MMUSIC
Internet-Draft
Expires: April 19, 2004

J. Rosenberg
dynamicsoft

October 20, 2003

Interactive Connectivity Establishment (ICE): A Methodology for Network Address Translator (NAT) Traversal for the Session Initiation Protocol (SIP)

draft-ietf-mmusic-ice-00

Status of this Memo

Copyright Notice

Abstract

This document describes a methodology for Network Address Translator (NAT) traversal for multimedia session signaling protocols, such as the Session Initiation Protocol (SIP). This methodology is called Interactive Connectivity Establishment (ICE). ICE is not a new protocol, but rather makes use of existing protocols, such as Simple Traversal of UDP Through NAT (STUN), Traversal Using Relay NAT (TURN) and even Real Specific IP (RSIP). ICE works through the mutual cooperation of both endpoints in a session.

Table of Contents

1. Introduction

A multimedia session signaling protocol is a protocol that exchanges messages between a pair of agents for the purposes of establishing the flow of media traffic between them. This media flow is distinct from the flow of control messages, and may take a different path through the network. Examples of such protocols are the Session Initiation Protocol (SIP) [4], the Real Time Streaming Protocol (RTSP) [5] and ITU H.323.

These protocols, by nature of their design, are difficult to operate through Network Address Translation (NAT). Because their purpose in life is to establish a flow of packets, they tend to carry IP addresses within their messages, which is known to be problematic through NAT [6]. The protocols also aim for peer to peer media flow, in order to reduce latency, which is also difficult to accomplish through NAT. Many other aspects of these protocols are fundamentally incompatible with 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 [7], Simple Traversal of UDP through NAT (STUN) [1], Traversal Using Relay NAT [14], Realm Specific IP [8][9], symmetric RTP [10], along with session description extensions needed to make them work, such as [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 a lot of 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. It is called Interactive Connectivity Establishment, or ICE. ICE makes use of many of the protocols above, but uses them in a specific methodology which avoids many of the pitfalls of using any one alone. ICE is not a new protocol, and does not require extensions from STUN, TURN or RSIP. However, it does require additional signaling capabilities to be introduced into the multimedia session signaling protocols. For those protocols which make use of the Session Description Protocol (SDP), this specification defines the necessary extensions to it. Other protocols will need to define their own mechanisms.

2. Multimedia Signaling Protocol Abstraction

This specification defines a general methodology that allows the media streams of multimedia signaling protocols to successfully traverse NAT. This methodology is independent of any particular signaling protocol. In order to discuss the methodology, we need to to define an abstraction of a multimedia signaling system, and define terms that can be used throughout this specification. Figure 1 shows the abstraction.

```
                              +-----------+
                              |           |
                              |           |
                            > | Signaling |\
                           /  | Relay     | \
                          /   |           |  \
              Initiate   /    |           |   \   Initiate
             Message    /   / +-----------+    \  Message
                       /   /                <   \
                      /   /                  \   \
                     /   /                    \   \
                    /   / Accept       Accept  \   \
                   /   /   Message     Message  \   >
                  /   /                           \
     +-----------+  /                             \   +-----------+
     |           | <                                |           |
     |           |             Media Stream         |           |
     |  Session  | ................................ | Session   |
     | Initiator |                                  | Responder |
     |           |             Media Stream         |           |
     |           | ................................ |           |
     +-----------+                                  +-----------+
```

Figure 1

Communications occur between two clients - the session initiator and the session responder, also referred to as the initiator and responder. The initiator is the one that decides to engage in communications. To do so, it sends an initiate message. The initiate message contains parameters that describe the capabilities and configuration of media streams for the initiator. This message may travel through signaling intermediaries, called a signaling relay, before finally arriving at the session responder. Assuming the session responder wishes to communication, it generates an accept message, which is relayed back to the initiator. This message contains capabilities and configuration of media streams for the

responder. As a result, media streams are established between the initiator and responder. The signaling protocol may also support an operation that allows for modification of the media stream parameters after establishment. We refer to this signaling messages as a modify message. Its positive response is a modify acceptance message. The signaling protocol may also support an operation that allows for termination of the communications session. We refer to this signaling message as a terminate message.

This abstraction is readily mapped to SIP, RTSP, and H.323, amongst others. For SIP, the initiator is the User Agent Client (UAC), the responder is the User Agent Server (UAS), the initiate message is an INVITE containing an SDP offer, the accept message is a 200 OK containing an SDP answer, the modify message is an INVITE with an offer, the modify acceptance response is a 200 OK with an answer, and the terminate message is a BYE. For RTSP, the initiator is the RTSP client, the responder is the RTSP server, the initiate message is a SETUP message, and the accept message is a SETUP response. The modify message is a SETUP message, and the modify acceptance message is a SETUP response.

This specification defines parameters that need to be included in these various signaling messages in order to implement the functionality described by ICE. Those parameters are represented in XML for convenience. Any multimedia signaling protocol that uses ICE will need to define how to map those parameters into its own protocol messages. Section 9 provides such a mapping for SIP.

3. Terminology

Several new terms are introduced in this specification:

Session Initiator: A software entity that, at the request of a user, tries to establish communications with another entity, called the session responder. A session initiator is also called an initiator.

