Stream:	Internet Engineering Task Force (IETF)
RFC:	9028
Category:	Experimental
Published:	June 2021
ISSN:	2070-1721
Authors:	A. Keränen	J. Melén	M. Komu, Ed.
---	---	---	---
	Ericsson	*Ericsson*	*Ericsson*

RFC 9028
Native NAT Traversal Mode for the Host Identity Protocol

Abstract

This document specifies a new Network Address Translator (NAT) traversal mode for the Host Identity Protocol (HIP). The new mode is based on the Interactive Connectivity Establishment (ICE) methodology and UDP encapsulation of data and signaling traffic. The main difference from the previously specified modes is the use of HIP messages instead of ICE for all NAT traversal procedures due to the kernel-space dependencies of HIP.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for examination, experimental implementation, and evaluation.

This document defines an Experimental Protocol for the Internet community. 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). Not all documents approved by the IESG are candidates for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9028.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions

with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

The Host Identity Protocol (HIP) [RFC7401] is specified to run directly on top of IPv4 or IPv6. However, many middleboxes found in the Internet, such as NATs and firewalls, often allow only UDP or TCP traffic to pass [RFC5207]. Also, NATs usually require the host behind a NAT to create a forwarding state in the NAT before other hosts outside of the NAT can contact the host behind the NAT. To overcome this problem, different methods, commonly referred to as NAT traversal techniques, have been developed.

As one solution, the HIP experiment report [RFC6538] mentions Teredo-based NAT traversal for HIP and related Encapsulating Security Payload (ESP) traffic (with double tunneling overhead). Another solution is specified in [RFC5770], which will be referred to as "Legacy ICE-HIP" in this document. The experimental Legacy ICE-HIP specification combines the Interactive Connectivity Establishment (ICE) protocol (originally [RFC5245]) with HIP so that basically, ICE is responsible for NAT traversal and connectivity testing, while HIP is responsible for end-host authentication and IPsec key management. The resulting protocol uses HIP, Session Traversal Utilities for NAT (STUN), and ESP messages tunneled over a single UDP flow. The benefit of using ICE and its STUN / Traversal Using Relays around NAT (TURN) messaging formats is that one can reuse the NAT traversal infrastructure already available in the Internet, such as STUN and TURN servers. Also, some middleboxes may be STUN aware and may be able to do something "smart" when they see STUN being used for NAT traversal.

HIP poses a unique challenge to using standard ICE, not only due to kernel-space dependencies of HIP, but also due to its close integration with kernel-space IPsec; and, while [RFC5770] provides a technically workable path, HIP incurs unacceptable performance drawbacks for kernel-space implementations. Also, implementing and integrating a full ICE/STUN/TURN protocol stack as specified in Legacy ICE-HIP results in a considerable amount of effort and code, which could be avoided by reusing and extending HIP messages and state machines for the same purpose. Thus, this document specifies an alternative NAT traversal mode referred to as "Native ICE-HIP" that employs the HIP messaging format instead of STUN or TURN for the connectivity checks, keepalives, and data relaying. Native ICE-HIP also specifies how mobility management works in

the context of NAT traversal, which is missing from the Legacy ICE-HIP specification. The native specification is also based on HIPv2, whereas the legacy specification is based on HIPv1. The differences to Legacy ICE-HIP are further elaborated in Appendix B.

Similar to Legacy ICE-HIP, this specification builds on the HIP registration extensions [RFC8003] and the base exchange procedure [RFC7401] and its closing procedures; therefore, the reader is recommended to get familiar with the relevant specifications. In a nutshell, the registration extensions allow a HIP Initiator (usually a "client" host) to ask for specific services from a HIP Responder (usually a "server" host). The registration parameters are included in a base exchange, which is essentially a four-way Diffie-Hellman key exchange authenticated using the public keys of the end hosts. When the hosts negotiate support for ESP [RFC7402] during the base exchange, they can deliver ESP-protected application payload to each other. When either of the hosts moves and changes its IP address, the two hosts re-establish connectivity using the mobility extensions [RFC8046]. The reader is also recommended to get familiar with the mobility extensions; basically, the process is a three-way procedure where the mobile host first announces its new location to the peer; then, the peer tests for connectivity (the so-called return routability check); and then, the mobile host must respond to the announcement in order to activate its new location. This specification builds on the mobility procedures, but modifies them to be compatible with ICE. The differences in the mobility extensions are specified in Appendix C. It is worth noting that multihoming support as specified in [RFC8047] is left for further study.

This specification builds heavily on the ICE methodology, so it is recommended that the reader is familiar with the ICE specification [RFC8445] (especially the overview). However, Native ICE-HIP does not implement all the features in ICE; hence, the different features of ICE are cross referenced using [RFC2119] terminology for clarity. Appendix B explains the differences to ICE, and it is recommended that the reader read this section in addition to the ICE specification.

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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document borrows terminology from [RFC5770], [RFC7401], [RFC8046], [RFC9068], [RFC8445], and [RFC8489]. The following terms recur in the text:

ICE:
: Interactive Connectivity Establishment (ICE) protocol as specified in [RFC8445].

Legacy ICE-HIP:
: Refers to the "Basic Host Identity Protocol (HIP) Extensions for Traversal of Network Address Translators" as specified in [RFC5770]. The protocol specified in this document offers an alternative to Legacy ICE-HIP.

Native ICE-HIP:
: The protocol specified in this document (Native NAT Traversal Mode for HIP).

Initiator:
: The host that initiates the base exchange using I1 message [RFC7401].

Responder:
: The host that receives the I1 packet from the Initiator [RFC7401].

Control Relay Server
: A registrar host that forwards any kind of HIP control plane packets between the Initiator and the Responder. This host is critical because it relays the locators between the Initiator and the Responder so that they can try to establish a direct communication path with each other. This host is used to replace HIP Rendezvous Servers [RFC8004] for hosts operating in private address realms. In the Legacy ICE-HIP specification [RFC5770], this host is denoted as "HIP Relay Server".

Control Relay Client:
: A requester host that registers to a Control Relay Server requesting it to forward control plane traffic (i.e., HIP control messages). In the Legacy ICE-HIP specification [RFC5770], this is denoted as "HIP Relay Client".

Data Relay Server:
: A new entity introduced in this document; a registrar host that forwards HIP related data plane packets, such as Encapsulating Security Payload (ESP) [RFC7402], between two hosts. This host implements similar functionality as TURN servers.

Data Relay Client:
: A requester host that registers to a Data Relay Server requesting it to forward data plane traffic (e.g. ESP traffic). This functionality is a new and introduced in this document.

Locator:
: As defined in [RFC8046]: "A name that controls how the packet is routed through the network and demultiplexed by the end host. It may include a concatenation of traditional network addresses such as an IPv6 address and end-to-end identifiers such as an ESP SPI. It may also include transport port numbers or IPv6 Flow Labels as demultiplexing context, or it may simply be a network address."

LOCATOR_SET (written in capital letters):
: Denotes a HIP control packet parameter that bundles multiple locators together [RFC8046].

HIP offer:
: Before two end hosts can establish a communication channel using the NAT traversal procedures defined in this document, they need to exchange their locators (i.e., candidates) with each other. In ICE, this procedure is called Candidate Exchange; it does not specify how the candidates are exchanged, but Session Description Protocol (SDP) "offer/answer" is mentioned as an example. In contrast, the Candidate Exchange in HIP is the base exchange

itself or a subsequent UPDATE procedure occurring after a handover. Following [RFC5770] and SDP-related naming conventions [RFC3264], "HIP offer" is the Initiator's LOCATOR_SET parameter in a HIP I2 or in an UPDATE control packet.

HIP answer:
The Responder's LOCATOR_SET parameter in a HIP R2 or UPDATE control packet. The HIP answer corresponds to the SDP answer parameter [RFC3264] but is HIP specific. Please refer also to the longer description of the "HIP offer" term above.

HIP connectivity checks:
In order to obtain a direct end-to-end communication path (without employing a Data Relay Server), two communicating HIP hosts try to "punch holes" through their NAT boxes using this mechanism. It is similar to the ICE connectivity checks but implemented using HIP return routability checks.

Controlling host:
The controlling host [RFC8445] is always the Initiator in the context of this specification. It nominates the candidate pair to be used with the controlled host.

Controlled host:
The controlled host [RFC8445] is always the Responder in the context of this specification. It waits for the controlling host to nominate an address candidate pair.

Checklist:
A list of address candidate pairs that need to be tested for connectivity (same as in [RFC8445]).

Transport address:
Transport-layer port and the corresponding IPv4/v6 address (same as in [RFC8445]).

