MMUSIC
Internet-Draft
Intended status: Standards Track
Expires: August 28, 2008

J. Rosenberg
Cisco

February 25, 2008

TCP Candidates with Interactive Connectivity Establishment (ICE)
draft-ietf-mmusic-ice-tcp-06

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 Session Traversal Utilities for NAT (STUN). ICE provides a general framework for describing candidates, 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) [I-D.ietf-mmusic-ice] defines a mechanism for NAT traversal for multimedia communication protocols based on the offer/answer model [RFC3264] 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 Session Traversal Utilities for NAT (STUN) [I-D.ietf-behave-rfc3489bis]. However, ICE 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 [RFC4975]. 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 [RFC4571]. 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 this specification, agents can gather 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. 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 Traversal Using Relay NAT [I-D.ietf-behave-turn]. In this case, one of the agents would connect to its TURN server using TCP, and obtain a TCP-based relayed candidate. It would offer this to its peer agent as a candidate. The answerer would initiate a TCP connection towards the TURN server. When that connection is established, media can flow over the connections, through the TURN server. 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 use case is particularly important.

This specification extends ICE by defining its usage with TCP candidates. It also defines how ICE can be used with RTP and SRTP to provide both TCP and UDP candidates. This specification does so by following the outline of ICE itself, and calling out the additions

and changes necessary in each section of ICE to support TCP candidates.

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 candidate pairs.

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 simultaneous-open candidates. 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. A simultaneous-open candidate is one for which the agent will attempt to open a connection simultaneously with its peer.

Unlike UDP, there are no lite implementation defined for TCP. Instead, an implementation that meets the criteria for a lite implementation as discussed in Appendix A of [I-D.ietf-mmusic-ice] can just uses the mechanisms defined in [RFC4145], with constraints defined here on selection of attribute values.

When gathering candidates from a host interface, the agent typically obtains an active, passive and simultaneous-open candidates. Similarly, communications with a STUN server will provide server

reflexive and relayed versions of all three types. Connections to the STUN server are kept open during ICE processing.

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 always paired with passive, and simultaneous-open candidates with each other. When a connectivity check is to be made on a candidate pair, each agent determines whether it is to make a connection attempt for this pair.

> Why have both active and simultaneous-open candidates? Why not just simultaneous-open? The reason is that NAT treatment of simultaneous opens is currently not well defined, though specifications are being developed to address this [I-D.ietf-behave-tcp]. Some NATs 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. Alternatively, using 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. Simultaneous-opens are preferred since, if it does work, it will not require a relay even when both sides are behind a different NAT.

The actual process of generating connectivity checks, managing the state of the check list, and updating the Valid list, work identically for TCP as they do for UDP.

ICE requires an agent to demultiplex STUN and application layer traffic, since they appear on the same port. This demultiplexing is described by ICE, and is done using the magic cookie and other fields

of the message. Stream-oriented transports introduce another wrinkle, since they require a way to frame the connection so that the application and STUN packets can be extracted in order to determine which is which. For this reason, TCP media streams utilizing ICE use the basic framing provided in RFC 4571 [RFC4571], even if the application layer protocol is not RTP.

When TLS is in use (for non-RTP traffic) or DTLS (for RTP traffic), it runs over the RFC 4571 framing shim, so that STUN runs outside of the D/TLS connection (D/TLS is shorthand for TLS or DTLS). Pictorially:

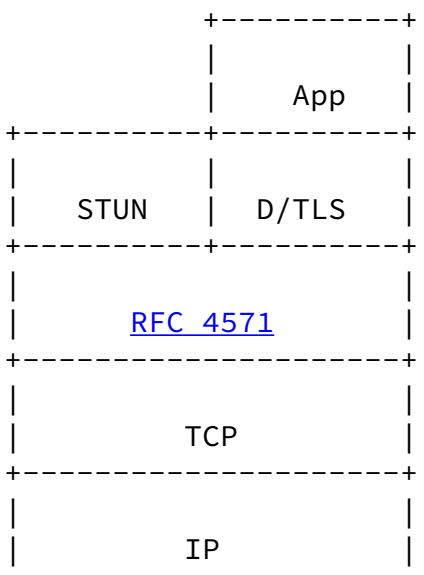

Figure 1: ICE TCP Stack

The implication of this is that, for any media stream protected by D/TLS, the agent will first run ICE procedures, exchanging STUN messages. Then, once ICE completes, D/TLS procedures begin. ICE and D/TLS are thus "peers" in the protocol stack. The STUN messages are not sent over the D/TLS connection, even ones sent for the purposes of keepalive in the middle of the media session.