Initiator: Another term for a session initiator.

Session Responder: A software entity that receives a request for establishment of communications from the session initiator, and either accepts or declines the request. A session responder is also called a responder.

Responder: Another term for a session responder.

Client: Either the initiator or responder.

Peer: From the perspective of one of the clients in a session, its peer is the other client. Specifically, from the perspective of the initiator, the peer is the responder. From the perspective of the responder, the peer is the initiator.

Signaling Relay: An intermediary of signaling messages. Examples are SIP proxies and H.323 Gatekeepers.

Initiate Message: The signaling message used by an initiator to establish communications. It contains capabilities and other information needed by the responder to send media to the initiator.

Accept Message: The signaling message used by a responder to agree to communications. It contains capabilities and other information needed by the initiator to send media to the responder.

Modify Message: The signaling message used by either an initiator or responder to change the capability and other information needed by the peer for sending media.

Modify Acceptance Message The signaling message used by a client to agree to the changes proposed in a modify message, and to present the capability or other information needed by its peer for sending media.

Terminate Message The signaling message used by a client to terminate the session and associated media streams.

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

Local Transport Address: A local transport address is transport address that has been allocated from the operating system on the host. This includes transport addresses obtained through VPNs, and also transport addresses obtained through RSIP (which lives at the operating system level). Transport addresses are typically obtained by binding to an interface.

Derived Transport Address: A derived transport address is a transport address which is associated with, but different from, a local transport address. The derived transport address is associated with the local transport address in that packets sent to the derived transport address are received on the socket bound to that local transport address. Derived addresses are obtained using protocols like STUN and TURN, and more generally, any UNSAF protocol [11].

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

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

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

Unilateral Allocations: Queries made to a network server which provides an UNSAF service.

Bilateral Allocations: Addresses obtained by using an UNSAF service that actually runs on the peer of the communications session. Peer derived transport addresses are synonymous with bilateral allocations.

4. Overview of ICE

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

The key assumption in ICE is that a client cannot know, apriori, whether the peer it wishes to communicate with is connected to one or all of the address realms it is in. Therefore, in order to communicate, it has to try them all, and choose the best one that works.

Before the initiator establishes a session, it obtains as many IP address and port combinations in as many address realms as it can. These adresses all represent potential points at which the initiator will receive a specific media stream. Any protocol that provides a client with an IP address and port on which it can receive traffic can be used. These include STUN, TURN, RSIP, and even a VPN. The client also uses any local interface addresses. A dual-stack v4/v6 client will obtain both a v6 and a v4 address/port. The only requirement is that, across all of these addresses, the initiator can be certain that at least one of them will work for any responder it might communicate with. This is easily guaranteed by using TURN, RSIP, MIDCOM or a VPN from a server on the public Internet to obtain one of the addresses.

The initiator then makes a STUN server available on each of the address/port combinations it has obtained. This STUN server is running locally, on the initiator. All of these addresses are placed into the initiate message, and they are ordered in terms of preference. Local IPv6 addresses always have the highest preference, followed by local IPv4 addresses, followed by STUN-allocated addresses, followed last by addresses allocated through protocols using relays, such as TURN and VPN. The initiate message also conveys the STUN username and password which are required to gain access to the STUN server on each address/port combination.

The initiate message is sent to the responder. This specification does not address the issue of how the signaling messages themselves traverse NAT. It is assumed that signaling protocol specific mechanisms are used for that purpose. Once the responder receives the initiate message, it sends STUN requests to each alternate address/

port in the initiate message. These STUN requests include the username and password obtained from the initiate message. None of the STUN flags are used. The STUN requests serve two purposes. The first is to check for connectivity. If a response is received, the responder knows that it can reach the initiator at that address. The second purpose is to obtain more addresses at which the responder can be contacted. If there were NATs between the responder and initiator, the responder may discover another address through the STUN responses. In its accept message, the responder includes all addresses that it can unilaterally determine (just as the initiator did), in addition to any that were discovered using the STUN messages to the initiator.

When the accept message arrives at the initiator, the initiator performs a similar operation. Using STUN, it checks connectivity to each of the addresses in the accept message. Through the STUN responses, it may learn of additional addresses that it can use to receive media. It can therefore generate a modify message to pass this address to the responder. Generally, at the end of the first exchange, both sides will have discovered one of more addresses which they are capable of successfully sending media to. Each side uses the most preferred address amongst the ones which worked.

In some cases, an additional exchange will be required.

5. Detailed ICE Algorithm

This section describes the detailed processing needed for ICE.

5.1 Gathering Transport Addresses

The initiator begins the process by gathering transport addresses. There are two types of addresses it can gather - local transport addresses, and derived transport addresses. Local transport addresses are obtained by binding to an ephemeral port on an interface (physical or virtual) on the host. A multi-homed host SHOULD attempt to bind on all interfaces for all media streams it wishes to receive. For media streams carried using the Real Time Transport Protocol (RTP) [12], the initiator will need to bind to an ephemeral port for both RTP and RTCP.