Candidate:
A transport address that is a potential point of contact for receiving data (same as in [RFC8445]).

Host candidate:
A candidate obtained by binding to a specific port from an IP address on the host (same as in [RFC8445]).

Server-reflexive candidate:
A translated transport address of a host as observed by a Control or Data Relay Server (same as in [RFC8445]).

Peer-reflexive candidate:
A translated transport address of a host as observed by its peer (same as in [RFC8445]).

Relayed candidate:
A transport address that exists on a Data Relay Server. Packets that arrive at this address are relayed towards the Data Relay Client. The concept is the same as in [RFC8445], but a Data Relay Server is used instead of a TURN server.

Permission:
: In the context of Data Relay Server, permission refers to a concept similar to TURN's [RFC8656] channels. Before a host can use a relayed candidate to forward traffic through a Data Relay Server, the host must activate the relayed candidate with a specific peer host.

Base:
: Similar to that described in [RFC8445], the base of a candidate is the local source address a host uses to send packets for the associated candidate. For example, the base of a server-reflexive address is the local address the host used for registering itself to the associated Control or Data Relay Server. The base of a host candidate is equal to the host candidate itself.

3. Overview of Operation

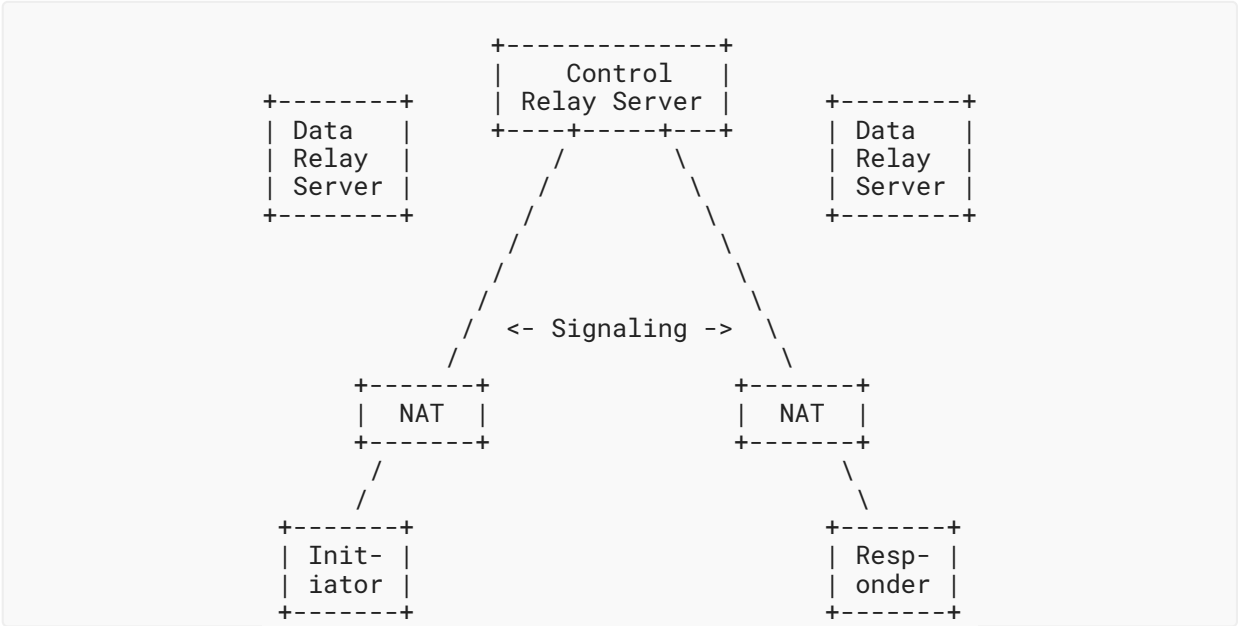

Figure 1: Example Network Configuration

In the example configuration depicted in Figure 1, both Initiator and Responder are behind one or more NATs, and both private networks are connected to the public Internet. To be contacted from behind a NAT, at least the Responder must be registered with a Control Relay Server reachable on the public Internet. The Responder may have also registered to a Data Relay Server that can forward the data plane in case NAT traversal fails. While, strictly speaking, the Initiator does not need a Data Relay Server, it may act in the other role with other hosts; connectivity with the Data Relay Server of the Responder may fail, so the Initiator may also need to register to a Control and/or Data Relay Server. It is worth noting that a Control and Data Relay does not forge the source address of a passing packet but always translates the source address and source port of a packet to be forwarded (to its own).

We assume, as a starting point, that the Initiator knows both the Responder's Host Identity Tag (HIT) and the address(es) of the Responder's Control Relay Server(s) (how the Initiator learns of the Responder's Control Relay Server is outside of the scope of this document, but it may be learned through DNS or another name service). The first steps are for both the Initiator and Responder to register with a Control Relay Server (need not be the same one) and gather a set of address candidates. The hosts use either Control Relay Servers or Data Relay Servers for gathering the candidates. Next, the HIP base exchange is carried out by encapsulating the HIP control packets in UDP datagrams and sending them through the Responder's Control Relay Server. As part of the base exchange, each HIP host learns of the peer's candidate addresses through the HIP offer/answer procedure embedded in the base exchange.

Once the base exchange is completed, two HIP hosts have established a working communication session (for signaling) via a Control Relay Server, but the hosts still have to find a better path, preferably without a Data Relay Server, for the ESP data flow. For this, connectivity checks are carried out until a working pair of addresses is discovered. At the end of the procedure, if successful, the hosts will have established a UDP-based tunnel that traverses both NATs with the data flowing directly from NAT to NAT or via a Data Relay Server. At this point, the HIP signaling can also be sent over the same address/port pair, and is demultiplexed (or, in other words, separated) from IPsec as described in the UDP encapsulation standard for IPsec [RFC3948]. Finally, the two hosts send NAT keepalives as needed in order keep their UDP-tunnel state active in the associated NAT boxes.

If either one of the hosts knows that it is not behind a NAT, hosts can negotiate during the base exchange a different mode of NAT traversal that does not use HIP connectivity checks, but only UDP encapsulation of HIP and ESP. Also, it is possible for the Initiator to simultaneously try a base exchange with and without UDP encapsulation. If a base exchange without UDP encapsulation succeeds, no HIP connectivity checks or UDP encapsulation of ESP are needed.

4. Protocol Description

This section describes the normative behavior of the "Native ICE-HIP" protocol extension. Most of the procedures are similar to what is defined in [RFC5770] but with different, or additional, parameter types and values. In addition, a new type of relaying server, Data Relay Server, is specified. Also, it should be noted that HIP version 2 [RFC7401] **MUST** be used instead of HIPv1 with this NAT traversal mode.

4.1. Relay Registration

In order for two hosts to communicate over NATed environments, they need a reliable way to exchange information. To achieve this, "HIP Relay Server" is defined in [RFC5770]. It supports the relaying of HIP control plane traffic over UDP in NATed environments and forwards HIP control packets between the Initiator and the Responder. In this document, the HIP Relay Server is denoted as "Control Relay Server" for better alignment with the rest of the terminology. The registration to the Control Relay Server can be achieved using the RELAY_UDP_HIP parameter as explained later in this section.

To also guarantee data plane delivery over varying types of NAT devices, a host **MAY** also register for UDP-encapsulated ESP relaying using Registration Type RELAY_UDP_ESP (value 3). This service may be coupled with the Control Relay Server or offered separately on another server. If the server supports relaying of UDP-encapsulated ESP, the host is allowed to register for a data-relaying service using the registration extensions in Section 3.3 of [RFC8003]. If the server has sufficient relaying resources (free port numbers, bandwidth, etc.) available, it opens a UDP port on one of its addresses and signals the address and port to the registering host using the RELAYED_ADDRESS parameter (as defined in Section 5.12 in this document). If the Data Relay Server would accept the data-relaying request but does not currently have enough resources to provide data-relaying service, it **MUST** reject the request with Failure Type "Insufficient resources" [RFC8003].

The registration process follows the generic registration extensions defined in [RFC8003]. The HIP control plane relaying registration follows [RFC5770], but the data plane registration is different. It is worth noting that if the HIP control and data plane relay services reside on different hosts, the client has to register separately to each of them. In the example shown in Figure 2, the two services are coupled on a single host. The text uses "Relay Client" and "Relay Server" as a shorthand when the procedures apply both to control and data cases.

