MMUSIC
Internet-Draft
Expires: December 28, 2006

J. Rosenberg
Cisco Systems

June 26, 2006

TCP Candidates with Interactive Connectivity Establishment (ICE draft-ietf-mmusic-ice-tcp-01

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) [6] 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 only run over TCP. Examples of such protocols are web and application sharing and instant messaging [9]. 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 [5]. 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 communications to be highly reliable.

The usage of RTP over TCP is particularly useful when combined with the STUN relay usage [7]. In that usage, one of the agents would connect to its STUN relay server using TCP, and obtain a TCP-based allocated address. It would offer this to its peer agent as a candidate. The answerer would initiate a TCP connection towards the STUN relay 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, three types of candidates can be obtained. These are active candidates, passive candidates, and actpass candidates [3]. An active candidate is one for which the agent will attempt to open an outbound connection, but will not receive incoming connection requests. A passive candidate is one for which the agent will receive incoming connection attempts, but not attempt a connection. An actpass candidate is one for which the agent will do both.

Not all types of candidates can be obtained from all types of transport addresses. With local interfaces, agents obtain both actpass and active candidates. Agents don't bother with passive ones, since that functionality is subsumed by the actpass candidate. Server reflexive candidates, by their nature, are always passive. Relayed transport addresses, like local candidates, can produce both actpass and active candidates.

When encoding these candidates into offers and answers, the type of the candidate is signaled. In the case of active candidates, an IP address and port is present, but it is meaningless, as it is ignored by the peer. As a consequence, active candidates do not need to be physically allocated at the time of address gathering. Rather, the physical allocations, which occur as a consequence of a connection attempt, occur at the time of the connectivity checks.

When the candidates are paired together, active candidates are not paired with active, and passive are not paired with passive. When a connectivity check is to be made for a transport address pair within a candidate pair, each agent determines whether it is to make a connection attempt for this pair. If the local candidate is either active or actpass, and the remote is either passive or actpass, it will make the attempt. This means that, for candidate pairs where both candidates are actpass, both agents will attempt to open a TCP connection (this is the so-called simultaneous open in TCP). In the other cases, only one side will try.

Why have both active and actpass candidates for local and relayed transport addresses? Why not just actpass? The reason is that NAT treatment of simultaneous opens is currently not well defined, though specifications are being developed to address this [8]. Some NATs generate block the second TCP SYN packet or improperly process the subsequent SYNACK, which will cause the connection attempt to fail. Therefore, if only simultaneous opens are used, connections may often fail. However, only doing unidirectional opens (where one side is active and the other is passive) is more reliable, but will always require a relay if both sides are behind NAT. Therefore, in the spirit of the ICE philosophy, both are tried. Actpass to actpass are preferred since, if it does work, it will not require a relay even when both sides are behind a same NAT.

Once a connection attempt succeeds, the agent which initiated the connection sends a STUN Binding Request over the connection, and the other agent generates a response. For simultaneous opens, it is possible that both sides will send a Binding Request. The binding request will serve the purpose of correlating the connection to a candidate pair. For candidate pairs where one side was active, the STUN Binding Request will always generate a peer derived candidate and corresponding candidate pair, which is placed immediately in the

Valid state, avoiding the need for additional connectivity checks and computations of new usernames. This derived candidate that is then associated with the TCP connection. For all other candidate pairs, peer derived candidates are not computed (even when the transport address is a new one), and the candidate pair identified by the STUN Binding Request is directly linked to the connection. It is actually possible that a single connection can be associated with multiple candidate pairs; this happens in several situations, and in particular, with connection attempts made to passive candidates. However, a single candidate pair is only ever associated with a single TCP connection.

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. The candidate (or the one which generated it, in the case of a peer-derived candidate) remains listed in a candidate attribute so that STUN-based keepalives can be used throughout the session. This requires demultiplexing STUN and application traffic on the same TCP connection.

