MMUSIC WG
Internet-Draft
Expires: April 26, 2009

A. B. Roach
Tekelec
B. B. Lowekamp
SIPeerior Technologies
October 23, 2008

A Proposal to Define Interactive Connectivity Establishment for the Transport Control Protocol (ICE-TCP) as an Extensible Framework
draft-lowekamp-mmusic-ice-tcp-framework-00

Abstract

The ICE-TCP mechanism is currently regarded as of limited usefulness due to the low success rate of TCP simultaneous open for NAT traversal. This document presents a vision of the ICE-TCP document as an extensible framework for negotiating a variety of approaches for establishing a TCP connection between NATed hosts. This document further proposes significantly extending the current set of

collection mechanisms to encompass a wide variety of technologies that are currently available, including UPnP, SOCKS, and Teredo. Because several of these technologies are already widely deployed, the direct connection rate should be significantly higher than using straight TCP alone. We envision that as future TCP connection establishment techniques are developed, they too will specify an ICE encoding that will allow their negotiation.

Table of Contents

1. Introduction

The ICE-TCP document [15] currently relies on a closed set of technologies for gathering candidates. While there is no prohibition on the use of alternate technologies, ICE-TCP limits its discussion to those technologies discussed in the base ICE specification [14]. Specifically, this approach discusses the use of host candidates, server reflexive candidates, and relayed candidates (with a focus on TURN).

Unfortunately, this focus has led to the impression that ICE-TCP must either use relayed candidates or rely on the "simultaneous open" approach that is known to have a low chance of success. In fact, both ICE and ICE-TCP can be extended to leverage any of a myriad of NAT traversal technologies.

The most appealing feature of these technologies is that many of them are already widely deployed. For example:

Teredo: Teredo establishes a UDP tunnel for other transport protocols that is visible to applications on a host as an IPv6 address. It is included in all current distributions of Windows and available for Mac OS X, Linux, and most BSD platforms as a freely installable package.

UPnP: deployed on the majority of residential-grade NAT/Firewall devices and allows hosts behind the NAT to request a publicly accessible TCP port.

SOCKS: Widely available as a relaying protocol, it has also been extended to act as a NAT traversal solution in many NATs.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [2].

3. Proposal

The authors propose that the ICE-TCP document be modified and expanded to clarify the way that candidates are gathered and prioritized.

3.1. Gathering Candidates

The current version of ICE-TCP discusses the use of STUN and TURN for gathering Server Reflexive and Relayed candidates, respectively. We propose this be written in a way that clarifies that such candidates can be gathered via myriad mechanisms, and gives advice on which types of candidates to gather.

To that end, we propose to replace the following text in section 3.1:

> 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.

With:

> 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 two Relayed Candidates (see Section 4.3).
>
> The agent should then obtain one or more non-relayed NAT candidates (see Section 4.2). The mechanisms for establishing such candidates and the number of candidates to collect vary from technique to technique. These considerations are discussed in the relevant sections, below.

> Once the relayed candidates and non-relayed NAT candidates have been prepared, the agent MUST keep the TCP connection open until ICE processing has completed. See Section 1 for important implementation guidelines.

(Note that, in the preceding text, references to Section 4.3 and Section 4.2 refer to sections in this document, not to sections in draft-ietf-mmusic-ice-tcp.)

3.2. Prioritization

The current prioritization scheme defined in ICE-TCP favors simultaneous-open candidates over active and passive candidates. This prioritization is presumably based on the prospect that non-relayed connections are the exclusive domain of STUN-discovered Server Reflexive Candidates. Such candidates necessarily rely on "fooling" the NAT into allowing TCP connections through; and one might assume that simultaneous open has a higher chance of succeeding in doing so.

Empirical evidence on the simultaneous open technique described in ICE-TCP has shown that, while it has a relatively high chance of establishing the proper state in a NAT, it suffers from a high failure rate on the actual endpoints.

Several NAT traversal techniques, both deployed and proposed, provide means for discovering NAT-allocated address/port combinations in such a way that the NAT is actively participating in the TCP establishment effort instead of impeding it. Others leverage the behavior of UDP binding in NATs to carry TCP traffic over UDP. In such cases, normal active and passive candidates actually have a higher chance of success than simultaneous-open candidates.

To reflect this reality, we propose that the prioritization scheme for ICE-TCP be revised. Specifically, we propose to replace the following text in section 3.2:

> 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.