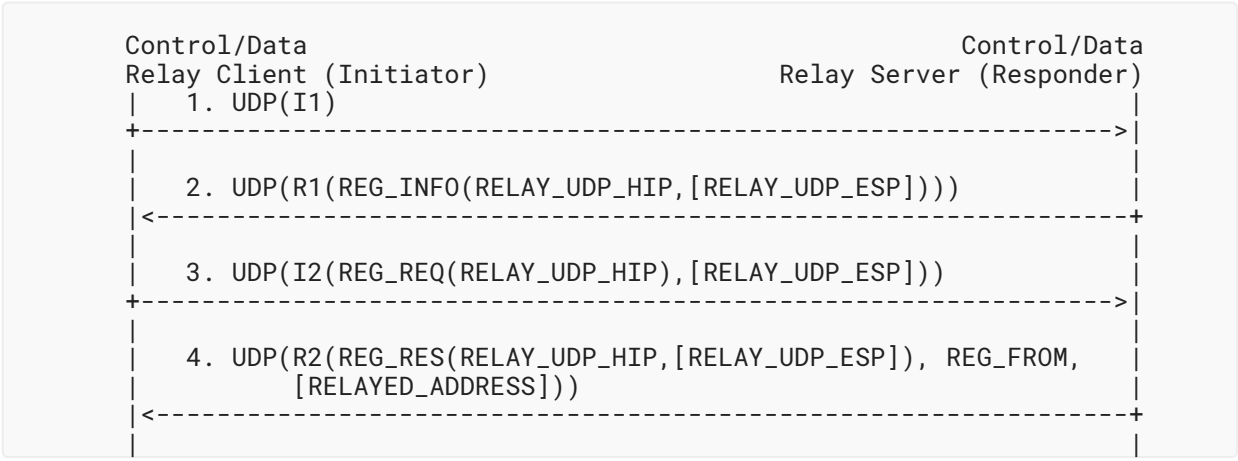

Figure 2: Example Registration with a HIP Relay

In step 1, the Relay Client (Initiator) starts the registration procedure by sending an I1 packet over UDP to the Relay Server. It is **RECOMMENDED** that the Relay Client select a random source port number from the ephemeral port range 49152-65535 for initiating a base exchange. Alternatively, a host **MAY** also use a single fixed port for initiating all outgoing connections. However, the allocated port **MUST** be maintained until all of the corresponding HIP associations are closed. It is **RECOMMENDED** that the Relay Server listen to incoming connections at UDP port 10500. If some other port number is used, it needs to be known by potential Relay Clients.

In step 2, the Relay Server (Responder) lists the services that it supports in the R1 packet. The support for HIP control plane over UDP relaying is denoted by the Registration Type value RELAY_UDP_HIP (see Section 5.9). If the server also supports the relaying of ESP traffic over UDP, it also includes the Registration Type value RELAY_UDP_ESP.

In step 3, the Relay Client selects the services for which it registers and lists them in the REG_REQ parameter. The Relay Client registers for the Control Relay service by listing the RELAY_UDP_HIP value in the request parameter. If the Relay Client also requires ESP relaying over UDP, it lists also RELAY_UDP_ESP.

In step 4, the Relay Server concludes the registration procedure with an R2 packet and acknowledges the registered services in the REG_RES parameter. The Relay Server denotes unsuccessful registrations (if any) in the REG_FAILED parameter of R2. The Relay Server also includes a REG_FROM parameter that contains the transport address of the Relay Client as observed by the Relay Server (server-reflexive candidate). If the Relay Client registered to ESP-relaying service, the Relay Server includes a RELAYED_ADDRESS parameter that describes the UDP port allocated to the Relay Client for ESP relaying. It is worth noting that the Data Relay Client must first activate this UDP port by sending an UPDATE message to the Data Relay Server that includes a PEER_PERMISSION parameter as described in Section 4.12.1 both after base exchange and handover procedures. Also, the Data Relay Server should follow the port allocation recommendations in Section 7.5.

After the registration, the Relay Client periodically sends NAT keepalives to the Relay Server in order to keep the NAT bindings between the Relay Client and the relay alive. The keepalive extensions are described in Section 4.10.

The Data Relay Client **MUST** maintain an active HIP association with the Data Relay Server as long as it requires the data-relaying service. When the HIP association is closed (or times out), or the registration lifetime passes without the Data Relay Client refreshing the registration, the Data Relay Server **MUST** stop relaying packets for that host and close the corresponding UDP port (unless other Data Relay Clients are still using it).

The Data Relay Server **SHOULD** offer a different relayed address and port for each Data Relay Client because not doing so can cause problems with stateful firewalls (see Section 7.5).

When a Control Relay Client sends an UPDATE (e.g., due to host movement or to renew service registration), the Control Relay Server **MUST** follow the general guidelines defined in [RFC8003], with the difference that all UPDATE messages are delivered on top of UDP. In addition to this, the Control Relay Server **MUST** include the REG_FROM parameter in all UPDATE responses sent to the Control Relay Client. This applies to both renewals of service registration and to host movement. It is especially important for the case of host movement, as this is the mechanism that allows the Control Relay Client to learn its new server-reflexive address candidate.

A Data Relay Client can request multiple relayed candidates from the Data Relay Server (e.g., for the reasons described in Section 4.12.3). After the base exchange with registration, the Data Relay Client can request additional relayed candidates similarly as during the base exchange. The Data Relay Client sends an UPDATE message REG_REQ parameter requesting for the RELAY_UDP_ESP service. The UPDATE message **MUST** also include a SEQ and an ECHO_REQUEST_SIGNED parameter. The Data Relay Server **MUST** respond with an UPDATE message that includes the corresponding response parameters: REG_RES, ACK and ECHO_REQUEST_SIGNED. In case the Data Relay Server allocated a new relayed UDP port for the Data Relay Client, the REG_RES parameter **MUST** list RELAY_UDP_ESP as a service and the UPDATE message **MUST** also include a

RELAYED_ADDRESS parameter describing the relayed UDP port. The Data Relay Server **MUST** also include the server-reflexive candidate in a REG_FROM parameter. It is worth mentioning that the Data Relay Client **MUST** activate the UDP port as described in Section 4.12.1 before it can be used for any ESP relaying.

A Data Relay Client may unregister a relayed candidate in two ways. It can wait for its lifetime to expire or it can explicitly request it with zero lifetime using the UPDATE mechanism. The Data Relay Client can send a REG_REQ parameter with zero lifetime to the Data Relay Server in order to expire all relayed candidates. To expire a specific relayed candidate, the Data Relay Client **MUST** also include a RELAYED_ADDRESS parameter as sent by the server in the UPDATE message. Upon closing the HIP association (CLOSE-CLOSE-ACK procedure initiated by either party), the Data Relay Server **MUST** also expire all relayed candidates.

Please also refer to Section 7.8 for protection against cross-protocol attacks for both Control Relay Client and Server.

4.2. Transport Address Candidate Gathering at the Relay Client

An Initiator needs to gather a set of address candidates before contacting a (non-relay) Responder. The candidates are needed for connectivity checks that allow two hosts to discover a direct, non-relayed path for communicating with each other. One server-reflexive candidate can be discovered during the registration with the Control Relay Server from the REG_FROM parameter (and another from Data Relay Server if one is employed).

The candidate gathering can be done at any time, but it needs to be done before sending an I2 or R2 in the base exchange if ICE-HIP-UDP mode is to be used for the connectivity checks. It is **RECOMMENDED** that all three types of candidates (host, server reflexive, and relayed) are gathered to maximize the probability of successful NAT traversal. However, if no Data Relay Server is used, and the host has only a single local IP address to use, the host **MAY** use the local address as the only host candidate and the address from the REG_FROM parameter discovered during the Control Relay Server registration as a server-reflexive candidate. In this case, no further candidate gathering is needed.

A Data Relay Client **MAY** register only a single relayed candidate that it uses with multiple other peers. However, it is **RECOMMENDED** that a Data Relay Client registers a new server relayed candidate for each of its peers for the reasons described in Section 4.12.3. The procedures for registering multiple relayed candidates are described in Section 4.1.

If a Relay Client has more than one network interface, it can discover additional server-reflexive candidates by sending UPDATE messages from each of its interfaces to the Relay Server. Each such UPDATE message **MUST** include the following parameters: the registration request (REG_REQ) parameter with Registration Type CANDIDATE_DISCOVERY (value 4) and the ECHO_REQUEST_SIGNED parameter. When a Control Relay Server receives an UPDATE message with registration request containing a CANDIDATE_DISCOVERY type, it **MUST** include a REG_FROM parameter, containing the same information as if this were a Control Relay Server registration, to the response (in addition to the mandatory ECHO_RESPONSE_SIGNED parameter). This request type **SHOULD NOT** create any state at the Control Relay Server.