When an updated offer is generated by the controlling endpoint, the SDP extensions for connection oriented media [RFC4145] are used to signal that an existing connection should be used, rather than opening a new one.

3. Sending the Initial Offer

If an offerer meets the criteria for lite as defined in Appendix A of [I-D.ietf-mmusic-ice], it omits any ICE attributes for its TCP-based media streams. Instead, the offerer follows the procedures defined in [RFC4145] for constructing the offer. However, the offerer MUST use a setup attribute of "actpass" for those streams.

For offerers making use of ICE for TCP streams, the procedures below are used.

3.1. Gathering Candidates

For each TCP capable media stream the agent wishes to use (including ones, like RTP, which can either be UDP or TCP), the agent SHOULD obtain two host candidates (each on a different port) for each

component of the media stream on each interface that the host has - one for the simultaneous open, and one for the passive candidate. If an agent is not capable of acting in one of these modes it would omit those candidates.

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.

Each agent SHOULD "obtain" an active host candidate for each component of each TCP capable media stream on each interface that the host has. The agent does not have to actually allocate a port for these candidates. These candidates serve as a placeholder for the creation of the check lists.

Next, the agent SHOULD take all host TCP candidates for a component that have the same foundation (there will typically be two - a passive and a simultaneous-open), and amongst them, pick two

arbitrarily. These two host candidates will be used to obtain

relayed and server reflexive candidates. To do that, the agent initiates a TCP connection from each candidate to the TURN server (resulting in two TCP connections). On each connection, it issues an Allocate request. One of the resulting relayed candidate is used as a passive relayed candidate, and the other, as a simultaneous-open relayed candidate. In addition, the Allocate responses will provide the agent with a server reflexive candidate for their corresponding host candidate.

For all of the remaining host candidates, if any, the agent only needs to obtain server reflexive candidates. To do that, it initiates a TCP connection from each host candidate to a STUN server, and uses a Binding request over that connection to learn the server reflexive candidate corresponding to that host candidate.

Once the Allocate or Binding request has completed, the agent MUST keep the TCP connection open until ICE processing has completed. See Section 1 for important implementation guidelines.

If a media stream is UDP-based (such as RTP), an agent MAY use an additional host TCP candidate to request a UDP-based candidate from a TURN server. Usage of the UDP candidate from the TURN server follows the procedures defined in ICE for UDP candidates.

Each agent SHOULD "obtain" an active relayed candidate for each component of each TCP capable media stream on each interface that the host has. The agent does not have to actually allocate a port for these candidates from the relay at this time. These candidates serve as a placeholder for the creation of the check lists.

Like its UDP counterparts, TCP-based STUN transactions are paced out at one every Ta seconds. This pacing refers strictly to STUN transactions (both Binding and Allocate requests). If performance of the transaction requires establishment of a TCP connection, then the connection gets opened when the transaction is performed.

3.2. Prioritization

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.

To accomplish this, the local preference SHOULD be defined as:

```
local-preference = (2^12)*(transport-pref) +
                   (2^9)*(direction-pref) +
                   (2^0)*(other-pref)
```

Transport-pref is the relative preference for candidates with this particular transport protocol (UDP or TCP), and direction-pref is the preference for candidates with this particular establishment directionality (active, passive, or simultaneous-open). Other-pref is used as a differentiator when two candidates would otherwise have identical local preferences.