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, server reflexive candidates and relayed 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 actpass candidates by binding to N TCP ports on each local interface (typically ephemeral), 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 actpass candidates by binding to N UDP and N TCP ports on each interface, where N is the number of transport addresses needed for the candidate.

Each agent SHOULD also "obtain" an active local candidate for each

local interface, each consisting of N transport addresses. It is not necessary to actually allocate active TCP candidates. These candidates will be signaled in the offer or answer, but they do not include any address and port information - just the STUN usernames and priorities.

Media streams carried using the Real Time Transport Protocol (RTP) [4] can run over TCP [5]. 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 NAT with endpoint indpendent mapping 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 NAT with address and port dependent mapping properties, 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. In environments where mobility and user control are possible, a

multiplicity of techniques is essential for reliability.

Server reflexive candidates are always passive only. They are derived from the STUN Binding Discovery usage or the STUN Relay usage. The latter is preferred since it will provide the client with both a server reflexive and a relayed transport address with a single transaction. It is possible that some STUN servers will only support the Relay usage or only the Binding Discovery usage, in which case a client might be configured with different servers depending on the usage. It is RECOMMENDED that agents obtain server reflexive TCP candidates. In many cases, the agent will not be able to receive incoming TCP connections on a reflexive server address. However, advertising such a transport address through ICE will allow the peer agent to perform a connection attempt through a STUN relay server to that transport address, thereby creating a permission for that IP address on the relay server. This is essential for allowing two clients behind restrictive NATs to rendezvous through the relay.

Relayed candidates can be both actpass and active, and both SHOULD be obtained. As with local candidates, active relayed candidates do not actually need to be allocated at the time of address gathering. Instead, when the agent needs to open a connection from the active relayed candidate, it uses a STUN Allocate request to obtain another allocation on the same interface as its actpass relayed candidate, and then uses the STUN Connect method to open the connection. This is discussed further below.

Obtaining server reflexive passive candidates and relayed actpass candidates for TCP is nearly identical to the UDP case. Like UDP, it can be accomplished with just the relay usage, or with the binding discovery usage and the relay usage separately. The only difference between TCP and UDP is that the client sends its requests to the STUN server by first establishing a TCP connection to the server, and then sending the STUN request over that connection. In addition, the client will request a TCP-based allocation for the relayed address, not a UDP allocation. As in the UDP case, the TCP connection to the STUN server MUST be opened from the local actpass transport address from which it is derived. Detection of duplicate transport addresses is also identical to the UDP case.

Like its UDP counterparts, TCP-based STUN transactions are paced out

at one every Ta seconds. This pacing refers to the establishment of a TCP connection to the server and the subsequent STUN request. That is, every Ta seconds, the agent will open a new TCP connection and send a STUN request, ideally an Allocate request, since it will provide multiple candidates with one 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.

In addition, it is RECOMMENDED that actpass candidates have higher priority than active or passive candidates. As discussed above, this allows for simultaneous opens to be preferred when they work, falling back to unidirectional opens when they do not.

Section 7.2 of ICE also defines guidelines for selecting an operating candidate in the initial 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.

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 operating. Furthermore, like UDP candidate pairs, once validated, a TCP candidate pair can be used immediately in anticipation of an updated offer that promotes the candidate to operating. Due to the time required and overhead of TCP connection establishment, it is RECOMMENDED that there be no operating candidate in the initial offer/answer exchange. This avoids opening a connection for temporary usage, followed by opening of a subsequent higher priority connection that is then used for the remainder of the session.

When media streams supporting mixed modes (both TCP and UDP) are used

with ICE, it is RECOMMENDED that, for real-time streams (such as RTP), the operating candidate be UDP-based.

5. Encoding

Section 7.3 of ICE defines procedures 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 [6]. However, the transport protocol is set to "tcp" for actpass candidates, "tcp-act" for active candidates and "tcp-pass" for passive candidates. The addr and port encoded into the candidate attribute for active candidates MUST be set to IP address that will be used for the attempt, but the port MUST be set to 9 (i.e., Discard). The rules for encoding the candidate type and related transport address are identical to those in [6].