The rules in Section 5.1.1 of [RFC8445] for candidate gathering are followed here. A number of host candidates (loopback, anycast and others) should be excluded as described in the ICE specification (Section 5.1.1.1 of [RFC8445]). Relayed candidates **SHOULD** be gathered in order to guarantee successful NAT traversal, and implementations **SHOULD** support this functionality even if it will not be used in deployments in order to enable it by software configuration update if needed at some point. Similarly, as explained in the ICE specification (Section 5.1.1.2 of [RFC8445]), if an IPv6-only host is in a network that utilizes NAT64 [RFC6146] and DNS64 [RFC6147] technologies, it may also gather IPv4 server-reflexive and/or relayed candidates from IPv4-only Control or Data Relay Servers. IPv6-only hosts **SHOULD** also utilize IPv6 prefix discovery [RFC7050] to discover the IPv6 prefix used by NAT64 (if any) and generate server-reflexive candidates for each IPv6-only interface, accordingly. The NAT64 server-reflexive candidates are prioritized like IPv4 server-reflexive candidates.

HIP-based connectivity can be utilized by IPv4 applications using Local Scope Identifiers (LSIs) and by IPv6-based applications using HITs. The LSIs and HITs of the local virtual interfaces **MUST** be excluded in the candidate gathering phase as well to avoid creating unnecessary loopback connectivity tests.

Gathering of candidates **MAY** also be performed by other means than described in this section. For example, the candidates could be gathered as specified in Section 4.2 of [RFC5770] if STUN servers are available, or if the host has just a single interface and no STUN or Data Relay Server are available.

Each local address candidate **MUST** be assigned a priority. The following recommended formula (as described in [RFC8445]) **SHOULD** be used:

priority = (2^{24})*(type preference) + (2^{8})*(local preference) + (2^{0})*(256 - component ID)

In the formula, the type preference follows the ICE specification (as defined in Section 5.1.2.1 of [RFC8445]): the **RECOMMENDED** values are 126 for host candidates, 100 for server-reflexive candidates, 110 for peer-reflexive candidates, and 0 for relayed candidates. The highest value is 126 (the most preferred) and lowest is 0 (last resort). For all candidates of the same type, the preference type value **MUST** be identical, and, correspondingly, the value **MUST** be different for different types. For peer-reflexive values, the type preference value **MUST** be higher than for server-reflexive types. It should be noted that peer-reflexive values are learned later during connectivity checks.

Following the ICE specification, the local preference **MUST** be an integer from 0 (lowest preference) to 65535 (highest preference) inclusive. In the case the host has only a single address candidate, the value **SHOULD** be 65535. In the case of multiple candidates, each local preference value **MUST** be unique. Dual-stack considerations for IPv6 also apply here as defined in Section 5.1.2.2 of [RFC8445].

Unlike with SDP used in conjunction with ICE, this protocol only creates a single UDP flow between the two communicating hosts, so only a single component exists. Hence, the component ID value **MUST** always be set to 1.

As defined in Section 14.3 of [RFC8445], the retransmission timeout (RTO) for address gathering from a Control/Data Relay Server **SHOULD** be calculated as follows:

RTO = MAX (1000 ms, Ta * (Num-Of-Cands))

where Ta is the value used for the connectivity check pacing and Num-Of-Cands is the number of server-reflexive and relay candidates. A smaller value than 1000 ms for the RTO **MUST NOT** be used.

4.3. NAT Traversal Mode Negotiation

This section describes the usage of a non-critical parameter type called NAT_TRAVERSAL_MODE with a new mode called ICE-HIP-UDP. The presence of the new mode in the NAT_TRAVERSAL_MODE parameter in a HIP base exchange means that the end host supports NAT traversal extensions described in this document. As the parameter is non-critical (as defined in Section 5.2.1 of [RFC7401]), it can be ignored by an end host, which means that the host is not required to support it or may decline to use it.

With registration with a Control/Data Relay Server, it is usually sufficient to use the UDP-ENCAPSULATION mode of NAT traversal since the Relay Server is assumed to be in public address space. Thus, the Relay Server **SHOULD** propose the UDP-ENCAPSULATION mode as the preferred or only mode. The NAT traversal mode negotiation in a HIP base exchange is illustrated in Figure 3. It is worth noting that the Relay Server could be located between the hosts, but is omitted here for simplicity.

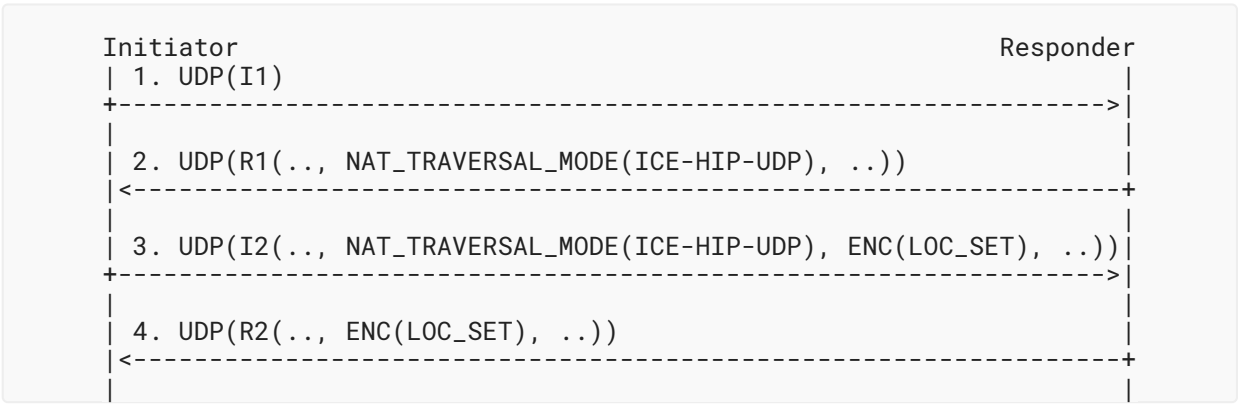

Figure 3: Negotiation of NAT Traversal Mode

In step 1, the Initiator sends an I1 to the Responder.

In step 2, the Responder responds with an R1. As specified in [RFC5770], the NAT_TRAVERSAL_MODE parameter in R1 contains a list of NAT traversal modes the Responder supports. The mode specified in this document is ICE-HIP-UDP (value 3).

In step 3, the Initiator sends an I2 that includes a NAT_TRAVERSAL_MODE parameter. It contains the mode selected by the Initiator from the list of modes offered by the Responder. If ICE-HIP-UDP mode was selected, the I2 also includes the "Transport address" locators (as defined in

Section 5.7) of the Initiator in a LOCATOR_SET parameter (denoted here with LOC_SET). With ICE-HIP-UDP mode, the LOCATOR_SET parameter **MUST** be encapsulated within an ENCRYPTED parameter (denoted here with ENC) according to the procedures in Sections 5.2.18 and 6.5 in [RFC7401]. The locators in I2 are the "HIP offer".

In step 4, the Responder concludes the base exchange with an R2 packet. If the Initiator chose ICE-HIP-UDP traversal mode, the Responder includes a LOCATOR_SET parameter in the R2 packet. With ICE-HIP-UDP mode, the LOCATOR_SET parameter **MUST** be encapsulated within an ENCRYPTED parameter according to the procedures in Sections 5.2.18 and 6.5 in [RFC7401]. The locators in R2, encoded like the locators in I2, are the "ICE answer". If the NAT traversal mode selected by the Initiator is not supported by the Responder, the Responder **SHOULD** reply with a NOTIFY packet with type NO_VALID_NAT_TRAVERSAL_MODE_PARAMETER and abort the base exchange.

4.4. Connectivity Check Pacing Negotiation

As explained in Legacy ICE-HIP [RFC5770], when a NAT traversal mode with connectivity checks is used, new transactions should not be started too fast to avoid congestion and overwhelming the NATs. For this purpose, during the base exchange, hosts can negotiate a transaction pacing value, Ta, using a TRANSACTION_PACING parameter in R1 and I2 packets. The parameter contains the minimum time (expressed in milliseconds) the host would wait between two NAT traversal transactions, such as starting a new connectivity check or retrying a previous check. The value that is used by both of the hosts is the higher of the two offered values.