Transport-pref MUST be between 0 and 15, with 15 being the most preferred. Direction-pref MUST be between 0 and 7, with 7 being the most preferred. Other-pref MUST be between 0 and 511, with 511 being the most preferred. For RTP-based media streams, it is RECOMMENDED that UDP have a transport-pref of 15 and TCP of 6. It is RECOMMENDED that, for all connection-oriented media, simultaneous-open candidates have a direction-pref of 7, active of 5 and passive of 2. If any two candidates have the same type-preference, transport-pref, and direction-pref, they MUST have a unique other-pref. With this specification, the only way that can happen is with multi-homed hosts, in which case other-pref is a preference amongst interfaces.

3.3. Choosing Default Candidates

The default candidate is chosen primarily based on the likelihood of it working with a non-ICE peer. 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 default candidates be UDP-based. However, the default SHOULD NOT be the simultaneous-open candidate.

If a media stream is inherently TCP-based, the agent MUST select the active candidate as default. This ensures proper directionality of connection establishment for NAT traversal with non-ICE implementations.

3.4. Encoding the SDP

TCP-based candidates are encoded into a=candidate lines identically to the UDP encoding described in [I-D.ietf-mmusic-ice]. However, the transport protocol is set to "tcp-so" for TCP simultaneous-open candidates, "tcp-act" for TCP active candidates, and "tcp-pass" for TCP 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). For active relayed candidates, the value for addr must be identical to the IP address of a passive or simultaneous-open

candidate from the same TURN server.

If the default candidate is TCP, the agent MUST include the a=setup and a=connection attributes from RFC 4145 [RFC4145], following the procedures defined there as if ICE was not in use. In particular, if an agent is the answerer, the a=setup attribute MUST meet the constraints in RFC 4145 based on the value in the offer. Since an ICE-tcp offerer always uses the active candidate as default, an ICE-tcp answerer will always use the passive attribute as default and include the a=setup:passive attribute in the answer.

If an agent is utilizing SRTP [RFC3711], it MAY include a mix of UDP and TCP candidates. If ICE selects a TCP candidate pair, the agent MUST still utilize SRTP, but run over the connection estabilised by ICE. The alternative, RTP over TLS, MUST NOT be used. This allows for the higher layer protocols (the security handshakes and media transport) to be independent of the underlying transport protocol. In the case of DTLS-SRTP [I-D.ietf-avt-dtls-srtp], the directionality attributes (a=setup) are utilized strictly to determine the direction of the DTLS handshake. Directionality of the TCP connection establishment are determined by the ICE attributes and procedures defined here.

If an agent is securing non-RTP media over TCP/TLS, he SDP MUST be constructed as described in RFC 4572 [RFC4572]. The directionality attributes (a=setup) are utilized strictly to determine the direction of the TLS handshake. Directionality of the TCP connection establishment are determined by the ICE attributes and procedures defined here.

4. Receiving the Initial Offer

4.1. Verifying ICE Support

Since this specification does not define a lite mode for ICE-tcp, a lite implementation will include candidate attributes for its UDP streams, but no such attributes for its TCP streams. An agent receiving such an offer MUST proceed with ICE in this case. ICE will

be used for the UDP streams, and [RFC4145] procedures will be used for the TCP streams. However, if the offer indicates a setup direction of actpass, the answerer MUST utilize a=setup:active in the answer. This is required to ensure proper directionality of connection establishment to work through NAT.

Similarly, if an agent is lite, and receives an offer that includes streams with TCP candidates, it will omit candidates from the answer for those streams. This will cause [RFC4145] procedures to be used

for those streams. In this case, the offer will indicate a direction of active, and the agent will use passive in its answer.

4.2. Forming the Check Lists

When forming candidate pairs, the following types of candidates can be paired with each other:

Local Candidate	Remote Candidate
tcp-so	tcp-so
tcp-act	tcp-pass
tcp-pass	tcp-act

When the agent prunes the check list, it MUST also remove any pair for which the local candidate is tcp-pass.

The remainder of check list processing works like the UDP case.