The result will be a set of local transport addresses. The initiator may also have access to servers that provide unilateral self-address fixing (UNSAF) [11]. Examples of such protocols include STUN, TURN, and TEREDO [13]. For each of these protocols, the initiator may have access to a multiplicity of servers. For example, a user connected to a natted cable access network might have access to a STUN server in the private cable network and in the public Internet. For each local transport address, the initiator SHOULD obtain an address from every server for each protocol it supports. The result of this will be a set of derived transport addresses, with each derived address associated with the local transport address it is derived from.

ICE works better the more options exist for connectivity. However, in order to communicate with the peer, at least one of the offered addresses has to be guaranteed to work with any peer that might be called. This generally requires that at least one of the derived addresses be obtained from a relay service (such as TURN) that exist within the public Internet. ICE requires that a client know, through configuration, which of the derived transport addresses is coming from a provider on the public Internet.

5.2 Enabling STUN on Each Transport Address

Once the initiator has obtained a set of transport addresses, it starts a STUN server on each local transport address (including ones used for RTCP). This, by definition, means that the STUN service will be reached for requests sent to the derived addresses.

However, the client does not need to provide STUN service on any other IP address or port, unlike the normal STUN usage as described in [1]. The need to run the service on multiple ports is to support the change flags. However, those flags are not needed with ICE, and

the server SHOULD reject any requests with these flags set.

Furthermore, there is no need to support TLS or to be prepared to receive SharedSecret request messages. Those messages are used to obtain shared secrets to be used with BindingRequests. However, with ICE, usernames and passwords are exchanged in the signaling protocol.

It is important to note that the client will receive both STUN requests and media packets on each local transport address. This will require the initiator to disambiguate STUN messages from messages for the underlying media stream protocol. In the case of RTP/RTCP, this disambiguation is easy. RTP and RTCP packets start with the bits 0b10 (v=2). The first two bits in STUN are always 0b00. Disambiguating STUN with other media stream protocols may be more complicated. However, it is guaranteed to always be possible by selecting an appropriately random username (see below).

The need to run STUN on the same transport address as the media stream represents the "ugliest" piece of ICE. However, it is an essential part of the story. By sending STUN requests to the very same place media is sent, any bindings learned through STUN will be useful even when communicating through symmetric NATs. This results in a substantial increase in the scope of applicability of STUN, in terms of cases where it can provide connectivity. In that sense, the usage of STUN here is radically different than the usage models outlined in [1], where STUN is generally useless for dealing with symmetric NAT.

For each local transport address where a STUN server is running, the client MUST choose a username and password. The username MUST be globally unique, so that no other host will select a username with the same value. This username and password will be passed to the responder in the initiate message. They are used by the responder to authenticate the STUN requests to the server.

The global uniqueness requirement stems from the lack of uniquenes afforded by IP addresses. Consider clients 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 initiates communications to C. C, in its accept message, provides A with its transport addresses. In this case, thats 10.0.1.1:8866 and 8877. As it turns out, B is in a session at that same time, and is also using 10.0.1.1:8866 and 8877. This means that B has a STUN server running on those ports, just as C does. A will send a STUN request to 10.0.1.1:8866 and 8877. However, these do not go to C as expected. Instead, they go to B. If B just replied to them, A would believe it has connectivity to C, when in fact it has connectivity to a

completely different user, B. To fix this, the STUN username takes on the role of a unique identifier. C provides A with a unique username. A uses this username in its STUN query to 10.0.1.1:8866. This STUN query arrives at B. However, the username is unknown to B, and so the request is rejected. A treats the rejected STUN request as if there were no connectivity to C (which is actually true). Therefore, the error is avoided.

Once the STUN server is started, it MUST run until the first media packet arrives on that address. Once that occurs, the agent MAY terminate the server. It is still possible that a late or lost STUN message will show up, but these will generally fail any media stream validity checks and be discarded (STUN packets always fail the RTP validity checks).

While the server is running, it MUST act as a normal STUN server, but MUST only accept STUN requests from clients that authenticate using the username and password handed out for the dialog.

5.3 Prioritizing the Transport Addresses

With the STUN servers starting, the next step is to prioritize the transport addresses. This priority reflects the desire that the UA has to receive media on that address, and is assigned as a value from 0 to 1 (1 being most preferred). Although any prioritization is possible, it is RECOMMENDED that the prioritization be based on the number of intermediaries that will be traversed. The fewer intermediaries, the higher the priority. Furthermore, it is RECOMMENDED that, given an equal number of intermediaries, an IPv6 address receive higher priority than an IPv4 address. As a result of this, local IPv6 transport addresses obtained from physical interfaces have highest priority. Next are local IPv4 transport addresses obtained from physical interfaces. Next are STUN derived transport addresses, followed by TURN, RSIP or TEREDO derived transport addresses. Peer derived transport addresses obtained through STUN requests sent through a TURN relay using the SEND command have a priority equal to TURN derived transport addresses. Last up are local transport addresses obtained from VPN interfaces.

Ordering of transport addresses within any one of the groups above (i.e., addresses obtained from relay services, such as TURN or RSIP), is more complicated. When a device is configured to use a multiplicity of these relays, each from the same provider, it is RECOMMENDED that the client be configured with the relative preference for each. This is useful, for example, in enterprises with multiple layers of NAT, each of which hosts a relay. In such a case, the preference for each relay would normally decrease as the relays move farther away from the client.