The minimum Ta value **SHOULD** be configurable, and if no value is configured, a value of 50 ms **MUST** be used. Guidelines for selecting a Ta value are given in Appendix A. Hosts **MUST NOT** use values smaller than 5 ms for the minimum Ta, since such values may not work well with some NATs (as explained in [RFC8445]). The Initiator **MUST NOT** propose a smaller value than what the Responder offered. If a host does not include the TRANSACTION_PACING parameter in the base exchange, a Ta value of 50 ms **MUST** be used as that host's minimum value.

4.5. Base Exchange via Control Relay Server

This section describes how the Initiator and Responder perform a base exchange through a Control Relay Server. Connectivity pacing (denoted as TA_P here) was described in Section 4.4 and is not repeated here. Similarly, the NAT traversal mode negotiation process (denoted as NAT_TM in the example) was described in Section 4.3 and is also not repeated here. If a Control Relay Server receives an R1 or I2 packet without the NAT traversal mode parameter, it **MUST** drop it and **SHOULD** send a NOTIFY error packet with type NO_VALID_NAT_TRAVERSAL_MODE_PARAMETER to the sender of the R1 or I2.

It is **RECOMMENDED** that the Initiator send an I1 packet encapsulated in UDP when it is destined to an IP address of the Responder. Respectively, the Responder **MUST** respond to such an I1 packet with a UDP-encapsulated R1 packet, and also the rest of the communication related to the HIP association **MUST** also use UDP encapsulation.

Figure 4 illustrates a base exchange via a Control Relay Server. We assume that the Responder (i.e., a Control Relay Client) has already registered to the Control Relay Server. The Initiator may have also registered to another (or the same Control Relay Server), but the base exchange will traverse always through the Control Relay Server of the Responder.

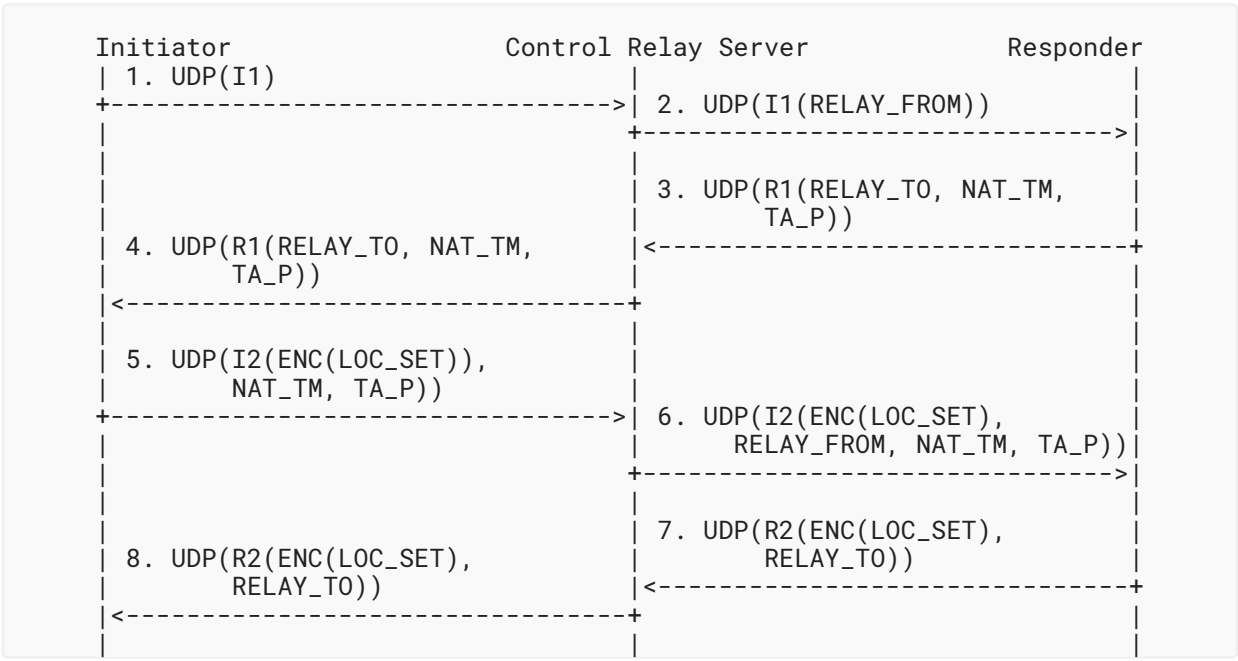

Figure 4: Base Exchange via a HIP Relay Server

In step 1 of Figure 4, the Initiator sends an I1 packet over UDP via the Control Relay Server to the Responder. In the HIP header, the source HIT belongs to the Initiator and the destination HIT to the Responder. The Initiator sends the I1 packet from its IP address to the IP address of the Control Relay Server over UDP.

In step 2, the Control Relay Server receives the I1 packet. If the destination HIT belongs to a successfully registered Control Relay Client (i.e., the host marked "Responder" in Figure 4), the Control Relay Server processes the packet. Otherwise, the Control Relay Server **MUST** drop the packet silently. The Control Relay Server appends a RELAY_FROM parameter to the I1 packet, which contains the transport source address and port of the I1 as observed by the Control Relay Server. The Control Relay Server protects the I1 packet with RELAY_HMAC, except that the parameter type is different as described in Section 5.8. The Control Relay Server changes the source and destination ports and IP addresses of the packet to match the values the Responder used when registering to the Control Relay Server, i.e., the reverse of the R2 used in the registration. The Control Relay Server **MUST** recalculate the transport checksum and forward the packet to the Responder.

In step 3, the Responder receives the I1 packet. The Responder processes it according to the rules in [RFC7401]. In addition, the Responder validates the RELAY_HMAC according to Section 5.8 and silently drops the packet if the validation fails. The Responder replies with an R1 packet to which it includes RELAY_TO and NAT traversal mode parameters. The Responder **MUST** include ICE-HIP-

UDP in the NAT traversal modes. The RELAY_TO parameter **MUST** contain the same information as the RELAY_FROM parameter, i.e., the Initiator's transport address, but the type of the parameter is different. The RELAY_TO parameter is not integrity protected by the signature of the R1 to allow pre-created R1 packets at the Responder.

In step 4, the Control Relay Server receives the R1 packet. The Control Relay Server drops the packet silently if the source HIT belongs to a Control Relay Client that has not successfully registered. The Control Relay Server **MAY** verify the signature of the R1 packet and drop it if the signature is invalid. Otherwise, the Control Relay Server rewrites the source address and port, and changes the destination address and port to match RELAY_TO information. Finally, the Control Relay Server recalculates the transport checksum and forwards the packet.

In step 5, the Initiator receives the R1 packet and processes it according to [RFC7401]. The Initiator **MAY** use the address in the RELAY_TO parameter as a local peer-reflexive candidate for this HIP association if it is different from all known local candidates. The Initiator replies with an I2 packet that uses the destination transport address of R1 as the source address and port. The I2 packet contains a LOCATOR_SET parameter inside an ENCRYPTED parameter that lists all the HIP candidates (HIP offer) of the Initiator. The candidates are encoded using the format defined in Section 5.7. The I2 packet **MUST** also contain a NAT traversal mode parameter that includes ICE-HIP-UDP mode. The ENCRYPTED parameter along with its key material generation is described in detail in Sections 5.2.18 and 6.5 in [RFC7401].

In step 6, the Control Relay Server receives the I2 packet. The Control Relay Server appends a RELAY_FROM and a RELAY_HMAC to the I2 packet similar to that explained in step 2, and forwards the packet to the Responder.

In step 7, the Responder receives the I2 packet and processes it according to [RFC7401]. The Responder validates the RELAY_HMAC according to Section 5.8 and silently drops the packet if the validation fails. It replies with an R2 packet and includes a RELAY_TO parameter as explained in step 3. The R2 packet includes a LOCATOR_SET parameter inside an ENCRYPTED parameter that lists all the HIP candidates (ICE answer) of the Responder. The RELAY_TO parameter is protected by the Hashed Message Authentication Code (HMAC). The ENCRYPTED parameter along with its key material generation is described in detail in Sections 5.2.18 and 6.5 in [RFC7401].

In step 8, the Control Relay Server processes the R2 as described in step 4. The Control Relay Server forwards the packet to the Initiator. After the Initiator has received the R2 and processed it successfully, the base exchange is completed.

Hosts **MUST** include the address of one or more Control Relay Servers (including the one that is being used for the initial signaling) in the LOCATOR_SET parameter in I2 and R2 messages if they intend to use such servers for relaying HIP signaling immediately after the base exchange completes. The traffic type of these addresses **MUST** be "HIP signaling" (see Section 5.7) and they **MUST NOT** be used for the connectivity tests described in Section 4.6. If the Control Relay Server locator used for relaying the base exchange is not included in I2 or R2 LOCATOR_SET parameters, it **SHOULD NOT** be used after the base exchange. Instead, further HIP signaling **SHOULD** use the

same path as the data traffic. It is **RECOMMENDED** to use the same Control Relay Server throughout the lifetime of the host association that was used for forwarding the base exchange if the Responder includes it in the locator parameter of the R2 message.

