MMUSIC
Internet-Draft
Expires: April 20, 2006

J. Rosenberg
Cisco Systems

October 17, 2005

# TCP Alternatives with Interactive Connectivity Establishment (ICE draft-rosenberg-mmusic-ice-tcp-00

## Status of this Memo



## Copyright Notice



## Abstract

Interactive Connectivity Establishment (ICE) defines a mechanism for NAT traversal for multimedia communication protocols based on the offer/answer model of session negotiation. ICE works by providing a set of candidate transport addresses for each media stream, which are then validated with peer-to-peer connectivity checks based on Simple Traversal of UDP over NAT (STUN). ICE provides a general framework for describing alternates, but only defines UDP-based transport protocols. This specification extends ICE to TCP-based media, including the ability to offer a mix of TCP and UDP-based candidates

for a single stream.

Table of Contents

## 1. Introduction

Interactive Connectivity Establishment (ICE) [5] defines a mechanism for NAT traversal for multimedia communication protocols based on the offer/answer model [2] of session negotiation. ICE works by providing a set of candidate transport addresses for each media stream, which are then validated with peer-to-peer connectivity checks based on Simple Traversal of UDP over NAT (STUN) [1]. ICE provides a general framework for describing alternates, but only defines procedures for UDP-based transport protocols.

There are many reasons why ICE support for TCP is important. Firstly, there are media protocols that run over TCP. Examples of such protocols are web and application sharing and instant messaging [7]. For these protocols to work in the presence of NAT, unless they define their own nat traversal mechanisms, ICE support for TCP is needed. In addition, RTP itself can run over TCP [8]. Typically, it is preferable to run RTP over UDP, and not TCP. However, in a variety of network environments, overly restrictive NAT and firewall devices prevent UDP-based communications altogether, but general TCP-based communications are permitted. In such environments, sending RTP over TCP, and thus establishing the media session, may be preferable to having it fail altogether. With ICE, agents can gather both UDP and TCP candidates for an RTP-based stream, list the UDP ones with higher priority, and then only use the TCP-based ones if the UDP ones fail altogether. This provides a fallback mechanism that allows multimedia communicatoins to be highly reliable.

The usage of RTP over TCP is particularly useful when combined with TURN. In that usage, one of the agents would connect to its TURN server using TCP, and obtain a TCP-based transport address. It would offer this up to its peer agent as a candidate. That agent would initiate a TCP connection towards the TURN server. When that connection is established, media can flow over the connections, through the relay. The benefit of this usage is that it only requires the agents to make outbound TCP connections to a server on the public network. This kind of operation is broadly interoperable through NAT and firewall devices. Since it is a goal of ICE and this extension to provide highly reliable communications that "just works" in as a broad a set of network deployments as possible, this usage is particularly important.

This specification extends ICE by defining its usage with TCP-based candidates. ICE indicates in each of its sections where there is transport-specific logic. It requests that specifications which define usage of ICE with other transport protocols - as this one does - define a version of that logic. This specification does so by following the outline of ICE itself, and calling out the transport

protocol specific logic needed in each section.

## 2. Overview of Operation

The usage of ICE with TCP is relatively straightforward. The main area of specification is around how and when connections are opened, and how those connections relate to transport address pairs and candidates.

When the agents perform address allocations to gather TCP-based candidates, the transport addresses they obtain are always used in a passive mode. As such, a candidate pair formed through an offer/ answer exchange will contain a pair of transport addresses, both of which can only be used in passive mode. When it comes time to perform a connectivity check on the candidate pair, both sides open a TCP connection, but do so from an ephemeral port on the same interface as their passive transport address. If the connection setup succeeds, the active side sends a STUN Binding Request over the connection. With TCP, the STUN Binding Requests are not so much for validation of connectivity (which TCP itself will provide), but rather, identification of the connection and correlation of it with a peer agent.

Since the connection was opened from a different port, the agent will see a new source IP address and port in the Binding Request. This will result in the creation of a new peer-derived TCP candidate and a candidate pair. If the connection attempt succeeded in the other direction, a second peer-derived TCP candidate and candidate pair would be created. The peer-derived candidate pairs each hold a single TCP connection, whereas the original candidate pair will never hold any. Effectively, the original candidate pair only serves the purpose of spawning peer-derived candidate pairs that actually contain the TCP connections.

