Internet Engineering Task Force (IETF)
Request for Comments: 6544
Category: Standards Track
ISSN: 2070-1721

J. Rosenberg
jdrosen.net
A. Keranen
Ericsson
B. B. Lowekamp
Skype
A. B. Roach
Tekelec

March 2012

# TCP Candidates with Interactive Connectivity Establishment (ICE)

## 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 media streams. 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.

## Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6544.

Table of Contents

## 1. Introduction

Interactive Connectivity Establishment (ICE) [RFC5245] 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) [RFC5389]. However, ICE only defines procedures for UDP-based transport protocols.

为什么对tcp的支持是需要的？

There are many reasons why ICE support for TCP is important. First, there are media protocols that only run over TCP. Such protocols are used, for example, for screen 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 can also 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 a media 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 Relays around NAT (TURN) [RFC5766]. 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 other agent 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 work" in as 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 Secure RTP (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 to support TCP candidates in ICE. The base behavior of ICE [RFC5245] remains unchanged except for the extensions in this document that define the usage of ICE with TCP candidates.

It should be noted that since TCP NAT traversal is more complicated than with UDP, ICE TCP is not generally as efficient as UDP-based ICE. Discussion about this topic can be found in Appendix A.

## 2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

This document uses the same terminology as ICE (see Section 3 of [RFC5245]).

## 3. Overview of Operation

The usage of ICE with TCP is relatively straightforward. This specification mainly deals with how and when connections are opened and how those connections relate to candidate pairs.

这里描述很重要

When agents perform address allocations to gather TCP-based candidates, three types of candidates can be obtained: active candidates, passive candidates, and simultaneous-open (S-O) 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. An S-O candidate is one for which the agent will attempt to open a connection simultaneously with its peer.

When gathering candidates from a host interface, the agent typically obtains active, passive, and S-O candidates. Similarly, one can use different techniques for obtaining, e.g., server reflexive, NAT-assisted, tunneled, or relayed candidates of these three types (see Section 5). Connections to servers used for relayed and server reflexive candidates 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, both IP address and port are present, but the port is meaningless (it is there only for making encoding of active candidates consistent with the other candidate types and 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 S-O 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.

The actual process of generating connectivity checks, managing the state of the check list, and updating the Valid list works identically for TCP as it does for UDP.

ICE requires an agent to demultiplex STUN and application-layer traffic, since they appear on the same port. This demultiplexing is described in [RFC5245] 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 differentiate STUN packets from application-layer traffic. 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 Transport Layer Security (TLS) or Datagram Transport Layer Security (DTLS) is used, they are also run over the RFC 4571 framing shim, while STUN runs outside of the (D)TLS connection. The resulting ICE TCP protocol stack is shown in Figure 1, with (D)TLS on the left side and without it on the right side.

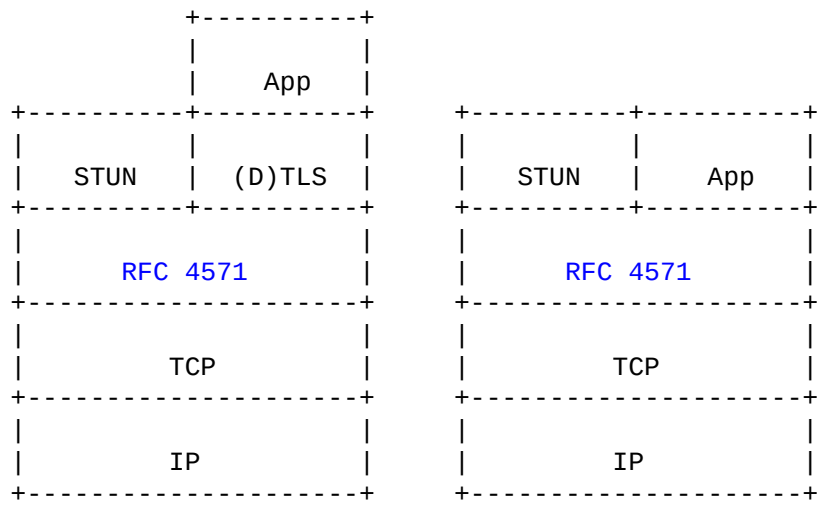


Figure 1: ICE TCP Stack with and without (D)TLS

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.

## 4. Sending the Initial Offer

For offerers making use of ICE for TCP streams, the procedures below are used. The main differences compared to UDP candidates are the new methods for gathering candidates, how TCP candidates are prioritized, and how they are encoded in the Session Description Protocol (SDP) offer and answer.