Encoding of the m/c-line, however, requires special considerations for TCP. If the m/c-line is TCP, and there is no operating candidate, the a=holdconn attribute as defined in RFC 4145 [3] MUST be included. 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.

Because this specification recommends that the initial offer and answer make use of an inactive candidate, an operating candidate generally appears there in subsequent offer/answer exchanges, after that candidate has been validated. Indeed, the ICE procedures will actually result in the selection of a candidate pair, which directly maps to a TCP connection. Thus, the purpose of the values in the m/c-line are to identify the TCP connection that will be used, using the candidate pair as the key. The candidate pair is signaled by having the agent include the native IP address and port of that candidate pair in the m/c-line. In the case of a peer-derived candidate pair, the native candidate on the active side will be an ephemeral IP address and port. 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. The media session MUST contain the a=existing attribute, indicating that an existing connection is to be used,

rather than opening a new one. The a=active, a=passive and a=actpass attributes are not relevant when a=existing attribute is present, and SHOULD NOT be included.

6. Ordering the Candidate Pairs

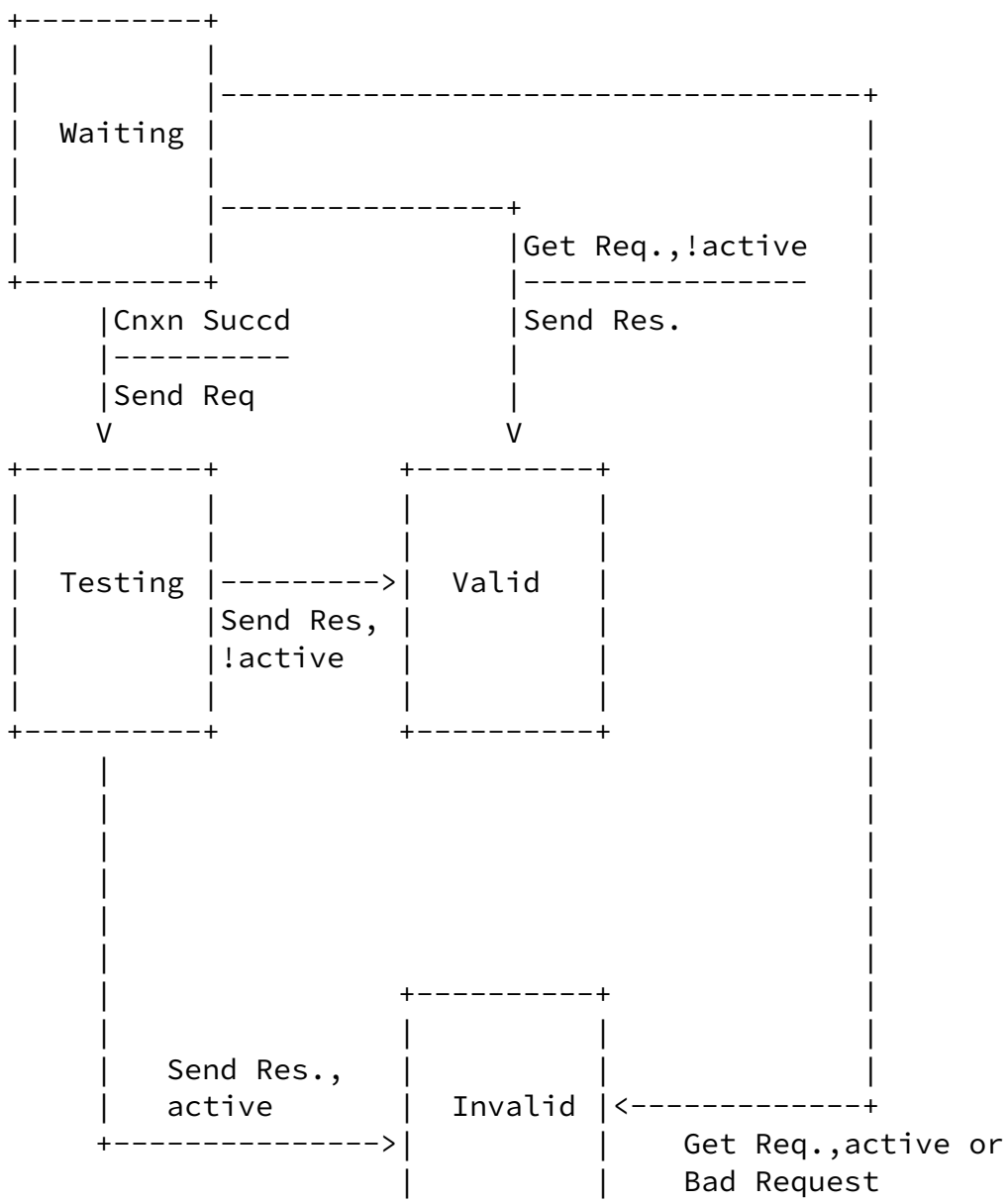
Section 7.5 of ICE defines procedures for ordering the candidate pairs and computing the transport address pair check ordered list. It specifies that if there are considerations that are specific to the transport protocol, these considerations should be called out in the ICE extension which defines usage with that transport protocol. This section describes considerations specific to TCP.