This is shown pictorially in Figure 1. The picture shows two agents L and R. Agent L has an IP interface with IP address M, and agent R has one with IP address N. Agent L binds to a TCP port on interface M with port X, and Agent R binds to a TCP port on interface N with port A. An offer answer exchange takes place, resulting in a candidate pair with a transport address pair containing M:X and N:A. When this candidate pair is selected for a connectivity check, agent L initiates the connection on interface M, but from ephemeral port Y. The connection is opened to N:A. Similarly, agent R opens a connection from interface N, but from ephemeral port B. The connection is opened to M:X. Each agent sends a STUN Binding Request over the connection it opened. This will result in a pair of peer derived candidates, each with a transport address pair and a TCP connection. One contains the transport address pair {N:B,M:X} and

the connection with that 5-tuple, and the other contains the transport address pair {M:Y,N:A} and the connection with that 5-tuple. Note how there is no TCP connection associated with the original candidate pair.

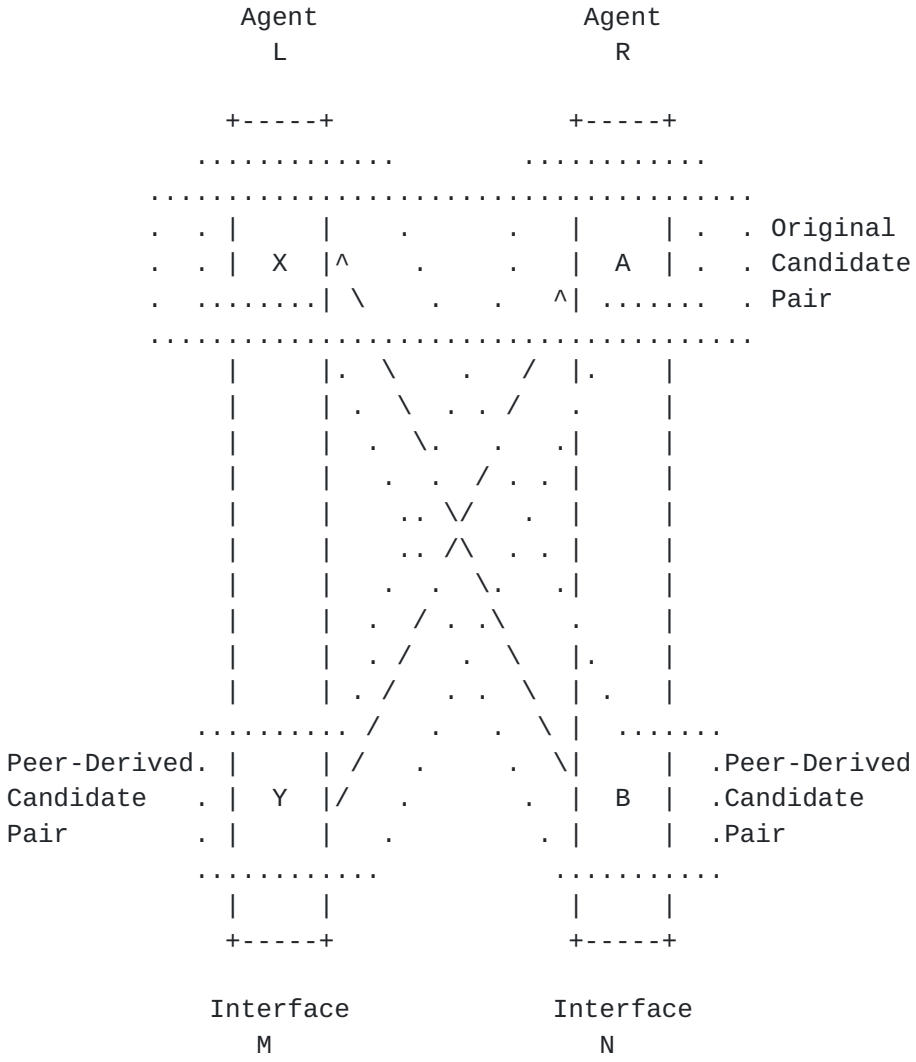


Figure 1

Because of the reliable nature of TCP, a Binding Request is needed only from the active side of the connection to the passive side, entirely for identification purposes. Once complete, the peer-derived candidate pair and its connection are valid, and can be promoted to the m/c-line.

The reason why connections are not opened from the same port as the transport address is that doing so would produce a simultaneous open between the agents in many cases. True simultaneous opens fail to

work through many NATs, therefore defeating the purpose of this specification.