4.6. Connectivity Checks

When the Initiator and Responder complete the base exchange through the Control Relay Server, both of them employ the IP address of the Control Relay Server as the destination address for the packets. The address of the Control Relay Server **MUST NOT** be used as a destination for data plane traffic unless the server also supports Data Relay Server functionality, and the Client has successfully registered to use it. When NAT traversal mode with ICE-HIP-UDP was successfully negotiated and selected, the Initiator and Responder **MUST** start the connectivity checks in order to attempt to obtain direct end-to-end connectivity through NAT devices. It is worth noting that the connectivity checks **MUST** be completed even though no ESP_TRANSFORM would be negotiated and selected.

The connectivity checks follow the ICE methodology [ICE-NONSIP], but UDP-encapsulated HIP control messages are used instead of ICE messages. As stated in the ICE specification, the basic procedure for connectivity checks has three phases: sorting the candidate pairs according to their priority, sending checks in the prioritized order, and acknowledging the checks from the peer host.

The Initiator **MUST** take the role of controlling host, and the Responder acts as the controlled host. The roles **MUST** persist throughout the HIP associate lifetime (to be reused even during mobility UPDATE procedures). In the case in which both communicating nodes are initiating communication to each other using an I1 packet, the conflict is resolved as defined in Section 6.7 of [RFC7401]; the host with the "larger" HIT changes its role to Responder. In such a case, the host changing its role to Responder **MUST** also switch to the controlled role.

The protocol follows standard HIP UPDATE sending and processing rules as defined in Sections 6.11 and 6.12 in [RFC7401], but some new parameters are introduced (CANDIDATE_PRIORITY, MAPPED_ADDRESS, NOMINATE, PEER_PERMISSION, and RELAYED_ADDRESS).

4.6.1. Connectivity Check Procedure

Figure 5 illustrates connectivity checks in a simplified scenario where the Initiator and Responder have only a single candidate pair to check. Typically, NATs drop messages until both sides have sent messages using the same port pair. In this scenario, the Responder sends a connectivity check first but the NAT of the Initiator drops it. However, the connectivity check from the Initiator reaches the Responder because it uses the same port pair as the first message. It is worth noting that the message flow in this section is idealistic, and, in practice, more messages would be dropped, especially in the beginning. For instance, connectivity tests always start with the candidates with the highest priority, which would be host candidates (which would not reach the recipient in this scenario).

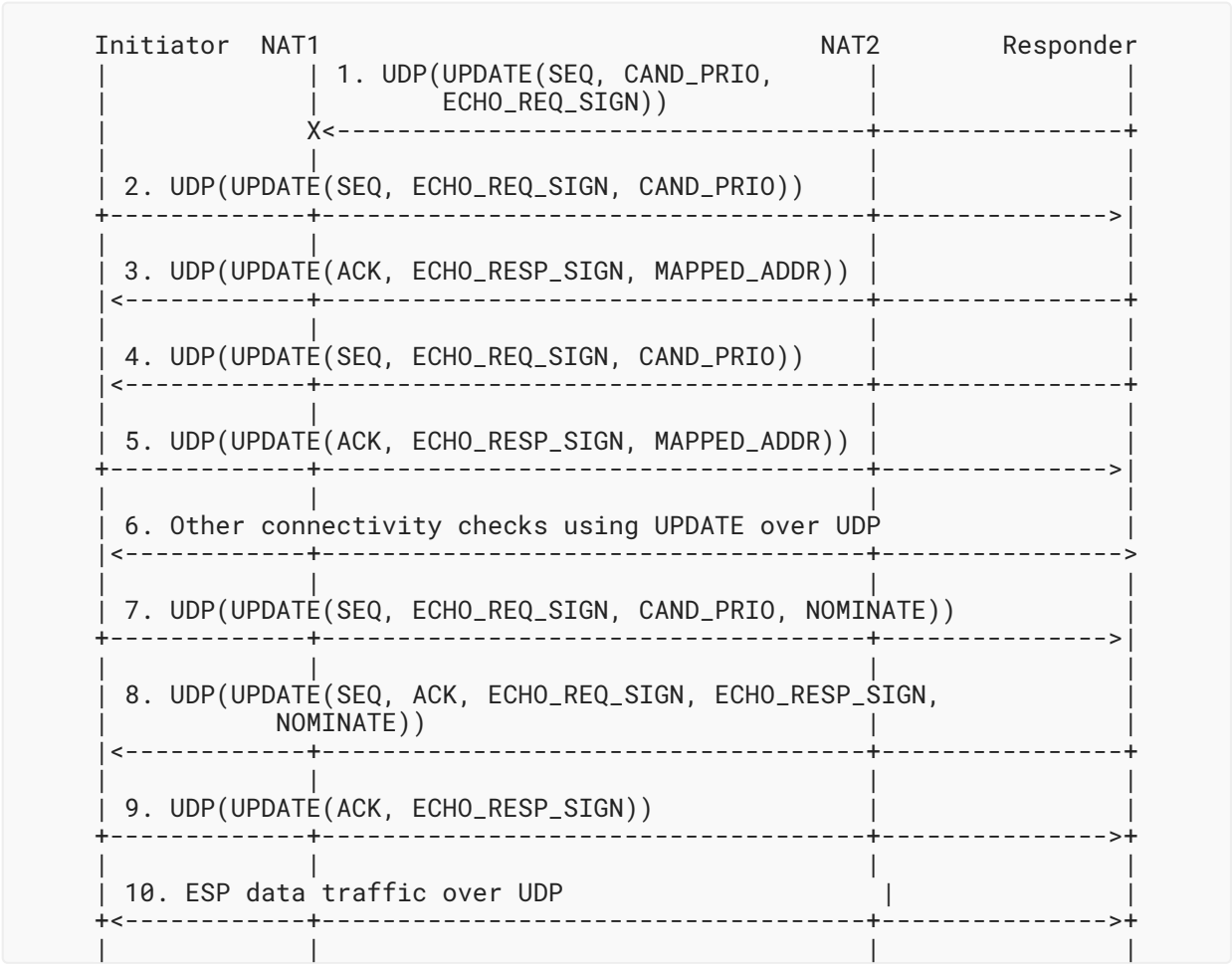

Figure 5: Connectivity Checks

In step 1, the Responder sends a connectivity check to the Initiator that the NAT of the Initiator drops. The message includes a number of parameters. As specified in [RFC7401], the SEQ parameter includes a running sequence identifier for the connectivity check. The candidate priority (denoted CAND_PRIO in the figure) describes the priority of the address candidate being tested. The ECHO_REQUEST_SIGNED (denoted ECHO_REQ_SIGN in the figure) includes a nonce that the recipient must sign and echo back as it is.

In step 2, the Initiator sends a connectivity check, using the same address pair candidate as in the previous step, and the message successfully traverses the NAT boxes. The message includes the same parameters as in the previous step. It should be noted that the sequence identifier is locally assigned by the Initiator, so it can be different than in the previous step.

In step 3, the Responder has successfully received the previous connectivity check from the Initiator and starts to build a response message. Since the message from the Initiator included a SEQ, the Responder must acknowledge it using an ACK parameter. Also, the nonce contained in the echo request must be echoed back in an ECHO_RESPONSE_SIGNED (denoted ECHO_RESP_SIGN) parameter. The Responder also includes a MAPPED_ADDRESS parameter

(denoted MAPPED_ADDR in the figure) that contains the transport address of the Initiator as observed by the Responder (i.e., peer-reflexive candidate). This message is successfully delivered to the Initiator; upon reception, the Initiator marks the candidate pair as valid.

In step 4, the Responder retransmits the connectivity check sent in the first step, since it was not acknowledged yet.

In step 5, the Initiator responds to the previous connectivity check message from the Responder. The Initiator acknowledges the SEQ parameter from the previous message using an ACK parameter and the ECHO_REQUEST_SIGNED parameter with ECHO_RESPONSE_SIGNED. In addition, it includes the MAPPED_ADDR parameter that includes the peer-reflexive candidate. This response message is successfully delivered to the Responder; upon reception, the Initiator marks the candidate pair as valid.