### 4.1. Gathering Candidates

Providers of real-time communications services may decide that it is preferable to have no media at all rather than to have media over TCP. To allow for choice, it is RECOMMENDED that it be possible to configure agents to either obtain or not 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. Besides, network characteristics and connectivity assumptions can, and will, change over time. Just because an 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 an agent is behind a NAT with endpoint-independent mapping today doesn't mean that tomorrow it won't pick up its 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 made 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 that 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.

First, agents SHOULD obtain host candidates as described in Section 5.1. Then, each agent SHOULD "obtain" (allocate a placeholder for) an active host candidate for each component of each TCP-capable media stream on each interface that the host has. The agent does not yet have to actually allocate a port for these candidates, but they are used for the creation of the check lists.

The agent SHOULD then obtain server reflexive, NAT-assisted, and/or UDP-tunneled candidates (see Section 5.2, Section 5.3, and Section 5.4). The mechanisms for establishing these candidates and the number of candidates to collect vary from technique to technique. These considerations are discussed in the relevant sections.

Next, agents SHOULD obtain passive (and possibly S-O) relayed candidates for each component as described in Section 5.5. Each agent SHOULD also allocate a placeholder for an active relayed candidate for each component of each TCP-capable media stream.

It is highly RECOMMENDED that a host obtains at least one set of host candidates and one set of relayed candidates. Obtaining additional candidates will increase the chance of successfully creating a direct connection.

Once the candidates have been obtained, the agent MUST keep the TCP connections open until ICE processing has completed. See Appendix B 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 (or some other relay with similar functionality). Usage of such UDP candidates follows the procedures defined in ICE for UDP candidates.

Like its UDP counterparts, TCP-based STUN transactions are paced out at one every Ta milliseconds (see Section 16 of [RFC5245]). 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.

## 4.2. 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.

In Section 4.1.2.1 of [RFC5245], a recommended formula for UDP ICE candidate prioritization is defined. For TCP candidates, the same formula and candidate type preferences SHOULD be used, and the RECOMMENDED type preferences for the new candidate types defined in this document (see Section 5) are 105 for NAT-assisted candidates and 75 for UDP-tunneled candidates.

When both UDP and TCP candidates are offered for the same media stream, and one transport protocol should be preferred over the other, the type preferences for the preferred transport protocol candidates SHOULD be increased and/or the type preferences for the other transport protocol candidates SHOULD be decreased. How much the values should be increased or decreased depends on whether it is more important to choose a certain transport protocol or a certain candidate type. If the candidate type is more important (e.g., even if UDP is preferred, TCP host candidates are preferred over UDP server reflexive candidates) changing type preference values by one for the other transport protocol candidates is enough. On the other hand, if the transport protocol is more important (e.g., any UDP candidate is preferred over any TCP candidate), all the preferred transport protocol candidates SHOULD have type preference higher than the other transport protocol candidates. However, it is RECOMMENDED that the relayed candidates are still preferred lower than the other candidate types. For RTP-based media streams, it is RECOMMENDED that UDP candidates are preferred over TCP candidates.

With TCP candidates, the local preference part of the recommended priority formula is updated to also include the directionality (active, passive, or simultaneous-open) of the TCP connection. The RECOMMENDED local preference is then defined as:

```
local preference = (2^13) * direction-pref + other-pref
```

The direction-pref MUST be between 0 and 7 (both inclusive), with 7 being the most preferred. The other-pref MUST be between 0 and 8191 (both inclusive), with 8191 being the most preferred. It is RECOMMENDED that the host, UDP-tunneled, and relayed TCP candidates have the direction-pref assigned as follows: 6 for active, 4 for passive, and 2 for S-O. For the NAT-assisted and server reflexive candidates, the RECOMMENDED values are: 6 for S-O, 4 for active, and 2 for passive.

The preference priorities listed here are simply recommendations that try to strike a balance between success probability and the resulting path's efficiency. Depending on the scenario where ICE TCP is used,

different values may be appropriate. For example, if the overhead of a UDP tunnel is not an issue, those candidates should be prioritized higher since they are likely to have a high success probability. Also, simultaneous-open is prioritized higher than active and passive candidates for NAT-assisted and server reflexive candidates since if TCP S-O is supported by the operating systems of both endpoints, it should work at least as well as the active-passive approach. If an implementation is uncertain whether S-O candidates are supported, it may be reasonable to prioritize them lower. For host, UDP-tunneled, and relayed candidates, the S-O candidates are prioritized lower than active and passive since active-passive candidates should work with them at least as well as the S-O candidates.