When a TCP-based candidate is promoted to the m/c-line, the SDP extensions for connection oriented media [3] are used to signal that an existing connection should be used, rather than opening a new one. In addition, the original candidate is no longer listed with a=candidate attributes. This is to prevent usage of STUN for keepalives. Separating STUN from the media data over the same TCP connection may not be possible, and for this reason application-layer keepalives are used with TCP.

## 3. Gathering Addresses

Section 7.1 of ICE defines the procedures for gathering of transport addresses for usage in candidates. These procedures are defined for local candidates, STUN-derived candidates and TURN-derived candidates. ICE indicates that these procedures are transport protocol specific, and requires extensions to ICE to define procedures for other transport protocols. This section defines those procedures for TCP.

For each TCP-only media stream the agent wishes to use, the agent obtains a set of candidates by binding to N ephemeral TCP ports on each interface, where N is the number of transport addresses needed for the candidate. For media streams that can support either UDP or TCP, the agent SHOULD obtain a set of candidates by binding to N ephemeral UDP and N ephemeral TCP ports on each interface, where N is the number of transport addresses needed for the candidate.

Media streams carried using the Real Time Transport Protocol (RTP) [4] can run over TCP [8]. As such, it is RECOMMENDED that both UDP and TCP candidates be obtained. However, providers of real-time communications services may decide that it is preferable to have no media at all than it is to have media over TCP. To allow for choice, it is RECOMMENDED that agents be configurable with whether they obtain TCP candidates for real time media.

> Having it be configurable, and then configuring it to be off, is far better than not having the capability at all. An important goal of this specification is to provide a single mechanism that can be used across all types of endpoints. As such, it is preferable to account for provider and network variation through configuration, instead of hard-coded limitations in an implementation. Furthermore, network characteristics and connectivity assumptions can, and will change over time. Just because a agent is communicating with a server on the public network today, doesn't mean that it won't need to communicate with

one behind a NAT tomorrow. Just because a agent is behind a full cone NAT today, doesn't mean that tomorrow they won't pick up their agent and take it to a public network access point where there is a symmetric NAT or one that only allows outbound TCP. The way to handle these cases and build a reliable system is for agents to implement a diverse set of techniques for allocating addresses, so that at least one of them is almost certainly going to work in any situation. Implementors should consider very carefully any assumptions that they make about deployments before electing not to implement one of the mechanisms for address allocation. In particular, implementors should consider whether the elements in the system may be mobile, and connect through different networks with different connectivity. They should also consider whether endpoints which are under their control, in terms of location and network connectivity, would always be under their control. Only in cases where there isn't now, and never will be, endpoint mobility or nomadicity of any sort, should a technique be omitted.

STUN-based candidates for TCP streams are not possible, since STUN only works with UDP.

To obtain a TURN-derived TCP candidates, the client takes a local TCP candidate, and for each configured TURN server, produces a TCP TURN candidate. It is anticipated that clients may have a multiplicity of TURN servers configured in network environments where there are multiple layers of NAT, and that layering is known to the provider of the client. To produce the TURN candidate from a local candidate, it iterates through the local transport addresses in the local candidate, and for for each one, initiates a TCP connection from the same interface of the local transport address to the TURN server. It MUST NOT initiate the connection from the actual port in the local transport address, but rather, from an ephemeral port. Following the procedures of Section 8 of [6], it initiates an Allocate Request transaction over the connection. The Allocate Response will provide the client with its TCP TURN derived transport address in the MAPPED-ADDRESS attribute. Once the TURN allocations against a particular TURN server succeed from all of the transport addresses in a particular local candidate, the client SHOULD NOT attempt any further TURN allocations to that particular server from the transport addresses in any other local candidates.

Like its UDP counterparts, TCP-based TURN allocations are paced out at one every Ta seconds. This pacing refers to the establishment of a TCP connection to the server and the subsequent TURN request. That is, every Ta seconds, the agent will open a new TCP connection and send a TURN Allocate request.

## 4. Prioritization

Section 7.2 of ICE defines guidelines for prioritizing the set of candidates learned through the gathering process. It specifies that if there are considerations that are specific to the transport protocol, these considerations should be called out in the ICE extension which defines usage with that transport protocol. This section describes considerations specific to TCP.

The transport protocol itself is a criteria for choosing one candidate over another. If a particular media stream can run over UDP or TCP, the UDP candidates might be preferred over the TCP candidates. This allows ICE to use the lower latency UDP connectivity if it exists, but fallback to TCP if UDP doesn't work.