In step 6, despite the two hosts now having valid address candidates, the hosts still test the remaining address candidates in a similar way as in the previous steps. It should be noted that each connectivity check has a unique sequence number in the SEQ parameter.

In step 7, the Initiator has completed testing all address candidates and nominates one address candidate to be used. It sends an UPDATE message using the selected address candidates that includes a number of parameters: SEQ, ECHO_REQUEST_SIGNED, CANDIDATE_PRIORITY, and the NOMINATE parameter.

In step 8, the Responder receives the message with the NOMINATE parameter from the Initiator. It sends a response that includes the NOMINATE parameter in addition to a number of other parameters. The ACK and ECHO_RESPONSE_SIGNED parameters acknowledge the SEQ and ECHO_REQUEST_SIGNED parameters from the previous message from the Initiator. The Responder includes SEQ and ECHO_REQUEST_SIGNED parameters in order to receive an acknowledgment from the Responder.

In step 9, the Initiator completes the candidate nomination process by confirming the message reception to the Responder. In the confirmation message, the ACK and ECHO_RESPONSE_SIGNED parameters correspond to the SEQ and ECHO_REQUEST_SIGNED parameters in the message sent by the Responder in the previous step.

In step 10, the Initiator and Responder can start sending application payload over the successfully nominated address candidates.

It is worth noting that if either host has registered a relayed address candidate from a Data Relay Server, the host **MUST** activate the address before connectivity checks by sending an UPDATE message containing the PEER_PERMISSION parameter as described in Section 4.12.1. Otherwise, the Data Relay Server drops ESP packets using the relayed address.

It should be noted that in the case in which both the Initiator and Responder are advertising their own relayed address candidates, it is possible that the two hosts choose the two relayed addresses as a result of the ICE nomination algorithm. While this is possible (and even could be desirable for privacy reasons), it can be unlikely due to low priority assigned for the relayed address candidates. In such an event, the nominated address pair is always symmetric; the

nomination algorithm prevents asymmetric address pairs (i.e., each side choosing different pair) such as a Data Relay Client using its own Data Relay Server to send data directly to its peer while receiving data from the Data Relay Server of its peer.

4.6.2. Rules for Connectivity Checks

The HITs of the two communicating hosts **MUST** be used as credentials in this protocol (in contrast to ICE, which employs username-password fragments). A HIT pair uniquely identifies the corresponding HIT association, and a SEQ number in an UPDATE message identifies a particular connectivity check.

All of the connectivity check messages **MUST** be protected with HIP_HMAC and signatures (even though the illustrations in this specification omit them for simplicity) according to [RFC7401]. Each connectivity check sent by a host **MUST** include a SEQ parameter and ECHO_REQUEST_SIGNED parameter; correspondingly, the peer **MUST** respond to these using ACK and ECHO_RESPONSE_SIGNED according to the rules specified in [RFC7401].

The host sending a connectivity check **MUST** validate that the response uses the same pair of UDP ports, and drop the packet if this is not the case.

A host may receive a connectivity check before it has received the candidates from its peer. In such a case, the host **MUST** immediately queue a response by placing it in the triggered-check queue and then continue waiting for the candidates. A host **MUST NOT** select a candidate pair until it has verified the pair using a connectivity check as defined in Section 4.6.1.

Section 5.3.5 of [RFC7401] states that UPDATE packets have to include either a SEQ or ACK parameter (but can include both). In the connectivity check procedure specified in Section 4.6.1, each SEQ parameter should be acknowledged separately. In the context of NATs, this means that some of the SEQ parameters sent in connectivity checks will be lost or arrive out of order. From the viewpoint of the recipient, this is not a problem since the recipient will just "blindly" acknowledge the SEQ. However, the sender needs to be prepared for lost sequence identifiers and ACK parameters that arrive out of order.

As specified in [RFC7401], an ACK parameter may acknowledge multiple sequence identifiers. While the examples in the previous sections do not illustrate such functionality, it is also permitted when employing ICE-HIP-UDP mode.

In ICE-HIP-UDP mode, a retransmission of a connectivity check **SHOULD** be sent with the same sequence identifier in the SEQ parameter. Some tested address candidates will never produce a working address pair and may thus cause retransmissions. Upon successful nomination of an address pair, a host **SHOULD** immediately stop sending such retransmissions.

Full ICE procedures for prioritizing candidates, eliminating redundant candidates, forming checklists (including pruning), and triggered-check queues **MUST** be followed as specified in Section 6.1 of [RFC8445], with the exception being that the foundation, frozen candidates, and default candidates are not used. From the viewpoint of the ICE specification [RFC8445], the

protocol specified in this document operates using a component ID of 1 on all candidates, and the foundation of all candidates is unique. This specification defines only "full ICE" mode, and the "lite ICE" is not supported. The reasoning behind the missing features is described in Appendix B.

The connectivity check messages **MUST** be paced by the Ta value negotiated during the base exchange as described in Section 4.4. If neither one of the hosts announced a minimum pacing value, a value of 50 ms **MUST** be used.

Both hosts **MUST** form a priority ordered checklist and begin to check transactions every Ta milliseconds as long as the checks are running and there are candidate pairs whose tests have not started. The retransmission timeout (RTO) for the connectivity check UPDATE packets **SHOULD** be calculated as follows:

RTO = MAX (1000 ms, Ta * (Num-Waiting + Num-In-Progress))

In the RTO formula, Ta is the value used for the connectivity check pacing, Num-Waiting is the number of pairs in the checklist in the "Waiting" state, and Num-In-Progress is the number of pairs in the "In-Progress" state. This is identical to the formula in [RFC8445] when there is only one checklist. A smaller value than 1000 ms for the RTO **MUST NOT** be used.

Each connectivity check request packet **MUST** contain a CANDIDATE_PRIORITY parameter (see Section 5.14) with the priority value that would be assigned to a peer-reflexive candidate if one was learned from the corresponding check. An UPDATE packet that acknowledges a connectivity check request **MUST** be sent from the same address that received the check and delivered to the same address where the check was received from. Each acknowledgment UPDATE packet **MUST** contain a MAPPED_ADDRESS parameter with the port, protocol, and IP address of the address where the connectivity check request was received from.

Following the ICE guidelines [RFC8445], it is **RECOMMENDED** to restrict the total number of connectivity checks to 100 for each host association. This can be achieved by limiting the connectivity checks to the 100 candidate pairs with the highest priority.

4.6.3. Rules for Concluding Connectivity Checks

The controlling agent may find multiple working candidate pairs. To conclude the connectivity checks, it **SHOULD** nominate the pair with the highest priority. The controlling agent **MUST** nominate a candidate pair essentially by repeating a connectivity check using an UPDATE message that contains a SEQ parameter (with a new sequence number), an ECHO_REQUEST_SIGNED parameter, the priority of the candidate in a CANDIDATE_PRIORITY parameter, and a NOMINATE parameter to signify conclusion of the connectivity checks. Since the nominated address pair has already been tested for reachability, the controlled host should be able to receive the message. Upon reception, the controlled host **SHOULD** select the nominated address pair. The response message **MUST** include a SEQ parameter with a new sequence identifier, acknowledgment of the sequence from the controlling host in an ACK parameter, a new ECHO_REQUEST_SIGNED parameter, an ECHO_RESPONSE_SIGNED parameter corresponding to the ECHO_REQUEST_SIGNED parameter from the controlling host, and the NOMINATE parameter. After sending this packet, the controlled host can create IPsec security associations using the nominated address candidate for delivering application payload to the

controlling host. Since the message from the controlled host included a new sequence identifier echo request for the signature, the controlling host **MUST** acknowledge this with a new UPDATE message that includes an ACK and ECHO_RESPONSE_SIGNED parameters. After this final concluding message, the controlling host also can create IPsec security associations for delivering application payload to the controlled host.

It is possible that packets are delayed by the network. Both hosts **MUST** continue to respond to any connectivity checks despite an address pair having been nominated.

If all the connectivity checks have failed, the hosts **MUST NOT** send ESP traffic to each other but **MAY** continue communicating using HIP packets and the locators used for the base exchange. Also, the hosts **SHOULD** notify each other about the failure with a CONNECTIVITY_CHECKS_FAILED NOTIFY packet (see Section 5.10).

4.7. NAT Traversal Optimizations

4.7.1. Minimal NAT Traversal Support

If the Responder has a fixed and publicly reachable IPv4 address and does not employ a Control Relay Server, the explicit NAT traversal mode negotiation **MAY** be omitted; thus, even the UDP-ENCAPSULATION mode does not have to be negotiated. In such a scenario, the Initiator sends an I1 message over UDP and the Responder responds with an R1 message over UDP without including any NAT traversal mode parameter. The rest of the base exchange follows the procedures defined in [RFC7401], except that the control and data plane use UDP encapsulation. Here, the use of UDP for NAT traversal is agreed upon implicitly. This way of operation is still subject to NAT timeouts, and the hosts **MUST** employ NAT keepalives as defined in Section 4.10.