With:

> It is RECOMMENDED that, for all connection-oriented media, candidates have a direction-pref assigned as follows:

7 NAT-Assisted Active Candidate
6 NAT-Assisted Passive Candidate
5 UDP-Tunneled Active Candidate
4 UDP-Tunneled Passive Candidate
3 Simultaneous Open Candidate
2 Non-NAT-Assisted Active Candidate
1 Non-NAT-Assisted Passive Candidate

It is RECOMMENDED that the type preference for NAT-Assisted candidates be set higher than that for server-reflexive candidates and that the type preference for UDP-Tunneled candidates be set lower than that for server-reflexive candidates. The RECOMMENDED values are 105 for NAT-Assisted candidates and 75 for UDP-Tunneled candidates.

TODO: The same information probably doesn't need to be encoded in both the type-pref and direction-pref. More work is needed to iron out how to represent appropriate priorities.

4. Initial Set of Candidate Collection Technologies

(The authors propose that the entirety of this Section 4 and its subsections, with the exception of this parenthetical paragraph, be included in ICE-TCP.)

The following sections discuss a number of techniques that can be used to obtain candidates for use with ICE-TCP. It is critical to note that this list is not intended to be exhaustive, nor is implementation of any specific technique 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.

4.1. Host 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.

For maximum interoperability with the techniques described below, 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 Teredo (Section 4.2.2.1) or SOCKS IPv4-IPv6 Gatewaying (Section 4.3.2).

4.2. Non-Relayed NAT Candidates

The following techniques can be used to gather candidates that represent NAT traversal, while not going through any additional relays. This includes Server Reflexive Candidates (non-NAT-assisted), candidates established in cooperation with the NAT (NAT-assisted), and candidates tunnel TCP over UDP to leverage widespread NAT UDP binding behavior (UDP-tunneling).

Generally, when several options are available, clients should favor NAT-assisted techniques over UDP-tunneling techniques, and UDP-tunneling techniques over non-NAT-assisted techniques.

4.2.1. NAT-Assisted

To traverse NATs, the best approach is to work with the NATs themselves, rather than trying to "game" their behavior with tricks and relays. To that end, clients behind NATs should favor approaches that work with NATs whenever possible.

Because these techniques interact with the NAT directly to acquire a publicly accessible transport address, once obtained these candidates are encoded as normally TCP candidates (typically tcp-pass) as specified in Section 3.4 of ICE-TCP.

4.2.1.1. UPnP IGD

The UPnP forum's Internet Gateway Device (IGD) protocol [19] is designed to facilitate client configuration of NAT port forwarding behavior. IGD is deployed on a majority of residential-grade NAT/Firewall devices, and is available for Linux- and FreeBSD-based firewalls.

Clients wishing to use IGD-obtained addresses as candidates do so by retrieving the ExternalIPAddress state variable; then, they use the AddPortMapping command to establish a new TCP binding at the NAT. The client is responsible for establishing the binding so that it corresponds to a Host Candidate, and for periodically refreshing the port mapping to keep the lease from expiring. When the IGD-acquired candidate is no longer necessary, the client SHOULD remove the binding with a DeletePortMapping command.

4.2.1.2. MIDCOM SNMP

The MIDCOM MIB [12] defines an SNMP-based mechanism for controlling

NATs, Firewalls, and other middleboxes.

TODO: add application notes about how to obtain candidates

4.2.1.3. SOCKS

Although originally designed as a relaying protocol, SOCKS [1] has been incorporated in a number of NATs as a NAT-assisted traversal technique. The approach for using SOCKS for NAT-assisted traversal is identical to that for using it as a relay protocol (see Section 4.3.1).

If the ICE agent is aware that SOCKS is being used as a NAT-assisted protocol instead of a relay protocol, it SHOULD set the local-preference accordingly.

4.2.1.4. RSIP

The Realm Specific IP (RSIP) protocol [4] is an experimental protocol designed to allow clients within a realm to communicate with gateways on the edge of that realm so as to lease globally-visible resources on those gateways.

TODO: add application notes about how to obtain candidates

TODO: examine RSIP as a v4/v6 bridging technology

4.2.1.5. SIMCO

The SIMCO protocol [11] an experimental mechanism for controlling NATs, Firewalls, and other middleboxes.