5.4 Constructing the Initiate Message

The next step is to construct the initiate message. Section 7 provides the XML schema for the initiate message. The message consists of a series of media streams. For each media stream, there is a default address and a list of alternates. The default address is the one that will be used by responders that don't understand ICE. This is mapped to the address currently conveyed in the signaling messages of SIP, H.323 and RTSP. The alternates provide additional points of contact.

For each media stream, the client chooses the transport address which has the highest probability of working with any arbitrary peer. Generally, this will be a transport address learned from a relay service on the public Internet, such as TURN. Frequently this is also the lowest priority transport address. This transport address is placed into default address in the initiate message. This is the address that will be used by a peer that doesn't understand ICE. Therefore, to maximize the ability to complete a call, the address which is most likely to succeed is used.

The client then encodes all of its available transport addresses (including the one that was set as the default) as a series of alternate elements. Each alternate element conveys a transport address for RTP, one for RTCP, the STUN username and STUN password. The client MUST assign each alternate a unique identifier. These identifiers MUST be unique across all alternates used within the session. This identifier is encoded in the "id" attribute of the alternate element. The priority for the transport address, as computed above, is included as an attribute as well.

Once the initiate message is constructed, it is sent.

5.5 Responder Processing - Connectivity Checks and Gathering

Once the responder receives the initiate message, it does several things in parallel. First, it performs the same processing described in Section 5.1. Specifically, for each media stream in the initiate message, the responder allocates a set of local transport addresses and the full set of derived transport addresses.

Note that these addresses can be "pre-gathered" before the call is even received, so that a set is always "on-deck". This will avoid any increase in call setup times, at the expense of holding onto addresses which may not get used. Retaining these addresses may also require refresh traffic that consumes network bandwidth.

While the unilateral derived addresses are being obtained, the

responder sends a STUN BindingRequest from each local transport address to each STUN server identified in an alternate in the initiate message. This BindingRequest MUST contain the USERNAME attribute, and the value of the USERNAME MUST equal the username from that alternate element. Similarly, the BindingRequest MUST contain a PASSWORD attribute, and the value of the PASSWORD MUST equal the password from the alternate element. The BindingRequest MUST contain a MESSAGE-INTEGRITY attribute, computed using the username and password from the alternate element in the initiate message. The BindingRequest MUST NOT contain the CHANGE-REQUEST or RESPONSE-ADDRESS attribute.

It is RECOMMENDED that these STUN requests be sent in parallel. The responder MAY alert the user during this time. Generally, if the user is a human (and not an automata), the STUN transactions will have completed before the call is answered.

If the responder had obtained an address from TURN, it MAY use the TURN SEND primitive to relay STUN BindingRequest messages, in addition to sending them from the associated local transport address. Generally, this would be done only by clients in networks that prevent the client from sending outbound traffic directly to the public Internet. These networks frequently require outbound traffic to pass through some kind of intermediary. A TURN server can play the role of such an intermediary. It is RECOMMENDED that, whether a client should or should not also relay its STUN BindingRequests through the TURN server, be configurable.

When STUN BindingRequest messages are being sent through a TURN server, the ordering of alternates which are tried makes a difference. When one of these BindingRequest messages elicits a response, the response packet causes the TURN server to lock down towards that alternate. Once locked down, the relay cannot be used for receiving traffic from other addresses. If the lock down occurs to an alternate with low preference, the result is sub-optimal. To avoid this, instead of trying each alternate in parallel, it is RECOMMENDED that the client try the addresses sequentially, starting with the alternate with the highest preference value.

OPEN ISSUE: Trying this sequentially is ugly. Any alternatives I was able to come up with required the client to control lock down. Once this happens, it becomes possible to misuse TURN to run public services behind a NAT, which is considered a non-starter. Is there some other way?

In all cases, if the STUN BindingRequest elicits a BindingResponse before the STUN transaction times out, the result is considered a success. For successful transactions, the responder stores the

transport address from the initiate message (which identifies both the STUN server and the place where media is sent), the local transport address from which the STUN request was sent, the id of that alternate from the initiate message, and the preference from that alternate. If the STUN transaction succeeds, the client knows for certain that the media can reach the destination as well. That is because the media traffic will be sent from the same transport address, to the same trasport address, as the STUN packet was.

When a client receives an initiate message, it MAY begin sending media immediately to the address and port specified in the default. If that address and port was also listed as an alternate, the client MUST send media from the same address and port used to send a STUN request to the peer. As the STUN transactions begin to complete, the client begins to learn which alternates it has connectivity to. If one of those alternates has a higher priority than the one currently in use, that transport address MUST be used instead (along with its corresponding local address). Note that, between two transport addresses with the same preference, a STUN derived address MUST be used. Furthermore, once a client sends media to a transport address with a specified priority, it MUST NOT, during the lifetime of the session, send media to a connected transport address with a lower priority.