5. Connectivity Checks

5.1. STUN Client Procedures

5.1.1. Sending the Request

When an agent wants to send a TCP-based connectivity check, it first opens a TCP connection if none yet exists for the 5-tuple defined by the candidate pair for which the check is to be sent. This

connection is opened from the local candidate of the pair to the remote candidate of the pair. If the local candidate is tcp-act, the agent MUST open a connection from the interface associated with that local candidate. This connection MUST be opened from an unallocated port. For host candidates, this is readily done by connecting from the candidates interface. For relayed candidates, the agent uses the procedures in [I-D.ietf-behave-turn] to initiate a new connection from the specified interface on the TURN server.

Once the connection is established, the agent MUST utilize the shim defined in RFC 4571 [RFC4571] for the duration this connection remains open. The STUN Binding requests and responses are sent ontop of this shim, so that the length field defined in RFC 4571 precedes each STUN message. If TLS or DTLS-SRTP is to be utilized for the media session, the TLS or DTLS-SRTP handshakes will take place ontop of this shim as well. However, they only start once ICE processing

has completed. In essence, the TLS or DTLS-SRTP handshakes are considered a part of the media protocol. STUN is never run within the TLS or DTLS-SRTP session.

If the TCP connection cannot be established, the check is considered to have failed, and a full-mode agent MUST update the pair state to Failed in the check list.

Once the connection is established, client procedures are identical to those for UDP candidates. Note that STUN responses received on an active TCP candidate will typically produce a remote peer reflexive candidate.

5.2. STUN Server Procedures

An agent MUST be prepared to receive incoming TCP connection requests on any host or relayed TCP candidate that is simultaneous-open or passive. When the connection request is received, the agent MUST accept it. The agent MUST utilize the framing defined in RFC 4571 [RFC4571] for the lifetime of this connection. Due to this framing, the agent will receive data in discrete frames. Each frame could be media (such as RTP or SRTP), TLS, DLTS, or STUN packets. The STUN packets are extracted as described in Section 8.2.

Once the connection is established, STUN server procedures are

identical to those for UDP candidates. Note that STUN requests received on a passive TCP candidate will typically produce a remote peer reflexive candidate.

6. Concluding ICE Processing

If there are TCP candidates for a media stream, a controlling agent MUST use a regular selection algorithm.

When ICE processing for a media stream completes, each agent SHOULD close all TCP connections except the one between the candidate pairs selected by ICE.

> These two rules are related; the closure of connection on completion of ICE implies that a regular selection algorithm has to be used. This is because aggressive selection might cause transient pairs to be selected. Once such a pair was selected, the agents would close the other connections, one of which may be about to be selected as a better choice. This race condition may result in TCP connections being accidentally closed for the pair that ICE selects.

7. Subsequent Offer/Answer Exchanges

7.1. ICE Restarts

If an ICE restart occurs for a media stream with TCP candidate pairs that have been selected by ICE, the agents MUST NOT close the connections after the restart. In the offer or answer that causes the restart, an agent MAY include a simultaneous-open candidate whose transport address matches the previously selected candidate. If both agents do this, the result will be a simultaneous-open candidate pair matching an existing TCP connection. In this case, the agents MUST NOT attempt to open a new connection (or start new TLS or DTLS-SRTP procedures). Instead, that existing connection is reused and STUN checks are performed.

Once the restart completes, if the selected pair does not match the previously selected pair, the TCP connection for the previously selected pair SHOULD be closed by the agent.

8. Media Handling

8.1. Sending Media

When sending media, if the selected candidate pair matches an existing TCP connection, that connection MUST be used for sending media.

The framing defined in RFC 4571 MUST be used when sending media. For media streams that are not RTP-based and do not normally use RFC 4571, the agent treats the media stream as a byte stream, and assumes that it has its own framing of some sort. It then takes an arbitrary number of bytes from the bytestream, and places that as a payload in the RFC 4571 frames, including the length. Next, the sender checks to see if the resulting set of bytes would be viewed as a STUN packet based on the rules in sections 6 and 8 of [I-D.ietf-behave-rfc3489bis]. This includes a check on the most significant two bits, the magic cookie, the length, and the fingerprint. If, based on those rules, the bytes would be viewed as a STUN message, the sender SHOULD utilize a different number of bytes so that the length checks will fail. Though it is normally highly unlikely that an arbitrary number of bytes from a bytestream would resemble a STUN packet based on all of the checks, it can happen if the content of the application stream happens to contain a STUN message (for example, a file transfer of logs from a client which includes STUN messages).