Section 7.2 of ICE also defines guidelines for selecting an active candidate in the initial offer. It specifies that if there are considerations that are specific to the transport protocol, these considerations should be called out in the ICE extension which defines usage with that transport protocol. This section describes considerations specific to TCP.

When TCP-based media streams are used with ICE, the ICE mechanisms described here are responsible for opening the connections and testing them. Once validated, they are promoted to active and then, and only then, can be used for media transport. For this reason, in an initial offer, prior to validation, the active candidate will either be non-TCP (for example, with RTP, it is anticipated that the active candidate would be UDP-based, with TCP candidates as lower priority alternatives), or there is no active candidate.

## 5. Encoding

Section 7.3 of ICE defines procedurs for encoding the candidates into an SDP offer or answer. It specifies that if there are considerations that are specific to the transport protocol, these considerations should be called out in the ICE extension which defines usage with that transport protocol. This section describes considerations specific to TCP.

TCP-based candidates are encoded into a=candidate lines identically to the UDP encoding described in [5]. However, the transport protocol is set to "tcp" rather than "udp".

Encoding of the active candidate in the m/c-line, however, requires special considerations for TCP. If there is no active candidate, the media session MUST include an a=holdconn attribute as defined in RFC 4145 [3]. This has the effect of suspending opening of the TCP

connections - exactly the desired effect since they are opened by the procedures defined in this specification. The IP address and port encoded into the m/c-line are inconsequential, since they are never used anyway.

If there is an active candidate, it will be because a candidate pair has been validated. The m/c-line contains the native IP address and port for the candidate, which will be the ephemeral port if the agent had opened the connection. This is in contrast to RFC 4145, which recommends that the active side of a connection place a port with value '9'. In addition, the media session MUST NOT contain the a=holdconn attribute. It MUST contain the a=active attribute if the agent had opened the TCP connection corresponding to the active candidate, and a=passive if it had been the passive side of the connection. Finally, the media session MUST contain the a=existing attribute, indicating that an existing connection is to be used, rather than opening a new one.

## 6. Ordering the Candidate Pairs

Section 7.5 of ICE defines procedurs for ordering the candidates into an SDP offer or answer. It specifies that if there are considerations that are specific to the transport protocol, these considerations should be called out in the ICE extension which defines usage with that transport protocol. This section describes considerations specific to TCP.

ICE defines two orderings for candidate pairs - a priority order and a check order. These differ only by the position of the active candidate in the list. However, with TCP, prior to validation, there is no active TCP candidate. As a consequence, the two lists are equivalent if there is no active candidate.

## 7. Performing the Connectivity Checks

Section 7.6 of ICE defines procedures for performing the connectivity checks. These are based on a state machine that captures progressions of the checks. This state machine is specific to the transport protocol, and the version for TCP is described here.

The set of states visited by the offerer and answerer are depicted graphically in Figure 2