If any two candidates have the same type-preference and direction-pref, they MUST have a unique other-pref. With this specification, this usually only happens with multi-homed hosts, in which case other-pref is the preference for the particular IP address from which the candidate was obtained. When there is only a single IP address, this value SHOULD be set to the maximum allowed value (8191).

### 4.3. 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 a simultaneous-open candidate.

If a media stream is inherently TCP-based, it is RECOMMENDED for an offering full agent to select an active candidate as the default candidate and use [RFC4145] "setup" attribute value "active". This increases the chances for a successful NAT traversal even without ICE support if the agent is behind a NAT and the peer is not. For the same reason, for a lite agent, it is RECOMMENDED to use a passive candidate and "setup" attribute value "passive" in the offer.

### 4.4. Lite Implementation Requirements

If an offerer meets the criteria for the lite mode as described in Appendix A of [RFC5245] (i.e., it will always have a public, globally unique IP address), it MAY use the lite mode of ICE for TCP candidates. In the lite mode, for TCP candidates, only passive host candidates are gathered, unless active candidates are needed as the default candidates. Otherwise, the procedures described for lite mode in [RFC5245] also apply to TCP candidates. If UDP and TCP candidates are mixed in a media stream, the mode (lite or full) applies to both UDP and TCP candidates.

### 4.5. Encoding the SDP

TCP-based candidates are encoded into a=candidate lines like the UDP candidates described in [RFC5245]. However, the transport protocol (i.e., value of the transport-extension token defined in [RFC5245], Section 15.1) is set to "TCP" and the connection type (active, passive, or S-O) is encoded using a new extension attribute. With TCP candidates, the candidate-attribute syntax with Augmented BNF [RFC5234] is then:

```
candidate-attribute   = "candidate" ":" foundation SP component-id SP
                        "TCP" SP
                        priority SP
                        connection-address SP
                        port SP
                        cand-type
                        [SP rel-addr]
                        [SP rel-port]
                        SP tcp-type-ext
                        *(SP extension-att-name SP
                             extension-att-value)

tcp-type-ext          = "tcptype" SP tcp-type
tcp-type              = "active" / "passive" / "so"
```

The connection-address encoded into the candidate-attribute for active candidates MUST be set to the IP address that will be used for the attempt, but the port(s) MUST be set to 9 (i.e., Discard). For active relayed candidates, the value for connection-address MUST be identical to the IP address of a passive or simultaneous-open candidate from the same relay server.

If the default candidate is TCP-based, the agent MUST include the a=setup and a=connection attributes from RFC 4145 [RFC4145], following the procedures defined there as if ICE were 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.

If an agent is utilizing SRTP [RFC3711], it MAY include a mix of UDP and TCP candidates. If ICE selects a TCP candidate pair, it is RECOMMENDED that the agent still utilizes SRTP but runs it over the connection established by ICE. The alternative, RTP over TLS, breaks RTP header compression and on-path RTP analysis tools and hence SHOULD be avoided. In the case of DTLS-SRTP [RFC5764], the directionality attributes (a=setup) are utilized strictly to determine the direction of the DTLS handshake. Directionality of the TCP connection establishment is determined by the ICE attributes and procedures defined here.

If an agent is securing non-RTP media over TCP/TLS, the 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 is determined by the ICE attributes and procedures defined here.

Examples of SDP offers and answers with ICE TCP extensions are shown in Appendix C.

## 5. Candidate Collection Techniques

The following sections discuss a number of techniques that can be used to obtain candidates for use with ICE TCP. It is important to note that this list is not intended to be exhaustive, nor is implementation of any specific technique beyond host candidates (Section 5.1) considered mandatory.

Implementors are encouraged to implement as many of the following techniques from the following list as is practical, as well as to explore additional NAT-traversal techniques beyond those discussed in this document. However, to get a reasonable success ratio, one SHOULD implement at least one relayed technique (e.g., TURN) and one technique for discovering the address given for the host by a NAT (e.g., STUN).

To increase the success probability with the techniques described below and to aid with transition to IPv6, implementors SHOULD take particular care to include both IPv4 and IPv6 candidates as part of the process of gathering candidates. If the local network or host does not support IPv6 addressing, then clients SHOULD make use of other techniques, e.g., TURN-IPv6 [RFC6156], Teredo [RFC4380], or SOCKS IPv4-IPv6 gatewaying [RFC3089], for obtaining IPv6 candidates.