This restriction allows a client to free derived transport addresses once it knows that its peer has been able to connect to a transport address with higher priority, or one of equal priority if it was peer derived. A client can know that a peer was able to connect to a transport address based on the receipt of a STUN BindingRequest against that transport address. The username and password in the STUN BindingRequest can be used to determine which transport address the STUN request was generated against. Note that the transport address that the STUN request was received on does not say anything about which transport address the peer sent to, and so the username and password are used. Such an address SHOULD be freed no earlier than 3 seconds after receipt of the STUN request. This provides a window of time for the peer to cease using the address and switch to a better one.

This connectivity check makes an important assumption. It assumes that if a STUN request is able to get from A to B, the STUN response will get from B to A (packet losses aside). To our knowledge this is generally true, since it is one of the definining characteristics of the client-server protocols that NATs have been designed to pass. We need to make this assumption in order to free up resources and also eliminate additional ICE cycles.

The drawback of this restriction is that if connectivity should be lost during the session, the client cannot fall back to lower priority address. We believe that it is more important to free unneeded resources than to hold onto them in case of the unlikely event of a problem.

For those successful STUN transactions, the responder compares the MAPPED-ADDRESS attribute in the response to the local transport address from which the STUN request was sent. If the two differ, the responder considers the MAPPED-ADDRESS as another transport address that has been gathered for usage in this session. This transport address is referred to as a peer derived transport address. The preference of this transport address is set to the value of the preference of that alternate from the initiate message. For example, if the initiator provides a transport address obtained from a local interface, it might set the preference to 1.0. If the responder sends a STUN request to the server and obtains a new transport address, that transport address is assigned a preference of 1.0. That preference will be used in comparison to other addresses gathered by the responder.

If any STUN BindingRequest results in a BindingErrorResponse, the ERROR-CODE is examined. If it is 401, 430, 432 or 500, the client SHOULD retry the request, applying any appropriate fixes specified by the error code. In the case of 400, 431 and 600, the client MUST NOT retry. This case is treated identically to a timeout, so that it is equal to no connectivity at all.

5.6 Generating the Accept Message

At some point, the responder will decide to accept or reject the communications. A rejection terminates ICE processing, of course. In the case of acceptance, the accept message is constructed as follows.

At the time when the accept message is to be sent, the responder will have gathered some number of transport addresses. Some of these will be local transport addresses, some will be unilaterally derived addresses, and some will be stun derived from the peer in the dialog. Each of these will have a preference, based on either the rules in Section 5.1 or Section 5.5.

Constructing the accept proceeds identically to the way in which the initiate message is constructed (Section 5.4). The transport address with the highest probability of success is placed into the default element. All of the alternatives (including the one placed into the default) are placed into an alternate element. Each is assigned a unique ID. For alternates that were peer-dervived STUN addresses, the derived element is present, and it contains the id of the initiator's

alternate it was derived from. Each alternate contains its preference as computed above. Each alternate contains a username and password that must be used to contact the STUN server listening on that address. Each address SHOULD have a different username and password.

The accept is then sent.

5.7 Initiator Processing of the Accept

There are two possible cases for processing of the Accept message. If the recipient of the Initiate message did not support ICE, the Accept message will only contain the default address information. As a result, the initiator knows that it cannot perform its connectivity checks. In this case, it SHOULD just sent to the transport address listed. However, if local configuration information tells the initiator to try connectivity checks by sending them through the TURN server, this means that packets sent directly to responder may be dropped by a local firewall. To deal with this, the initiator SHOULD allocate, using TURN, a new TURN derived transport address. It does not advertise this address anywhere. Rather, it issues a SEND command using this new transport address. The SEND command contains the media packet to send to the responder. Once this command has been accepted, the initiator SHOULD send all media packets to the TURN server, which will then forward them towards the responder.

If the Accept message contains alternates, the processing of the accept by the initiator is nearly identical to that of the responder processing the initiate message. Specifically, the initiator will send STUN requests to the STUN servers listed in the accept. This results in the same connectivity processing, and will also result in the gathering of new STUN derived addresses. The initiator can begin sending media to the responder immediately using the address in the default. Once STUN has verified connectivity of higher priority addresses, media is sent to those addresses instead. When a client sends media to an alternate with higher priority, if that alternate contained the derived element, the client MUST send media from the local transport address with the id contained in the derived element.

5.8 Additional ICE Cycles

After the completion of the initiate/accept exchange, both sides may continue to obtain more derived transport addresses. This may occur because a STUN transaction took too long to complete, and thus missed the "window" of the previous initiate message/accept exchange. Or, it may occur because the previous initiate/accept exchange provided additional addresses which resulted in new STUN derived attributes.

At any point when either client has one or more new gathered

addresses, it MAY initiate a modify message, and therefore a new corresponding ICE cycle. This modify message is identical to the initiate or accept message generated previously by that client. However, it would include any additional alternates learned since the last message was sent. However, if the preference of those new gathered addresses is lower than the preference for an address that the peer has established connectivity to, the client SHOULD NOT initiate a modify exchange just to convey this address. If an modify exchange is taking place anyway (because a higher priority address is available), the lower qvalue addresses SHOULD be included. A client can determine which addresses a peer has established connectivity to by checking if a STUN request was sent by the peer to that address.