If TLS or DTLS-SRTP procedures are being utilized to protect the

media stream, those procedures start at the point that media is permitted to flow, as defined in the ICE specification [I-D.ietf-mmusic-ice]. The TLS or DTLS-SRTP handshakes occur ontop of the RFC 4571 shim, and are considered part of the media stream for purposes of this specification.

8.2. Receiving Media

The framing defined in RFC 4571 MUST be used when receiving media. For media streams that are not RTP-based and do not normally use RFC 4571, the agent extracts the payload of each RFC 4571 frame, and

determines if it is a STUN or an application layer data based on the procedures in ICE [I-D.ietf-mmusic-ice]. If media is being protected with DTLS-SRTP, the DTLS, RTP and STUN packets are demultiplexed as described in Section 3.6.2 of [I-D.ietf-avt-dtls-srtp].

For non-STUN data, the agent appends this to the ongoing bytestream collected from the frames. It then parses the bytestream as if it had been directly received over the TCP connection. This allows for ICE-tcp to work without regard to the framing mechanism used by the application layer protocol.

9. Connection Management

9.1. Connections Formed During Connectivity Checks

Once a TCP or TCP/TLS connection is opened by ICE for the purpose of connectivity checks, its lifecycle depends on how it is used. If that candidate pair is selected by ICE for usage for media, an agent SHOULD keep the connection open until:

- The session terminates
- The media stream is removed
- An ICE restart takes place, resulting in the selection of a different candidate pair.

In these cases, the agent SHOULD close the connection when that event occurs. This applies to both agents in a session, in which case usually one of the agents will end up closing the connection first.

If a connection has been selected by ICE, an agent MAY close it anyway. As described in the next paragraph, this will cause it to be reopened almost immediately, and in the interim media cannot be sent. Consequently, such closures have a negative effect and are NOT RECOMMENDED. However, there may be cases where an agent needs to

close a connection for some reason.

If an agent needs to send media on the selected candidate pair, and its TCP connection has closed, either on purpose or due to some

error, then:

- If the agent's local candidate is tcp-act or tcp-so, it MUST reopen a connection to the remote candidate of the selected pair.

- If the agent's local candidate is tcp-pass, the agent MUST await an incoming connection request, and consequently, will not be able to send media until it has been opened.

If the TCP connection is established, the framing of RFC 4571 is utilized. If the agent opened the connection, it MUST send a STUN connectivity check. An agent MUST be prepared to receive a connectivity check over a connection it opened or accepted (note that this is true in general; ICE requires that an agent be prepared to receive a connectivity check at any time, even after ICE processing completes). If an agent receives a connectivity check after re-establishment of the connection, it MUST generate a triggered check over that connection in response if it has not already sent a check. Once an agent has sent a check and received a successful response, the connection is considered Valid and media can be sent (which includes a TLS or DTLS-SRTP session resumption or restart).

If the TCP connection cannot be established, the controlling agent SHOULD restart ICE for this media stream. This will happen in cases where one of the agents is behind a NAT with connection dependent mapping properties [I-D.ietf-behave-tcp].

9.2. Connections formed for Gathering Candidates

If the agent opened a connection to a STUN server for the purposes of gathering a server reflexive candidate, that connection SHOULD be closed by the client once ICE processing has completed. This happens irregardless of whether the candidate learned from the STUN server was selected by ICE.

If the agent opened a connection to a TURN server for the purposes of gathering a relayed candidate, that connection MUST be kept open by the client for the duration of the media session if:

- A relayed candidate learned by the TURN server was selected by ICE,

- or an active candidate established as a consequence of a Connect request sent through that TCP connection was selected by ICE.

Otherwise, the connection to the TURN server SHOULD be closed once ICE processing completes.