The pruning operation defined in Section 7.5, which removes transport address pairs whose origination transport address matches a previous pair, MUST NOT be used on TCP-based transport address pairs. The reason is that it is redundant with, and interferes with, a similar operation which has agents initiating connections only from active and actpass transport addresses.

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 1


Waiting
Cnxn Succd
Send Req
Get Req.,!active
Send Res.
Testing
Send Res,
!active
Valid
Send Res.,
active
Invalid
Get Req.,active or
Bad Request

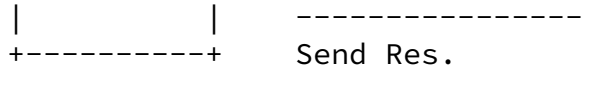


Figure 1

The state machine has four states - Waiting, Testing, Valid and Invalid. Initially, all transport address pairs start in the Waiting state. It is important to understand that the progression of this state machine is driven by the STUN transactions, since it is the STUN requests that identify the candidate pairs. This is distinct from the process of opening and closing connections, which does not directly impact this state machine. First, however, connections need to be opened.

Every Tb seconds, the agent performs a new connection attempt. This attempt is started for first transport address pair in the transport address pair check ordered list that is in the Waiting state and for which the agent is expected to open a connection. An agent is expected to open a connection if its native transport address is either active or actpass, and the remote transport address is either passive or actpass. If the transport address pair meets this criteria, the agent makes a connection attempt.

If the native transport address is active, the agent will use an ephemeral port for the attempt. For a local transport address, the agent initiates an outbound connection from the local interface, towards the remote transport address. The ephemeral port MUST NOT be the same as the port used in an actpass local candidate on the same interface. For an active relayed transport address, the procedure is different. The agent will initiate a new TCP connection to its STUN relay server, from an ephemeral port, but from the same interface as its current connection to that STUN relay server. As with local candidates, this connection to the STUN relay server MUST NOT be from the same port as the current local candidate on the same interface. Once connected, it allocates a TCP transport address. However, it does not need to know its IP address and port. Instead, the agent uses the STUN Connect request, and asks the relay to open a TCP connection towards the remote transport address in the candidate pair.

If the native transport address is actpass, the agent initiates the connection from that transport address. For local transport addresses, this is done by initiating an outbound connection directly from the same IP address and port it is already listening for incoming connection attempts on. For relayed candidates, the agent asks the relay server to initiate a connection from the relayed transport address to the remote transport address. For STUN servers, this is done by issuing a STUN Connect request over the existing connection to the server.

If the connection attempt fails, the agent does nothing. It does not set the state of the transport address pair to Invalid. Indeed, it may still yet be valid if its peer is able to open a connection to the agent. If the connection attempt succeeds, the agent immediately 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. At this point, the state for the corresponding transport address pair

moves from Waiting to Testing.