While implementations SHOULD support as many techniques as feasible, they SHOULD also consider which of them to use if multiple options are available. Since different candidates are paired with each other, offering a large number of candidates results in a large check list and potentially long-lasting connectivity checks. For example, using multiple NAT-assisted techniques with the same NAT usually results only in redundant candidates. Similarly, using just one of the multiple UDP tunneling or relaying techniques is often enough.

### 5.1. Host Candidates

Host candidates are the most simple candidates since they only require opening TCP sockets on the host's interfaces and sending/receiving connectivity checks from them. However, if the hosts are

behind different NATs, host candidates usually fail to work. On the other hand, if there are no NATs between the hosts, host candidates are the most efficient method since they require no additional NAT traversal protocols or techniques.

For each TCP-capable media stream the agent wishes to use (including ones like RTP that can be either 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.

## 5.2. Server Reflexive Candidates

Server reflexive techniques aim to discover the address a NAT has given for the host by asking that from a server on the other side of the NAT and then creating proper bindings (unless such already exist) on the NATs with connectivity checks sent between the hosts. Success of these techniques depends on the NATs' mapping and filtering behavior [RFC5382] and also on whether the NATs and hosts support the TCP simultaneous-open technique.

Obtaining server reflexive passive candidates may require initiating connections from host's passive candidates; see Appendix B for implementation details on this. Server reflexive active candidates can be derived from passive or S-O candidates by using the same IP addresses and interfaces as those candidates. It is useful to obtain both server reflexive passive and S-O candidates since which one actually works better depends on the hosts and NATs. Furthermore, some techniques (e.g., TURN relaying) require knowing the IP address of the peer's active candidates beforehand, so active server reflexive candidates are needed for such techniques to function properly.

A widely used protocol for obtaining server reflexive candidates is STUN. Its TCP-specific behavior is described in [RFC5389], Section 7.2.2.

## 5.3. NAT-Assisted Candidates

NAT-assisted techniques communicate with the NATs directly and, in this way, discover the address that the NAT has given to the host. NAT-assisted techniques also create proper bindings on the NATs. The benefit of these techniques over the server reflexive techniques is that the NATs can adjust their mapping and filtering behavior so that connections can be successfully created. A downside of NAT-assisted techniques is that they commonly allow communicating only with a NAT

that is in the same subnet as the host and thus often fail in scenarios with multiple layers of NATs. These techniques also rely on NATs supporting the specific protocols and allowing the users to modify their behavior.

These candidates are encoded in the ICE offer and answer like the server reflexive candidates, but they (commonly) use a higher priority (as described in Section 4.2) and hence are tested before the server reflexive candidates.

Currently, the Universal Plug and Play (UPnP) forum's Internet Gateway Device (IGD) protocol [UPnP-IGD] and the NAT Port Mapping Protocol (PMP) [NAT-PMP] are widely supported NAT-assisted techniques. Other known protocols include Port Control Protocol (PCP) [PCP-BASE], SOCKS [RFC1928], Realm Specific IP (RSIP) [RFC3103], and Simple Middlebox Configuration (SIMCO) [RFC4540]. Also, the Middlebox Communications (MIDCOM) MIB [RFC5190] defines a mechanism based on the Simple Network Management Protocol (SNMP) for controlling NATs.

### 5.4. UDP-Tunneled Candidates

UDP-tunneled NAT traversal techniques utilize the fact that UDP NAT traversal is simpler and more efficient than TCP NAT traversal. With these techniques, the TCP packets (or possibly complete IP packets) are encapsulated in UDP packets. Because of the encapsulation, these techniques increase the overhead for the connection and may require support from both of the endpoints, but on the other hand, UDP tunneling commonly results in reliable and fairly simple TCP NAT traversal.

UDP-tunneled candidates can be encoded in the ICE offer and answer either as relayed or server reflexive candidates, depending on whether the tunneling protocol utilizes a relay between the hosts. The UDP-tunneled candidates may appear to applications as host candidates from a local pseudo-interface. Treating these candidates as host candidates results in incorrect prioritization and possibly non-optimal candidate selection. Implementations may attempt to detect pseudo-interfaces, e.g., from the address prefix of the interface, but detection details vary from technique to technique.

For example, the Teredo protocol [RFC4380] [RFC6081] provides automatic UDP tunneling and IPv6 interworking. The Teredo UDP tunnel is visible to the host application as an IPv6 address; thus, Teredo candidates are encoded as IPv6 addresses.