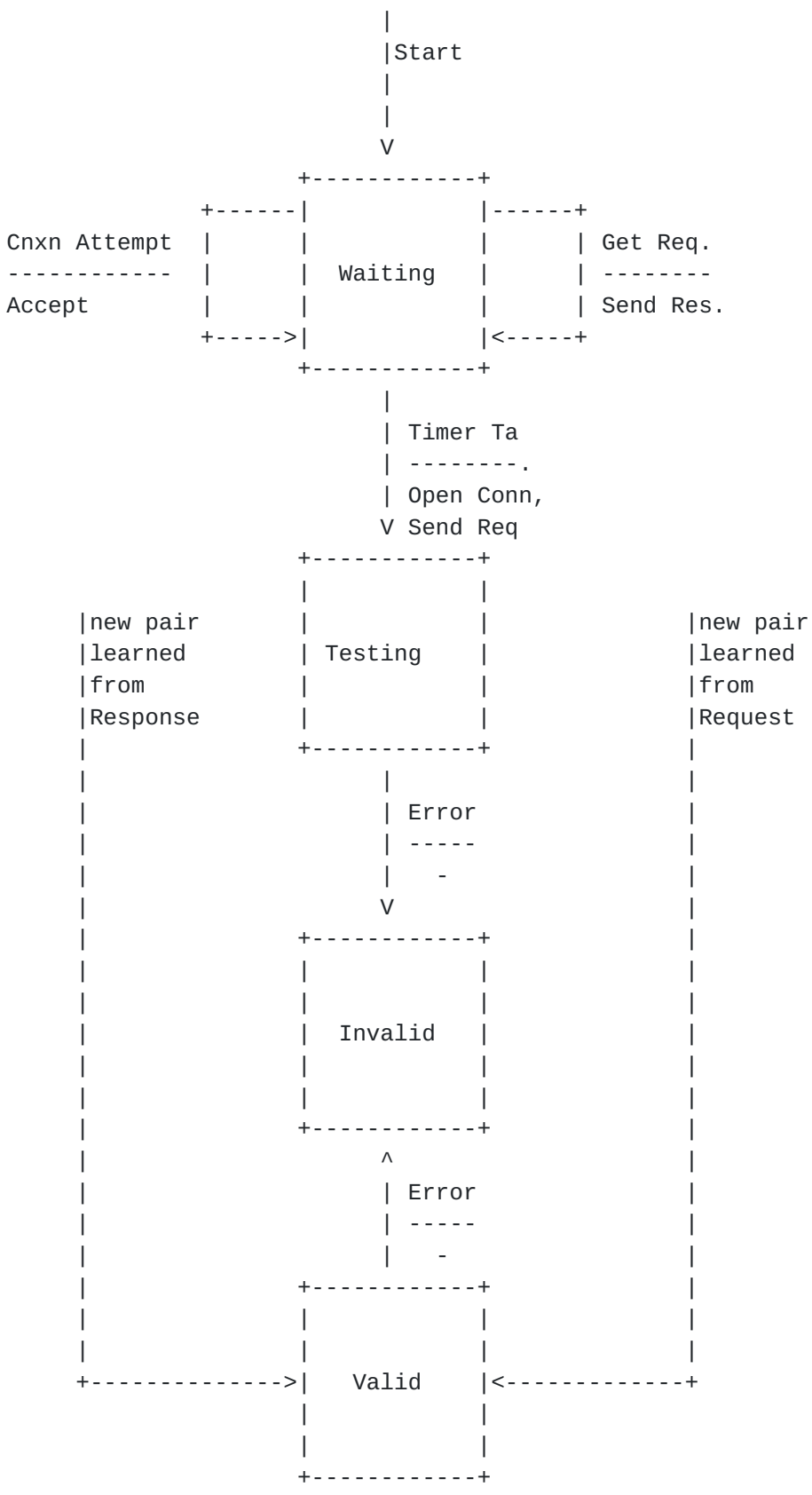
Start
Cnxn Attempt
Accept
Waiting
Get Req.
Send Res.
Timer Ta
Open Conn,
Send Req
new pair
learned
from
Response
Testing
new pair
learned
from
Request
Error
Invalid
Error
Valid

Figure 2

The state machine has four states - waiting, testing, Valid and Invalid. Initially, all transport address pairs start in the waiting state. In this state, the agent waits for one of a chance to open a connection and send a Binding Request.

STUN Binding Requests and Responses are mapped to transport address pairs and their state machines as described in Section 7.6 of ICE.

Every Ta seconds, the agent starts a new connectivity check for a transport address pair. The check is started for the first transport address pair in the transport address pair check ordered list that is in the Waiting state. The state machine for this transport address pair is moved to the Testing state, and the agent opens a TCP connection to the remote transport address in the transport address pair, and do so "from" its native transport address. Here, "from" means something different than the UDP case. If the native transport address is a local transport address, the agent opens the TCP connection from the same IP interface used to obtain the local transport address, but from a different and ephemeral port. Indeed, that port MUST NOT be the same as the port in the local transport address. If the native transport address is a TURN-derived TCP transport address, no attempt is made to open a connection at all. TURN-derived TCP transport addresses can only be used in passive mode.

Once the connection is opened, the agent sends a STUN Binding Request according to the procedures of Section 7.7 of ICE. That section indicates that STUN extensions should define any transport specific considerations for transmission of the STUN request. In the case of TCP, the STUN request is sent on the connection that was just opened. The STUN request is not retransmitted. STUN messages include length indicators, allowing them to be framed over a connection-oriented transport protocol.

If, while in the Waiting state, the agent receives a connection setup attempt on one of its candidates, it creates the connection. If it receives a STUN Binding Request, it generates a response according to the procedures in Section 7.8 of ICE, including generation of the MAPPED-ADDRESS attribute in the response. Note that, in the case of TCP, there is no need to disambiguate STUN and media traffic sent over the same connection. When a connection is opened initially, the first packet sent (and received) is a stun message. No further STUN messages are sent; the connection is either eventually torn down, or promoted to active, in which case media packets will follow.