Furthermore, an agent will be listening for incoming TCP connection establishment requests on each local actpass transport address. For passive reflexive transport addresses, the agent is already listening for incoming requests as a consequence of listening on the local actpass transport address. When an incoming connection request is received, it is accepted, and a TCP connection is set up. However, no attempt is made at this time to change the states of the state machines. Those changes are effected only through STUN requests and responses. For relayed actpass transport addresses, the relay is listening, and will inform the client of progress. In the case of STUN relays, the agent won't actually find out that a connection attempt to the server succeeded. That is not an issue, since the acceptance of connections has no impact on ICE processing. Instead, the agent is informed of data that is ultimately sent over that connection. In the case of ICE, that first data will be a STUN Binding request. It is that request which the client needs to

perform ICE processing.

STUN Binding Requests and Responses are mapped to transport address pairs and their state machines based on the USERNAME, as described in Section 7.6 of ICE. Note, however, that the concepts of a binding request being a match or a miss for a transport address pair, and of matching a binding request to a different transport address pair, do not apply to TCP-based transport address pairs. Rather, the logic described below is followed.

If an agent receives a STUN Binding Request, it generates a response according to the procedures in Section 7.8 of ICE, including generation of the XOR-MAPPED-ADDRESS attribute in the response. If the remote transport address is active, the agent moves this transport address pair into the Invalid state. Furthermore, the agent MUST compute a peer-derived candidate as described in Section 9. In addition, the TCP connection on which the Binding Request was received is then linked with the peer-derived candidate pair.

If the remote transport address is not active, the agent moves this transport address pair into the Valid state. The TCP connection on which the Binding Request was received is then linked with the transport address pair.

If the STUN transaction produces an error, the state machine moves into the Invalid state.

If an agent receives a successful STUN Binding Response, and the native transport address is active, the agent moves this transport

address pair into the Invalid state. Furthermore, the agent MUST compute a peer-derived candidate as described in Section 9. In addition, the TCP connection on which the Binding Request was received is then linked with the peer-derived transport address pair.

If the native transport address is not active, the agent moves this transport address pair into the Valid state. The TCP connection on which the Binding Request was received is then linked with the transport address pair.

8. Promoting a Candidate to Operating

Promotion of a candidate to operating occurs as described in Section 7.9 of ICE. There are no special considerations for TCP.

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 successful TCP connection attempt and STUN connectivity check will always result in a peer-derived candidate being constructed when one transport address was active. ICE talks about learning new peer-derived candidates as a consequence of address and port dependent mapping properties in a NAT. Here, they are learned as a consequence of opening TCP connections from an ephemeral port.

When a new peer-derived transport address is formed as a result of receipt of a STUN Binding Request that generates a successful response, the state machine for that transport address pair 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 transport address pair enters the Valid state. In both cases, the new candidate pair is inserted into the priority 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. In addition, the a=remote-candidate attribute is not used with TCP candidates. An agent SHOULD NOT place one into an offer, and an agent MUST NOT process one if received if in an offer.

Once the offer/answer exchange has completed, the m/c-lines from each agent, when put together, form a set of transport address pairs. These transport address pairs are matched to the transport address pairs across all of the Valid candidate pairs, based on IP address and port comparisons. The highest priority candidate pair amongst the matching ones is selected, and the TCP connections to which it is linked are used, one for each component. It is those TCP connections which will be used for the transport of media. Since there is only ever one TCP connection associated with a transport address pair, and since a single candidate pair is always selected, ICE can guarantee that media is transported between peers over a single TCP connection per component.

It is very important to note that the actual 5-tuple associated with a TCP connection that is used for media might not match the values in the transport address pair.

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 attribute in the m-line. 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 identified by the 5-tuple constructed from the m/c-line in the offer and the m/c-line in the

answer, as described above.

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 operating one ceases to be retained in a subsequent offer/answer exchange.