### 5.5. Relayed Candidates

Relaying packets through a relay server is often the NAT traversal technique that has the highest success probability: communicating via a relay that is in the public Internet looks like normal client-server communication for the NATs and is supported in practice by all existing NATs, regardless of their filtering and mapping behavior. However, using a relay has several drawbacks, e.g., it usually results in a sub-optimal path for the packets, the relay needs to exist and it needs to be discovered, the relay is a possible single point of failure, relaying consumes potentially a lot of resources of the relay server, etc. Therefore, relaying is often used as the last resort when no direct path can be created with other NAT traversal techniques.

With relayed candidates, the host commonly needs to obtain only a passive candidate since any of the peer's server reflexive (and NAT-assisted if the peer can communicate with the outermost NAT) active candidates should work with the passive relayed candidate. However, if the relay is behind a NAT or a firewall, also using active and S-O candidates will increase success probability.

Relaying protocols capable of relaying TCP connections include TURN TCP [RFC6062] and SOCKS [RFC1928] (which can also be used for IPv4-IPv6 gatewaying [RFC3089]). It is also possible to use a Secure SHell (SSH) [RFC4251] tunnel as a relayed candidate if a suitable server is available and the server permits this.

## 6. Receiving the Initial Offer and Answer

Handling an ICE offer with TCP candidates works in a similar way as with UDP candidates. First, ICE support is verified (including the check for ice-mismatch described in Section 5.1 of [RFC5245]) and agent roles are determined. Candidates are gathered using the techniques described in Section 5 and prioritized as described in Section 4.2. Default candidates are selected taking into account the considerations in Section 4.3. The SDP answer is encoded as in Section 4.3 of [RFC5245], with the exception of TCP candidates whose encoding is described in Section 4.5.

When the offerer receives the initial answer, it also verifies ICE support and determines its role. If both of the agents use lite implementations, the offerer takes the controlling role and uses the procedures defined in [RFC4145] to select the most preferred candidate pair with a new offer.

### 6.1. Considerations with Two Lite Agents

If both agents are using the lite mode and if the offerer uses the a=setup:active attribute [RFC4145] in the new offer, the offerer MAY initiate the TCP connection on the selected pair in parallel with the new offer to speed up the connection establishment. Consequently, the answerer MUST still accept incoming TCP connections to any of the passive candidates it listed in the answer, from any of the IP addresses the offerer listed in the initial offer.

If the answerer receives the new offer matching the candidate pair where a connection was already created in parallel with the new offer, it MUST accept the offer and respond to it while keeping the already-created connection. If the connection that was created in parallel with the new offer does not match the candidate pair in the new offer, the connection MUST be closed, and ICE restart SHOULD be performed.

Since the connection endpoints are not authenticated using the connectivity checks in the scenario where both agents use the lite mode, unless media-level security (e.g., TLS) is used, it is RECOMMENDED to use the full mode instead. For more lite versus full implementation considerations, see Appendix A of [RFC5245].

### 6.2. Forming the Check Lists

As with UDP, check lists are formed only by full ICE implementations. When forming candidate pairs, the following types of TCP candidates can be paired with each other:

| Local Candidate | Remote Candidate |
|---|---|
| tcp-so | tcp-so |
| tcp-active | tcp-passive |
| tcp-passive | tcp-active |

When the agent prunes the check list, it MUST also remove any pair for which the local candidate is a passive TCP candidate. With pruning, the NAT-assisted candidates are treated like server reflexive candidates if the base is also used as a host candidate.

The remainder of check list processing works in the same way as the UDP case.

## 7. Connectivity Checks

The TCP connectivity checks, like with UDP, are generated only by full implementations. The TCP candidate pairs are in the same check list with the UDP candidate pairs, and they are scheduled for connectivity checks, as described in Section 5.8 of [RFC5245], based on the priority order.

### 7.1. STUN Client Procedures

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-active, the agent MUST open a connection from the interface associated with that local candidate. This connection SHOULD be opened from an unallocated port. For host candidates, this is readily done by connecting from the local candidate's interface. For relayed, NAT-assisted, and UDP-tunneled candidates, the agent may need to use additional procedures specific to the protocol.

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 on top 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 on top 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 as part of the ICE procedures.

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. See Section 7.2.2 of [RFC5389] for more details on STUN over TCP.

Once the connection is established, client procedures are identical to those for UDP candidates. However, retransmissions of the STUN connectivity check messages are not needed, since TCP takes care of reliable delivery of the messages. Note also that STUN responses received on an active TCP candidate will typically produce a peer reflexive candidate. If the response to the first connectivity check on the established TCP connection is something other than a STUN

message, the remote candidate address apparently was not one of the peer's addresses, and the agent SHOULD close the connection and consider all pairs with that remote candidate as failed.