TODO: add application notes about how to obtain candidates

4.2.1.6. NAT-PMP

The NAT Port Mapping Protocol (PMP) [18] is designed to allow clients to determine the external IP address of a NAT, learn about any changes in that IP address, and create and refresh UDP and TCP bindings on the NAT. NAT-PMP is currently supported in a number of field-deployed products, such as the Apple Airport Express, Apple Airport Extreme, and Apple Time Capsule, as well as a large number of primarily peer-to-peer software applications.

Clients wishing to use PMP-obtained addresses as candidates do so by retrieving the external IP address, using the PMP opcode 0; then, they use the PMP opcode 2 to establish a new TCP binding at the NAT. The client is responsible for establishing the binding so that it

corresponds to a Host Candidate, and for periodically refreshing the port mapping to keep the lease from expiring. When the PMP-obtained candidate is no longer necessary, the client SHOULD remove the binding with a PMP opcode 2 with the port mapping lifetime set to 0.

4.2.2. UDP Tunneled

4.2.2.1. Teredo

The Teredo protocol [10] defines a system allow nodes behind one or more NATs to obtain IPv6 addresses by tunneling IPv6 over UDP. Teredo it included in all modern Windows operating systems by default, and is available for most other major operating systems, such as Linux, OS X, and *BSD.

Teredo essentially provides a UDP tunnel for other transport protocols that is visible to the host application as an IPv6 address. Therefore, Teredo candidates are encoded as IPv6 addresses in the SDP.

The Teredo framework includes provisions for routing between Teredo IPv6 addresses and native IPv6 addresses; therefore, the efficacy of Teredo tunneling will be significantly improved for each ICE-TCP implementation that advertises at least one globally-routable IPv6 address candidate (whether Teredo, SOCKS tunneled, 6-to-4 relayed, IPv6 tunneled, or native).

TODO: add application notes about how to obtain candidates

4.2.2.2. TCP over UDP

TODO: Describe TCP/UDP/IP, as defined in [17].

TODO: add application notes about how to obtain candidates; need to include discussion of SDP extensions necessary to specify encoding for TCP over UDP.

4.2.3. Non-NAT-Assisted

4.2.3.1. STUN

TODO: Describe STUN, as defined in [13].

To obtain STUN server reflexive candidates, the agent initiates a TCP connection from two host candidates to a STUN server, and uses a Binding request over that connection to learn the server reflexive candidate corresponding to that host candidate.

4.3. Relayed Candidates

4.3.1. SOCKS

TODO: Describe SOCKS, as defined in [1]

TODO: add application notes about how to obtain candidates

4.3.2. SOCKS IPv4-IPv6 Gateway

TODO: Describe IPv4/IPv6 bridging technique described in [3]

TODO: add application notes about how to obtain candidates

4.3.3. SSH Tunnels

TODO: Describe SSH Tunneling technique described in [5] [6] [7] [8] [9]

TODO: add application notes about how to obtain candidates

4.3.4. TURN TCP

TODO: Describe TURN TCP protocol described in [16]

To acquire TURN TCP candidates, the agent initiates a TCP connection from two host candidates 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.

5. References

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

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

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

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

[5] Lehtinen, S. and C. Lonvick, "The Secure Shell (SSH) Protocol

Assigned Numbers", RFC 4250, January 2006.

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

[7] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Authentication Protocol", RFC 4252, January 2006.

[8] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, January 2006.

[9] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Connection Protocol", RFC 4254, January 2006.

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

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

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

[13] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for (NAT) (STUN)", draft-ietf-behave-rfc3489bis-16 (work in progress), July 2008.

[14] 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.

[15] Rosenberg, J., "TCP Candidates with Interactive Connectivity Establishment (ICE)", draft-ietf-mmusic-ice-tcp-07 (work in progress), July 2008.

[16] Rosenberg, J. and R. Mahy, "Traversal Using Relays around NAT (TURN) Extensions for TCP Allocations", draft-ietf-behave-turn-tcp-00 (work in progress), November 2007.

[17] Denis-Courmont, R., "UDP-Encapsulated Transport Protocols", draft-denis-udp-transport-00 (work in progress), July 2008.

[18] Cheshire, S., "NAT Port Mapping Protocol (NAT-PMP)", draft-cheshire-nat-pmp-03 (work in progress), April 2008.

[19] 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.

Authors' Addresses

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

Email: adam@nostrum.com

Bruce B. Lowekamp
SIPeerior Technologies
5251-18 John Tyler Highway #330
Williamsburg, VA 23185
USA

Phone: +1 757 565 0101
Email: bbl@lowekamp.net

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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