OPEN ISSUE: This optimization doesnt work in cases where the peer establishes connectivity by sending media through its TURN relay, since the resulting priority is less. The client doesnt have any way to know whether or not the connectivity check was made by sending through a relay.

Typically, there won't be more than one or two ICE cycles before convergence. Assuming that there is no network packet loss (which can extend the STUN transaction) and zero network latency, it appears that a maximum of two ICE cycles are needed to reach convergence.

6. Running STUN on Derived Transport Addresses

One of the seemingly bizarre operations done during the ICE processing is the transmission of a STUN request to a transport address which is obtained through TURN or STUN itself. This actually does work, and in fact, has extremely useful properties. The subsections below go through the detailed operations that would occur at each point to demonstrate correctness and the properties derived from it.

6.1 STUN on a TURN Derived Transport Address

Consider a client A that is behind a NAT. It connects to a TURN server on the public side of the NAT. To do that, A binds to a local transport address, say 10.0.1.1:8866, and then sends a TURN request to the TURN server. The NAT translates the net-10 address to 192.0.2.88:5063. Assume that the TURN server is running on 192.0.2.1 and listening for TURN traffic on port 7764. The TURN server allocates a derived transport address 192.0.2.1:26524 to the client, and returns it in the TURN response. Remember that all traffic from the TURN server to the client is sent from 192.0.2.1:7764 to 10.0.1.1:8866.

Now, the client runs a STUN server on 10.0.1.1:8866, and advertises that its server actually runs on 192.0.2.1:26524. Another client, B, sends a STUN request to this server. It sends it from a local transport address, 192.0.2.77:1296. When it arrives at 192.0.2.1:26524, the TURN server "locks down" outgoing traffic, so that data packets received from A are sent to 192.0.2.77:1296. The STUN request is then forwarded to the client, sent with a source address of 192.0.2.1:7764 and a destination address of 192.0.2.88:5063. This passes through the NAT, which rewrites the source address to 10.0.1.1:8866. This arrives at A's STUN server. The server observes the source address of 192.0.2.1:7764, and generates a STUN response containing this value in the MAPPED-ADDRESS attribute. The STUN response is sent with a source address fo 10.0.1.1:8866, and a destination of 192.0.2.1:7764. This arrives at the TURN server, which, because of the lock-down, sends the STUN response with a source address of 192.0.2.1:26524 and destination of 192.0.2.77:1296, which is B's STUN client.

Now, as far as B is concerned, it has obtained a new STUN derived transport address of 192.0.2.1:7764. And indeed, it has! STUN derived transport addresses are scoped to the session, so they can only be used by the peer in the session. Furthermore, that peer has to send requests from the socket on which the STUN server was running. In this case, A is the peer, and its STUN server was on 10.0.1.1:8866. If it sends to 192.0.2.1:7764, the packet goes to the TURN server,

and due to lock-down, is forwarded to B, and specifically, is forwarded to the transport address B sent the STUN request from. Therefore, the address is indeed a valid STUN derived transport address.

The benefit of this is that it allows two clients to share the same TURN server for media traffic in both directions. With "normal" TURN usage, both clients would obtain a derived address from their own TURN servers. The result is that, for a single call, there are two bindings allocated by each side from their respective servers, and all four are used. With ICE, that drops to two bindings allocated from a single server. Of course, all four bindings are allocated initially. However, once one of the clients begins receiving media on its STUN derived address, it can deallocate its TURN resources.

[[TODO: Include a diagram that shows this pictorially.]]

6.2 STUN on a STUN Derived Transport Address

Consider a client A that is behind a NAT. It connects to a STUN server on the public side of the NAT. To do that, A binds to a local transport address, say 10.0.1.1:8866, and then sends a STUN request to the STUN server. The NAT translates the net-10 address to 192.0.2.88:5063. Assume that the STUN server is running on 192.0.2.1 and listening for STUN traffic on port 3478, the default STUN port. The STUN server sees a source IP address of 192.0.2.88:5063, and returns that to the client in the STUN response. The NAT forwards the response to the client.

Now, the client runs a STUN server on 10.0.1.1:8866, and advertises that its server actually runs on 192.0.2.88:5063. Another client, B, sends a STUN request to this address. It sends it from a local transport address, 192.0.2.77:1296. When it arrives at 192.0.2.88:5063 (on the NAT), the NAT rewrites the source address to 10.0.1.1:8866, assuming that it is of the full-cone variety [1], or is restricted, and the permission for 192.0.2.77:1296 is open. This arrives at A's STUN server. The server observes the source address of 192.0.2.77:1296, and generates a STUN response containing this value in the MAPPED-ADDRESS attribute. The STUN response is sent with a source address of 10.0.1.1:8866, and a destination of 192.0.2.77:1296. This arrives at B's STUN client.

Now, as far as B is concerned, it has obtained a new STUN derived transport address of 192.0.2.77:1296. Of course, this is the same address as the local transport address, and therefore this derived address is not used. However, had there been additonal NATs between B and A's NAT, B would end up seeing the binding allocated by that outermost NAT. The net result is that STUN requests sent to a STUN

derived address behave as normal STUN would. However, these STUN requests have the side-effect of creating permissions in the NATs which see those requests in the public to private direction. This turns out to be very useful for traversing restricted NATs.