### 7.2. STUN Server Procedures

An ICE TCP agent, full or lite, MUST be prepared to receive incoming TCP connection requests on the base of any 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, DTLS, or STUN packets. The STUN packets are extracted as described in Section 10.2.

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.

## 8. Concluding ICE Processing

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

When ICE processing for a media stream completes, each agent SHOULD close all TCP connections (that were opened due to this ICE session) except the ones 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 is 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.

## 9. Subsequent Offer/Answer Exchanges

### 9.1. Updated Offer

When an updated offer is generated by the controlling endpoint after the connectivity checks have succeeded, 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.

### 9.2. 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.

## 10. Media Handling

### 10.1. 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, if needed. It then takes an arbitrary number of bytes from the byte stream 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 [RFC5389]. 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 MUST 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 byte stream 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 that 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 [RFC5245]. The TLS or DTLS-SRTP handshakes occur on top of the RFC 4571 shim and

are considered part of the media stream for the purposes of this specification.

### 10.2. 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 [RFC5245]. If media is being protected with DTLS-SRTP, the DTLS, RTP, and STUN packets are demultiplexed as described in Section 5.1.2 of [RFC5764].

For non-STUN data, the agent appends this to the ongoing byte stream collected from the frames. It then parses the byte stream 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.

## 11. Connection Management

### 11.1. Connections Formed during Connectivity Checks

Once a TCP or TCP/TLS connection is opened by ICE for the purpose of connectivity checks, its life cycle 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, or
- an ICE restart takes place, resulting in the selection of a different candidate pair.

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

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

- If the agent's local candidate is tcp-passive, 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 and is a full agent, 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 a full 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 [RFC5382].

### 11.2. Connections Formed for Gathering Candidates

If the agent opened a connection to a STUN server, or another similar 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 regardless of whether the candidate learned from the 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 from the TURN server 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. Similar measures SHOULD apply also to other types of relaying servers.

## 12. Security Considerations

The main threat in ICE is hijacking of connections for the purposes of directing media streams to denial-of-service (DoS) targets or to malicious users. When full implementations are used, 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 [RFC3261]. This shared secret, in turn, is protected by that protocol exchange. In the case of SIP, the usage of the SIP Secure (SIPS) [RFC3261] 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.

If the rendezvous protocol exchange is compromised, the shared secret can be learned by an attacker, and the attacker may be able to fake the connectivity check validation and open a TCP connection to the target. Hence, using additional security mechanisms (e.g., application-layer security) that mitigate these risks is RECOMMENDED.

A STUN amplification attack is described in Section 18.5.2 of [RFC5245]. The same considerations apply to TCP, but the amplification effect with TCP is larger due to need for establishing a TCP connection before any checks are performed. Therefore, an ICE agent SHOULD NOT have more than 5 outstanding TCP connection attempts with the same peer to the same IP address.

If both agents use the lite mode, no connectivity checks are sent, and additional procedures (e.g., media-level security) are needed to validate the connection. The lack of connectivity checks is especially problematic if one of the hosts is behind a NAT and has an address from a private address space: the peer may accidentally connect to a host in a different subnet that uses the same private address space. This is one of the reasons why the lite mode is not appropriate for an ICE agent located behind a NAT.

A more detailed analysis of different attacks and the various ways ICE prevents them are described in [RFC5245]. Those considerations apply to this specification.

## 13. IANA Considerations

IANA has created a new sub-registry "ICE Transport Protocols" in the "Interactive Connectivity Establishment (ICE)" registry for ICE candidate-attribute transport extensions. Initial values are given below; future assignments are to be made through IETF Review or IESG Approval [RFC5226]. Assignments consist of an extension token (as defined in Section 15.1 of [RFC5245]) and a reference to the document defining the extension.

| Token | Reference |
|---|---|
| UDP | RFC 5245, Section 15.1 |
| TCP | RFC 6544, Section 4.5 |

## 14. Acknowledgements

The authors would like to thank Tim Moore, Saikat Guha, Francois Audet, Roni Even, Simon Perreault, Alfred Heggestad, Hadriel Kaplan, Jonathan Lennox, Flemming Andreasen, Dan Wing, and Vijay Gurbani for the reviews and input on this document. Special thanks to Marc Petit-Huguenin for providing the SDP examples.

## 15. References

### 15.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

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

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

[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.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.