11. Binding Keepalives

STUN-based keepalives are used for TCP-based media streams, just as they are for UDP-based media streams, and are performed as described in Section 7.12 of ICE. This requires demultiplexing of STUN and application data traffic on the same TCP connection. For media streams based on RTP, this is easily done as follows. The framing mechanism in [5] MUST be used on the TCP connection. In addition, instead of just an RTP or RTCP packet appearing after the LENGTH field, a STUN packet can appear. The agent determines whether the packet is RTP or STUN by looking for the magic cookie in bits 32-63 of the data. If present, it indicates that the packet is STUN, and if not, indicates that it is RTP.

In the case of non-RTP traffic, ICE-TCP can be used with any application protocol which provides some kind of framing into application messages with a well-defined start. When the application framing mechanism points to the start of an application message, the agent looks ahead to bits 32-63. If they equal the magic cookie, the message is a STUN message. Its length is determined by the message length in bits 16 to 31 of the STUN packet. That STUN message is extracted and processed, and then the pointer in the data stream moves to the end of the STUN packet, and the process begins afresh. If bits 32-63 were not equal to the magic cookie, the agent uses application protocol specific framing to find the end of the application packet, and the process begins afresh.

The need to perform this demultiplexing, even over TCP, is the ugliest part of this specification. However, it is necessary to provide substantial reductions in call setup time possible by sending media on a validated candidate prior to its promotion to the m/c-line.

12. Sending Media

The procedures for sending media in the case of TCP are identical to

those defined in Section 7.13 of ICE, including the ability to use a

validated candidate immediately, in anticipation of its promotion into the m/c-line of a subsequent offer. This means that a connection can be opened and validated by ICE, and then immediately used for application traffic. This will require the demultiplexing described in the previous section to disambiguate STUN and application data.

In cases where the TCP connection is used for TLS, the TLS handshake procedures require one side to send the ClientHello message. This is normally the client which opened the TCP connection. However, in cases where a TCP connection was simultaneously opened, some mechanism is needed to decide who will send the ClientHello. With ICE-tcp, an agent knows that a TCP connection was simultaneously opened if it both sends and receives a STUN Binding Request on that connection. In such a case, the offerer of the associated candidate pair MUST send the TLS ClientHello.

13. Security Considerations

The main threat in ICE is hijacking of connections for the purposes of directing media streams to DoS targets or to malicious users. ICE-tcp prevents that by only using TCP connections that have been validated. Validation requires a STUN transaction to take place over the connection. This transaction cannot complete without both participants knowing a shared secret exchanged in the rendezvous protocol used with ICE, such as SIP. This shared secret, in turn, is protected by that protocol exchange. In the case of SIP, the usage of the sips mechanism is RECOMMENDED. When this is done, an attacker, even if it knows or can guess the port on which an agent is listening for incoming TCP connections, will not be able to open a connection and send media to the agent.

A more detailed analysis of this attack and the various ways ICE prevents it are described in [6]. Those considerations apply to this specification.

14. IANA Considerations

There are no IANA considerations associated with this specification.

15. Acknowledgements

The authors would like to thank Tim Moore, Francois Audet and Roni Even for the reviews and input on this document.

16. References

16.1. Normative References

[1] Rosenberg, J., "Simple Traversal of UDP Through Network Address Translators (NAT) (STUN)", draft-ietf-behave-rfc3489bis-03 (work in progress), March 2006.

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

[6] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Methodology for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", draft-ietf-mmusic-ice-08 (work in progress), March 2006.

[7] Rosenberg, J., "Obtaining Relay Addresses from Simple Traversal of UDP Through NAT (STUN)", draft-ietf-behave-turn-00 (work in progress), March 2006.

16.2. Informative References

[8] Guha, S., "NAT Behavioral Requirements for Unicast TCP", draft-ietf-behave-tcp-00 (work in progress), February 2006.

[9] Campbell, B., "The Message Session Relay Protocol", draft-ietf-simple-message-sessions-14 (work in progress), February 2006.

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

copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

Copyright Statement

Acknowledgment

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