If the STUN transaction produces an error, the state machine moves

into the Invalid state. Note that there is no change of state if it produces a success response. Rather, that response will yield a new peer-derived transport address and corresponding state machine that moves directly to the Valid state, as described below. Similarly, if an agent receives a STUN request that generates a success response, a peer derived transport address is created, and its corresponding transport address pair moves to the Valid state.

## 8. Promoting a Candidate to Active

Promotion of a candidate to active occurs as described in Section 7.9 of ICE. The only difference to note is that, with TCP, the candidate pair priority ordered list and candidate pair check ordered list are identical, since there is no active TCP candidate. As a consequence, as soon as a candidate is validated, if it is the first in the priority list, an offer is sent immediately. Otherwise, timer Tws is set, and the offer will be sent when it fires.

## 9. Learning New Candidates from Connectivity Checks

Section 7.10 of ICE describes procedures for learning new candidates from connectivity checks. ICE indicates that the behavior of the state machines are transport protocol specific, and extensions to ICE for new transport protocols are asked to describe the behavior of the state machines. This section does so for TCP.

Firstly, it is important to realize that a successul TCP connection attempt and STUN connectivity check will always result in a peer-derived candidate being constructed. ICE talks about learning new peer-derived candidates as a consequence of symmetric NAT. Here, they are learned as a consequence of opening TCP connections from an ephemeral port.

When a new peer-derived candidate is formed as a result of receipt of a STUN Binding Request that generates a successful response, the state machine for that candidate enters the Valid state. Unlike UDP, a Binding Request is not sent back to the source of the request. Similarly, when a new peer-derived candidate is formed as a result of receipt of a successful STUN Binding Response, the state machine for that candidate enters the Valid state. In both cases, the new candidate pair is inserted into the ordered list of pairs and processing follows the logic described in Section 7.

## 10. Subsequent Offers

Section 7.11 of ICE describes procedures for subsequent offer/answer exchanges. ICE indicates that if there are any considerations that are transport protocol specific, new transport protocols are asked to

describe them. This section does so for TCP.

The procedures defined in Section 7.11 of ICE apply to TCP as defined. However, if a candidate is not valid, it MUST NOT be placed into the m/c-line of a subsequent offer or answer. Only valid candidates are placed into the m/c-line for TCP. This is in contrast to UDP, where a partially valid one can be used.

Once the offer/answer exchange has completed, the m/c-lines from each agent, when put together, identify a complete 5-tuple. This 5-tuple is used to identify the TCP connection on which media can now be sent.

In addition, if a candidate pair is removed as a consequence of the processing defined in Section 7.11, and that candidate pair was TCP-based, its corresponding TCP connection (if any) is torn down.

Additional considerations do apply, however, to the usage of RFC 4145 attributes in the m/c-line. The offerer will include the a=existing and either a=active or a=inactive attributes in the m-line, depending on whether the agent had opened or closed the connection. When the answerer receives this, it follows the procedures of RFC 4145 to generate the attributes in the response. It MUST indicate that the existing connection is being reused, by including an a=existing attribute in the answer.

Furthermore, RFC 4145 defines the a=existing attribute to mean the reuse of the existing connection established as a consequence of RFC 4145 processing for this media stream. This specification broadens that definition. The existing connection can also be one established as a consequence of the mechanisms defined in this specification, and the specific TCP connection to use is defined by the 5-tuple constructed from the m/c-line in the offer and the m/c-line in the answer.

RFC 4145 also describes TCP connection lifecycle management procedures. If the TCP connection used in the m/c-line was opened by ICE processing, it is closed by ICE processing as well. This occurs when the session terminates, or when the generating candidate for the active one ceases to be retained in a subsequent offer/answer exchange.

## 11. Binding Keepalives

As mention in ICE, STUN-based keepalives are not used for TCP-based media streams. Instead, application layer keepalives MUST be used. For RTP, the considerations described in Section 7.12 of ICE for communicating with non-ICE endpoints apply to the selection of a

keepalive mechanism.

## 12. Security Considerations

## 13. IANA Considerations

There are no IANA considerations associated with this specification.

## 14. References

### 14.1 Normative References

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

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

[3] Yon, D. and G. Camarillo, "TCP-Based Media Transport in the Session Description Protocol (SDP)", RFC 4145, September 2005.

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

[5] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Methodology for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", draft-ietf-mmusic-ice-05 (work in progress), July 2005.

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

### 14.2 Informative References

[7] Campbell, B., "The Message Session Relay Protocol", draft-ietf-simple-message-sessions-11 (work in progress), July 2005.

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

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.