[RFC5764] McGrew, D. and E. Rescorla, "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)", RFC 5764, May 2010.

[RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, April 2010.

## 15.2. Informative References

[IMC05] Guha, S. and P. Francis, "Characterization and Measurement of TCP Traversal through NATs and Firewalls", Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement, 2005.

[NAT-PMP] Cheshire, S., Krochmal, M., and K. Sekar, "NAT Port Mapping Protocol (NAT-PMP)", Work in Progress, April 2008.

[PCP-BASE] Wing, D., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", Work in Progress, March 2012.

[RFC1928] Leech, M., Ganis, M., Lee, Y., Kuris, R., Koblas, D., and L. Jones, "SOCKS Protocol Version 5", RFC 1928, March 1996.

[RFC3089] Kitamura, H., "A SOCKS-based IPv6/IPv4 Gateway Mechanism", RFC 3089, April 2001.

[RFC3103] Borella, M., Grabelsky, D., Lo, J., and K. Taniguchi, "Realm Specific IP: Protocol Specification", RFC 3103, October 2001.

[RFC4251] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Protocol Architecture", RFC 4251, January 2006.

[RFC4380] Huitema, C., "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)", RFC 4380, February 2006.

[RFC4540] Stiemerling, M., Quittek, J., and C. Cadar, "NEC's Simple Middlebox Configuration (SIMCO) Protocol Version 3.0", RFC 4540, May 2006.

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

[RFC5190] Quittek, J., Stiemerling, M., and P. Srisuresh, "Definitions of Managed Objects for Middlebox Communication", RFC 5190, March 2008.

[RFC5382] Guha, S., Biswas, K., Ford, B., Sivakumar, S., and P. Srisuresh, "NAT Behavioral Requirements for TCP", BCP 142, RFC 5382, October 2008.

[RFC6062] Perreault, S. and J. Rosenberg, "Traversal Using Relays around NAT (TURN) Extensions for TCP Allocations", RFC 6062, November 2010.

[RFC6081] Thaler, D., "Teredo Extensions", RFC 6081, January 2011.

[RFC6156] Camarillo, G., Novo, O., and S. Perreault, "Traversal Using Relays around NAT (TURN) Extension for IPv6", RFC 6156, April 2011.

[UPnP-IGD] Warrier, U., Iyer, P., Pennerath, F., Marynissen, G., Schmitz, M., Siddiqi, W., and M. Blaszczak, "Internet Gateway Device (IGD) Standardized Device Control Protocol V 1.0", November 2001.

## Appendix A. Limitations of ICE TCP

Compared to UDP-based ICE, ICE TCP has, in general, lower success probability for enabling connectivity without a relay if both of the hosts are behind a NAT. This happens because many of the currently deployed NATs have endpoint-dependent mapping behavior, or they do not support the flow of TCP handshake packets seen in the case of TCP simultaneous-open, e.g., some NATs do not allow incoming TCP SYN packets from an address where a SYN packet has been sent to recently or the subsequent SYN-ACK is not processed properly.

It has been reported in [IMC05] that with the population of NATs deployed at the time of the measurements (2005), one of the NAT traversal techniques described here, TCP simultaneous-open, worked in roughly 45% of the cases. Also, not all operating systems implement TCP simultaneous-open properly and thus are not able to use such candidates. However, when more NATs comply with the requirements set by [RFC5382] and operating system TCP stacks are fixed, the success probability of simultaneous-open is likely to increase. Also, it is important to implement additional techniques with higher success ratios, such as Teredo, whose success in different scenarios is described in Figure 1 of [RFC6081].

Finally, it should be noted that implementing various techniques listed in Section 5 should increase the success probability, but many of these techniques require support from the endpoints and/or from some network elements (e.g., from the NATs). Without comprehensive experimental data on how well different techniques are supported, the actual increase of success probability is hard to evaluate.



## Appendix B. Implementation Considerations for BSD Sockets

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

In particular, ICE requires an agent to obtain a local TCP candidate, bound to a local IP and port, then initiate a TCP connection from that local port (e.g., 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.

## Appendix C. SDP Examples

This section shows two examples of SDP offer and answer when the ICE TCP extension is used. Both examples are based on the simplified topology of Figure 8 in [RFC5245], with the same IP addresses. The examples shown here should be considered strictly informative.