7. XML Schema for ICE Messages

This section contains the XML schema used to define the initiate, accept, and modify messages. Any protocol that uses ICE needs to map the parameters defined here into its own messages.

Note that STUN allows both the username and password to contain the space character. However, usernames and passwords used with ICE cannot contain the space.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="media-streams">
    <xs:annotation>
      <xs:documentation>This is the root element, which holds a series
      of media stream elements.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="media-stream" minOccurs="0" maxOccurs="unbounded">
          <xs:annotation>
            <xs:documentation>There are zero or more media stream
             elements. Each defines attributes for a specific media
             stream.</xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:sequence>
              <xs:element name="default-address">
                <xs:annotation>
                  <xs:documentation>The default address is used for
                  sending media before connectivity has been
                  verified.</xs:documentation>
                </xs:annotation>
                <xs:complexType>
                  <xs:complexContent>
                    <xs:extension base="rtp-info"/>
                  </xs:complexContent>
                </xs:complexType>
              </xs:element>
              <xs:sequence>
                <xs:element name="alternate" minOccurs="0"
                 maxOccurs="unbounded">
                  <xs:annotation>
                    <xs:documentation>Each alternate is a
                     possible point of contact.</xs:documentation>
                  </xs:annotation>
```

```
                    <xs:complexType>
                      <xs:complexContent>
                        <xs:extension base="transport-data">
                          <xs:attribute name="preference"
                           type="xs:double" use="required"/>
                          <xs:attribute name="id" type="xs:string"
                           use="required"/>
                        </xs:extension>
                      </xs:complexContent>
                    </xs:complexType>
                  </xs:element>
                </xs:sequence>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:complexType name="transport-data">
      <xs:sequence>
        <xs:element name="stun-username" type="xs:string"/>
        <xs:element name="stun-password" type="xs:string"/>
        <xs:element name="derived-from" type="xs:string" minOccurs="0"/>
        <xs:element name="rtp-address" type="transport-address"/>
        <xs:element name="rtcp-address" type="transport-address"/>
      </xs:sequence>
    </xs:complexType>
    <xs:complexType name="transport-address">
      <xs:sequence>
        <xs:element name="ip-address" type="xs:string"/>
        <xs:element name="port">
          <xs:simpleType>
            <xs:restriction base="xs:integer">
              <xs:minInclusive value="1"/>
              <xs:maxInclusive value="65535"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
    <xs:complexType name="rtp-info">
      <xs:sequence>
        <xs:element name="rtp-address" type="transport-address"/>
        <xs:element name="rtcp-address" type="transport-address"/>
      </xs:sequence>
    </xs:complexType>
  </xs:schema>
```

8. Examples

TODO. Fill in with example XML message exchanges.

9. Mapping ICE into SIP

In this section, we show how to map ICE into SIP. This requires extensions to SDP.

A new SDP attribute is defined to support ICE. It is called "alt". The alt attribute MUST be present within a media block of the SDP. It contains an alternative IP address and port (or pair of IP addresses and ports in the case of RTP) that the recipient of the SDP can use instead of the ones indicated in the m and c lines. There MAY be multiple alt attributes in a media block. In that case, each of them MUST contain a different IP address and port (or a differing pair of IP address and ports in the case of RTP).

The syntax of this attribute is:

```
alt-attribute = "alt" ":" id SP qvalue SP derived-from SP
                  username SP password SP
                  unicast-address SP port [unicast-address SP port]
                  ;qvalue from RFC 3261
                  ;unicast-address, port from RFC 2327
username      = non-ws-string
password      = non-ws-string
id            = token
derived-from  = ":" / id
```

With the addition of the alt attribute, the mapping of the ICE messages to SIP/SDP is straightforward. The ICE initiate message corresponds to a SIP message with an SDP offer. The ICE accept message corresponds to a SIP message with a SDP answer. The ICE modify message corresponds to a SIP INVITE or UPDATE with an offer, and the ICE modify accept message corresponds to an INVITE or UPDATE response with an answer.

Each media stream element in an ICE message maps to a media block in the SDP. The default address maps to the m and c lines in the SDP. If the ICE message indicates an RTCP address and port that are not one higher than that of the RTP, the SDP RTCP attribute [2] MUST be used to convey them.

Each alternate element in an ICE message maps either to an alt attribute in the SDP, or a new media block, depending on the IP version of the alternate. For the highest priority IPv6 alternate, it is mapped into a separate media block, using the IPV grouping [3]. Any additional IPv6 addresses are placed as alternates within this media block. For alternates that are IPv4 addresses, the alt attribute is used. The rtp-address element maps to the first

unicast-address and port components of the alt attribute. The rtcp-address element maps to the second unicast-address and port components of the alt attribute. Note that, if the RTCP address is identical to the RTP address, and the port is one higher, the second unicast-address and port MAY be omitted. The preference value from the alternate element is mapped to the q-value component of the alt attribute. The STUN username and password elements map to the username and password components of the alt attribute. When the derived element is present in the ICE message, it is represented in the derived-from component of the alt attribute. If it is not present in the ICE message, the derived-from component of the alt attribute has a value of ":".

10. Security Considerations

ICE conveys the STUN username and password within its messages. If an eavesdropper should see the username and password, the worst they can do is send STUN requests to the host. Since STUN is a stateless protocol, the attacker can not alter the processing of the call or otherwise disrupt it. They could flood the server with BindingRequest packets. However, this would be no worse than if the attacker simply floods the host with any kind of packet.

However, integrity protection of the username and password are more important. If an attacker is capable of intercepting the message and modifying the username or password, they could prevent connectivity from being established between peers, and therefore disrupt the call. Of course, if the attacker can intercept the message, there are many other ways in which they could do that, such as simply discarding the message. Injecting fake messages with incorect usernames and passwords can also disrupt a call, and does not require the compromise of an intermediate server. A similar attack is possible by modifying most of the ICE message attributes. To prevent these kinds of attacks, it is RECOMMENDED that the actual protocols the ICE maps to make use of security mechanisms that provide message integrity protection.

11. IANA Considerations

This specification defines one new media attribute: alt. Its syntax is defined in Section 9.

12. IAB Considerations

The IAB has studied the problem of "Unilateral Self Address Fixing", which is the general process by which a client attempts to determine its address in another realm on the other side of a NAT through a collaborative protocol reflection mechanism [11]. ICE is an example of a protocol that performs this type of function. Interestingly, the process for ICE is not unilateral, but bilateral, and the difference has a signficant impact on the issues raised by IAB. The IAB has mandated that any protocols developed for this purpose document a specific set of considerations. This section meets those requirements.

12.1 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 client to determine an address that is reachable by another peer with which it wishes to communicate.

12.2 Exit Strategy

From RFC 3424, any UNSAF proposal must provide:

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

ICE itself doesn't easily get phased out. However, it is useful even in a globally connected Internet, to serve as a means for detecting whether a router failure has temporarily disrupted connectivity, for example. However, what ICE does is help phase out other UNSAF mechanisms. ICE effectively selects amongst those mechanisms,

prioritizing ones that are better, and deprioritizing ones that are worse. Local IPv6 addresses are always the most preferred. As NATs begin to dissipate as IPv6 is introduced, derived transport addresses from other UNSAF mechanisms simply never get used, because higher priority connectivity exists. Therefore, the servers get used less and less, and can eventually be remove when their usage goes to zero.

Indeed, ICE can assist in the transition from IPv4 to IPv6. It can be used to determine whether to use IPv6 or IPv4 when two dual-stack hosts communicate with SIP (IPv6 gets used). It can also allow a client in a v6 island to communicate with a v4 host on the other side of a 6to4 NAT, by allowing the v6 host to address-fix against the v4 host, and in the process, obtain a v4 address which can be handed to the v4 client.

12.3 Brittleness Introduced by ICE

From RFC3424, any UNSAF proposal must provide:

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

ICE actually removes brittleness from existing UNSAF mechanisms. In particular, traditional STUN (the usage described in RFC 3489) has several points of brittleness. One of them is the discovery process which requires a client to try and classify the type of NAT it is behind. This process is error-prone. With ICE, that discovery process is simply not used. Rather than unilaterally assessing the validity of the address, its validity is dynamically determined by measuring connectivity to a peer. The process of determining connectivity is very robust. The only potential problem is that bilaterally fixed addresses through STUN can expire if traffic does not keep them alive. However, that is substantially less brittleness than the STUN discovery mechanisms.

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

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

provided a usable address.

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

Traditional STUN also introduces some security considerations. Unfortunately, since ICE still uses network resident STUN servers, those security considerations still exist.

12.4 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.

12.5 Issues with Existing NAPT Boxes

From RFC 3424, any UNSAF proposal must provide:

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

A number of NAT boxes are now being deployed into the market which try and provide "generic" ALG functionality. These generic ALGs hunt for IP addresses, either in text or binary form within a packet, and rewrite them if they match a binding. This will interfere with proper operation of any UNSAF mechanism, including ICE.

13. Acknowledgements

The authors would like to thank Douglas Otis and Francois Audet for their comments and input.

Normative References

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

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

[3] Camarillo, G. and J. Rosenberg, "The Alternative Semantics for the Session Description Protocol Grouping Framework", draft-camarillo-mmusic-alt-01 (work in progress), June 2003.

Informative References

[4] 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.

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

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

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

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

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

[10] Yon, D., "Connection-Oriented Media Transport in SDP", draft-ietf-mmusic-sdp-comedia-05 (work in progress), March 2003.

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

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

[13] Huitema, C., "Teredo: Tunneling IPv6 over UDP through NATs", draft-ietf-ngtrans-shipworm-08 (work in progress), September 2002.

[14] Rosenberg, J., "Traversal Using Relay NAT (TURN)", draft-rosenberg-midcom-turn-02 (work in progress), October 2003.

Author's Address

Jonathan Rosenberg
dynamicsoft
600 Lanidex Plaza
Parsippany, NJ 07054
US

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

Intellectual Property Statement

Full Copyright Statement

Acknowledgement

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