If, despite efforts of the client, a TCP connection to a TURN server fails during the lifetime of the media session utilizing a transport address allocated by that server, the client SHOULD reconnect to the TURN server, obtain a new allocation, and restart ICE for that media stream.

10. 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 [I-D.ietf-mmusic-ice]. Those considerations apply to this specification.

11. IANA Considerations

There are no IANA considerations associated with this specification.

12. Acknowledgements

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

13. References

13.1. Normative References

[I-D.ietf-behave-rfc3489bis]
Rosenberg, J., Mahy, R., Matthews, P., and D. Wing,

draft-ietf-behave-rfc3489bis-14 (work in progress), February 2008.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[RFC4145] Yon, D. and G. Camarillo, "TCP-Based Media Transport in the Session Description Protocol (SDP)", RFC 4145, September 2005.

[RFC4571] Lazzaro, J., "Framing Real-time Transport Protocol (RTP) and RTP Control Protocol (RTCP) Packets over Connection-Oriented Transport", RFC 4571, July 2006.

[RFC4572] Lennox, J., "Connection-Oriented Media Transport over the Transport Layer Security (TLS) Protocol in the Session Description Protocol (SDP)", RFC 4572, July 2006.

[I-D.ietf-mmusic-ice]
Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", draft-ietf-mmusic-ice-19 (work in progress), October 2007.

[I-D.ietf-avt-dtls-srtp]
McGrew, D. and E. Rescorla, "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for Secure Real-time Transport Protocol (SRTP)", draft-ietf-avt-dtls-srtp-01 (work in progress), November 2007.

[I-D.ietf-behave-turn]
Rosenberg, J., Mahy, R., and P. Matthews, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", draft-ietf-behave-turn-06 (work in progress), January 2008.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K.

Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

13.2. Informative References

[I-D.ietf-behave-tcp]
Guha, S., "NAT Behavioral Requirements for TCP", draft-ietf-behave-tcp-07 (work in progress), April 2007.

[RFC4975] Campbell, B., Mahy, R., and C. Jennings, "The Message Session Relay Protocol (MSRP)", RFC 4975, September 2007.

1. Implementation Considerations for BSD Sockets

This specification requires unusual handling of TCP connections, the implementation of which in traditional BSD socket APIs is non-trivial.

In particular, ICE requirs an agent to obtain a local TCP candidate, bound to a local IP and port, and then from that local port, initiate a TCP connection (to the STUN server, in order to obtain server reflexive candidates, to the TURN server, to obtain a relayed candidate, or to the peer as part of a connectivity check), and be prepared to receive incoming TCP connections (for passive and simultaneous-open candidates). A "typical" BSD socket is used either for initiating or receiving connections, and not for both. The code required to allow incoming and outgoing connections on the same local IP and port is non-obvious. The following pseudocode, contributed by Saikat Guha, has been found to work on many platforms:

```
for i in 0 to MAX
   sock_i = socket()
   set(sock_i, SO_REUSEADDR)
   bind(sock_i, local)

listen(sock_0)
connect(sock_1, stun)
connect(sock_2, remote_a)
connect(sock_3, remote_b)
```

The key here is that, prior to the listen() call, the full set of sockets that need to be utilized for outgoing connections must be allocated and bound to the local IP address and port. This number, MAX, represents the maximum number of TCP connections to different destinations that might need to be established from the same local candidate. This number can be potentially large for simultaneous-open candidates. If a request forks, ICE procedures may take place with multiple peers. Furthermore, for each peer, connections would need to be established to each passive or simultaneous-open candidate for the same component. If we assume a worst case of 5 forked branches, and for each peer, five simultaneous-open candidates, that results in MAX=25. For a passive candidate, MAX is equal to the number of STUN servers, since the agent only initiates TCP connections on a passive candidate to its STUN server.

Author's Address

Jonathan Rosenberg
Cisco
Edison, NJ
US

Email: jdrosen@cisco.com
URI: http://www.jdrosen.net

Full Copyright Statement

Intellectual Property

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).