In the first example, the offer contains only TCP candidates (lines are folded in examples to satisfy RFC formatting rules):

```
v=0
o=jdoe 2890844526 2890842807 IN IP4 10.0.1.1
s=
c=IN IP4 192.0.2.3
t=0 0
a=ice-pwd:asd88fgpdd777uzjYhagZg
a=ice-ufrag:8hhY
m=audio 45664 TCP/RTP/AVP 0
b=RS:0
b=RR:0
a=rtpmap:0 PCMU/8000
a=setup:active
a=connection:new
a=candidate:1 1 TCP 2128609279 10.0.1.1 9 typ host tcptype active
a=candidate:2 1 TCP 2124414975 10.0.1.1 8998 typ host tcptype passive
a=candidate:3 1 TCP 2120220671 10.0.1.1 8999 typ host tcptype so
a=candidate:4 1 TCP 1688207359 192.0.2.3 9 typ srflx raddr 10.0.1.1
  rport 9 tcptype active
a=candidate:5 1 TCP 1684013055 192.0.2.3 45664 typ srflx raddr
  10.0.1.1 rport 8998 tcptype passive
a=candidate:6 1 TCP 1692401663 192.0.2.3 45687 typ srflx raddr
  10.0.1.1 rport 8999 tcptype so
```

The answer to that offer could look like this:

```
v=0
o=bob 2808844564 2808844564 IN IP4 192.0.2.1
s=
c=IN IP4 192.0.2.1
t=0 0
a=ice-pwd:YH75Fviy6338Vbrhrlp8Yh
a=ice-ufrag:9uB6
m=audio 3478 TCP/RTP/AVP 0
b=RS:0
b=RR:0
a=setup:passive
a=connection:new
a=rtpmap:0 PCMU/8000
a=candidate:1 1 TCP 2128609279 192.0.2.1 9 typ host tcptype active
a=candidate:2 1 TCP 2124414975 192.0.2.1 3478 typ host tcptype passive
a=candidate:3 1 TCP 2120220671 192.0.2.1 3482 typ host tcptype so
```

In the second example, UDP and TCP media streams are mixed, but S-O candidates are omitted due to hosts not supporting TCP simultaneous-open, and UDP candidates are preferred (but preference order for candidate types is kept the same) by decreasing the TCP candidate type preferences by one (i.e., using type preference 125 for the host candidates and 99 for the server reflexive candidates):

```
v=0
o=jdoe 2890844526 2890842807 IN IP4 10.0.1.1
s=
c=IN IP4 192.0.2.3
t=0 0
a=ice-pwd:asd88fgpdd777uzjYhagZg
a=ice-ufrag:8hhY
m=audio 45664 RTP/AVP 0
b=RS:0
b=RR:0
a=rtpmap:0 PCMU/8000
a=candidate:1 1 TCP 2111832063 10.0.1.1 9 typ host tcptype active
a=candidate:2 1 TCP 2107637759 10.0.1.1 9012 typ host tcptype passive
a=candidate:3 1 TCP 1671430143 192.0.2.3 9 typ srflx raddr 10.0.1.1
  rport 9 tcptype active
a=candidate:4 1 TCP 1667235839 192.0.2.3 44642 typ srflx raddr
  10.0.1.1 rport 9012 tcptype passive
a=candidate:5 1 UDP 2130706431 10.0.1.1 8998 typ host
a=candidate:6 1 UDP 1694498815 192.0.2.3 45664 typ srflx raddr
  10.0.1.1 rport 8998
```

The corresponding answer could look like this:

```
v=0
o=bob 2808844564 2808844564 IN IP4 192.0.2.1
s=
c=IN IP4 192.0.2.1
t=0 0
a=ice-pwd:YH75Fviy6338Vbrhrlp8Yh
a=ice-ufrag:9uB6
m=audio 3478 RTP/AVP 0
b=RS:0
b=RR:0
a=rtpmap:0 PCMU/8000
a=candidate:1 1 TCP 2111832063 192.0.2.1 9 typ host tcptype active
a=candidate:2 1 TCP 2107637759 192.0.2.1 3478 typ host tcptype passive
a=candidate:3 1 UDP 2130706431 192.0.2.1 3478 typ host
```

## Authors' Addresses

Jonathan Rosenberg
jdrosen.net

EMail: jdrosen@jdrosen.net
URI: http://www.jdrosen.net

Ari Keranen
Ericsson
Hirsalantie 11
02420 Jorvas
Finland

EMail: ari.keranen@ericsson.com

Bruce B. Lowekamp
Skype

EMail: bbl@lowekamp.net

Adam Roach
Tekelec
17210 Campbell Rd., Suite 250
Dallas, TX 75252
US

EMail: adam@nostrum.com