 2*MAX(O-P,A-P) + (O-P>A-P?1:0)
```

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

maximum of the two sequence numbers. For pairs that have the same value of the maximum sequence number, the minimum sequence number is used to sort amongst them. If the max and the min sequence numbers are the same, the offerers priority is used as the tie breaker in the last part of the expression. The factor of 2*32 is used since the priority of a single candidate is always less than 2*32, resulting in the pair priority being a "concatenation" of the two component priorities. This creates the desired sorting property.

B.6. The Frozen State

The Frozen state is used for two purposes. Firstly, it allows ICE to first perform checks for the first component of a media stream. Once a successful check has completed for the first component, the other components of the same type and local preference will get performed. Secondly, when there are multiple media streams, it allows ICE to first check candidates for a single media stream, and once a set of candidates has been found, candidates of that same type for other media streams can be checked first. This effectively 'caches' the results of a check for one media stream, and applies them to another. For example, if only the relayed candidates for audio (which were the last resort candidates) succeed, ICE will check the relayed candidates for video first.

B.7. The remote-candidates attribute

The a=remote-candidates attribute exists to eliminate a race condition between the updated offer and the response to the STUN Binding Request that moved a candidate into the Valid list. This race condition is shown in Figure 18. On receipt of message 4, agent A adds a candidate pair to the valid list. If there was only a single media stream with a single component, agent A could now send an updated offer. However, the check from agent B has not yet generated a response, and agent B receives the updated offer (message 7) before getting the response (message 10). Thus, it does not yet know that this particular pair is valid. To eliminate this condition, the actual candidates at B that were selected by the offerer (the remote candidates) are included in the offer itself. Note, however, that agent B will not send media until it has received this STUN response.

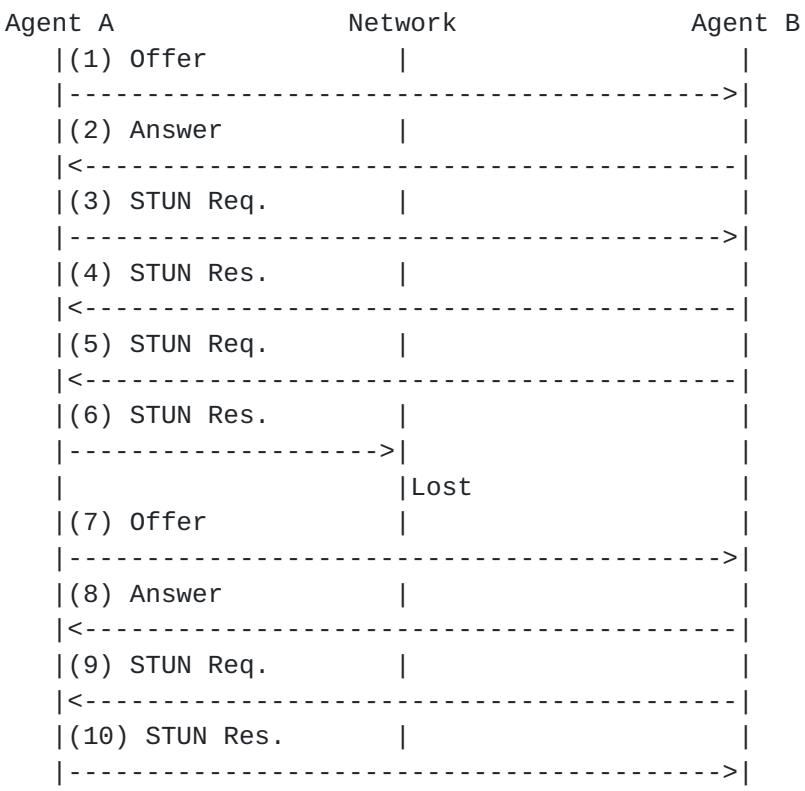


Figure 18

B.8. Why are Keepalives Needed?

Once media begins flowing on a candidate pair, it is still necessary to keep the bindings alive at intermediate NATs for the duration of the session. Normally, the media stream packets themselves (e.g., RTP) meet this objective. However, several cases merit further discussion. Firstly, in some RTP usages, such as SIP, the media streams can be "put on hold". This is accomplished by using the SDP "sendonly" or "inactive" attributes, as defined in RFC 3264 [4]. 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 [30], may send packets so infrequently that the interval exceeds the NAT binding timeouts.

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

For these reasons, the media packets themselves cannot be relied

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

B.9. Why Prefer Peer Reflexive Candidates?

Section 4.1.2 describes procedures for computing the priority of candidate based on its type and local preferences. That section requires that the type preference for peer reflexive candidates always be lower than server reflexive. Why is that? The reason has to do with the security considerations in Section 16. It is much easier for an attacker to cause an agent to use a false server reflexive candidate than it is for an attacker to cause an agent to use a false peer reflexive candidate. Consequently, attacks against the STUN binding discovery usage are thwarted by ICE by preferring the peer reflexive candidates.

B.10. Why Send an Updated Offer?

Section 11.1 describes rules for sending media. Both agents can send media once ICE checks complete, without waiting for an updated offer. Indeed, the only purpose of the updated offer is to "correct" the m/c-line so that it matches where media is being sent, based on ICE procedures.

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

B.11. Why are Binding Indications Used for Keepalives?

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

The primary reason has to do with network QoS mechanisms. Once media

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

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

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.