When UDP-ENCAPSULATION mode is chosen either explicitly or implicitly, the connectivity checks as defined in this document **MUST NOT** be used. When hosts lose connectivity, they **MUST** instead utilize [RFC8046] or [RFC8047] procedures, but with the difference being that UDP-based tunneling **MUST** be employed for the entire lifetime of the corresponding HIP association.

4.7.2. Base Exchange without Connectivity Checks

It is possible to run a base exchange without any connectivity checks as defined in Legacy ICE-HIP (Section 4.8 of [RFC5770]). The procedure is also applicable in the context of this specification, so it is repeated here for completeness.

In certain network environments, the connectivity checks can be omitted to reduce initial connection setup latency because a base exchange acts as an implicit connectivity test itself. For this to work, the Initiator **MUST** be able to reach the Responder by simply UDP encapsulating HIP and ESP packets sent to the Responder's address. Detecting and configuring this particular scenario is prone to failure unless carefully planned.

In such a scenario, the Responder **MAY** include UDP-ENCAPSULATION NAT traversal mode as one of the supported modes in the R1 packet. If the Responder has registered to a Control Relay Server in order to discover its address candidates, it **MUST** also include a LOCATOR_SET parameter encapsulated inside an ENCRYPTED parameter in an R1 message that contains a preferred address where the Responder is able to receive UDP-encapsulated ESP and HIP

packets. This locator **MUST** be of type "Transport address", its Traffic type **MUST** be "both", and it **MUST** have the "Preferred bit" set (see Table 2). If there is no such locator in R1, the Initiator **MUST** use the source address of the R1 as the Responder's preferred address.

The Initiator **MAY** choose the UDP-ENCAPSULATION mode if the Responder listed it in the supported modes and the Initiator does not wish to use the connectivity checks defined in this document for searching for a more optimal path. In this case, the Initiator sends the I2 with UDP-ENCAPSULATION mode in the NAT traversal mode parameter directly to the Responder's preferred address (i.e., to the preferred locator in R1 or to the address where R1 was received from if there was no preferred locator in R1). The Initiator **MAY** include locators in I2 but they **MUST NOT** be taken as address candidates, since connectivity checks defined in this document will not be used for connections with UDP-ENCAPSULATION NAT traversal mode. Instead, if R2 and I2 are received and processed successfully, a security association can be created and UDP-encapsulated ESP can be exchanged between the hosts after the base exchange completes according to the rules in Section 4.4 of [RFC7401].

The Control Relay Server can be used for discovering address candidates but it is not intended to be used for relaying end-host packets using the UDP-ENCAPSULATION NAT mode. Since an I2 packet with UDP-ENCAPSULATION NAT traversal mode selected **MUST NOT** be sent via a Control Relay Server, the Responder **SHOULD** reject such I2 packets and reply with a NO_VALID_NAT_TRAVERSAL_MODE_PARAMETER NOTIFY packet (see Section 5.10).

If there is no answer for the I2 packet sent directly to the Responder's preferred address, the Initiator **MAY** send another I2 via the Control Relay Server, but it **MUST NOT** choose UDP-ENCAPSULATION NAT traversal mode for that I2.

4.7.3. Initiating a Base Exchange Both with and without UDP Encapsulation

It is possible to run a base exchange in parallel both with and without UDP encapsulation as defined in Legacy ICE-HIP (Section 4.9 of [RFC5770]). The procedure is also applicable in the context of this specification, so it is repeated here for completeness.

The Initiator **MAY** also try to simultaneously perform a base exchange with the Responder without UDP encapsulation. In such a case, the Initiator sends two I1 packets, one without and one with UDP encapsulation, to the Responder. The Initiator **MAY** wait for a while before sending the other I1. How long to wait and in which order to send the I1 packets can be decided based on local policy. For retransmissions, the procedure is repeated.

The I1 packet without UDP encapsulation may arrive directly, without passing a Control Relay Server, at the Responder. When this happens, the procedures in [RFC7401] are followed for the rest of the base exchange. The Initiator may receive multiple R1 packets, with and without UDP encapsulation, from the Responder. However, after receiving a valid R1 and answering it with an I2, further R1 packets that are not retransmissions of the R1 message received first **MUST** be ignored.

The I1 packet without UDP encapsulation may also arrive at a HIP-capable middlebox. When the middlebox is a HIP Rendezvous Server and the Responder has successfully registered with the rendezvous service, the middlebox follows rendezvous procedures in [RFC8004].

If the Initiator receives a NAT traversal mode parameter in R1 without UDP encapsulation, the Initiator **MAY** ignore this parameter and send an I2 without UDP encapsulation and without any selected NAT traversal mode. When the Responder receives the I2 without UDP encapsulation and without NAT traversal mode, it will assume that no NAT traversal mechanism is needed. The packet processing will be done as described in [RFC7401]. The Initiator **MAY** store the NAT traversal modes for future use, e.g., in case of a mobility or multihoming event that causes NAT traversal to be used during the lifetime of the HIP association.

4.8. Sending Control Packets after the Base Exchange

The same considerations with regard to sending control packets after the base exchange as described in Legacy ICE-HIP (Section 5.10 of [RFC5770]) also apply here, so they are repeated here for completeness.

After the base exchange, the two end hosts **MAY** send HIP control packets directly to each other using the transport address pair established for a data channel without sending the control packets through any Control Relay Servers. When a host does not receive acknowledgments, e.g., to an UPDATE or CLOSE packet after a timeout based on local policies, a host **SHOULD** resend the packet through the associated Data Relay Server of the peer (if the peer listed it in its LOCATOR_SET parameter in the base exchange according to the rules specified in Section 4.4.2 of [RFC7401]).

If a Control Relay Client sends a packet through a Control Relay Server, the Control Relay Client **MUST** always utilize the RELAY_TO parameter. The Control Relay Server **SHOULD** forward HIP control packets originating from a Control Relay Client to the address denoted in the RELAY_TO parameter. In the other direction, the Control Relay Server **SHOULD** forward HIP control packets to the Control Relay Clients and **MUST** add a RELAY_FROM parameter to the control packets it relays to the Control Relay Clients.

If the Control Relay Server is not willing or able to relay a HIP packet, it **MAY** notify the sender of the packet with a MESSAGE_NOT_RELAYED error notification (see Section 5.10).

4.9. Mobility Handover Procedure

A host may move after base exchange and connectivity checks. Mobility extensions for HIP [RFC8046] define handover procedures without NATs. In this section, we define how two hosts interact with handover procedures in scenarios involving NATs. The specified extensions define only simple mobility using a pair of security associations, and multihoming extensions are left to be defined in later specifications. The procedures in this section offer the same functionality as "ICE restart" specified in [RFC8445]. The example described in this section shows only a Control Relay Server for the peer host for the sake of simplicity, but the mobile host may also have a Control Relay Server.

The assumption here is that the two hosts have successfully negotiated and chosen the ICE-HIP-UDP mode during the base exchange as defined in Section 4.3. The Initiator of the base exchange **MUST** store information that it was the controlling host during the base exchange. Similarly, the Responder **MUST** store information that it was the controlled host during the base exchange.

Prior to starting the handover procedures with all peer hosts, the mobile host **SHOULD** first send its locators in UPDATE messages to its Control and Data Relay Servers if it has registered to such. It **SHOULD** wait for all of them to respond for a configurable time, by default two minutes, and then continue with the handover procedure without information from the Relay Server that did not respond. As defined in Section 4.1, a response message from a Control Relay Server includes a REG_FROM parameter that describes the server-reflexive candidate of the mobile host to be used in the candidate exchange during the handover. Similarly, an UPDATE to a Data Relay Server is necessary to make sure the Data Relay Server can forward data to the correct IP address after a handover.

The mobility extensions for NAT traversal are illustrated in Figure 6. The mobile host is the host that has changed its locators, and the peer host is the host it has a host association with. The mobile host may have multiple peers, and it repeats the process with all of its peers. In the figure, the Control Relay Server belongs to the peer host, i.e., the peer host is a Control Relay Client for the Control Relay Server. Note that the figure corresponds to figure 3 in [RFC8046], but the difference is that the main UPDATE procedure is carried over the relay and the connectivity is tested separately. Next, we describe the procedure of that figure in detail.

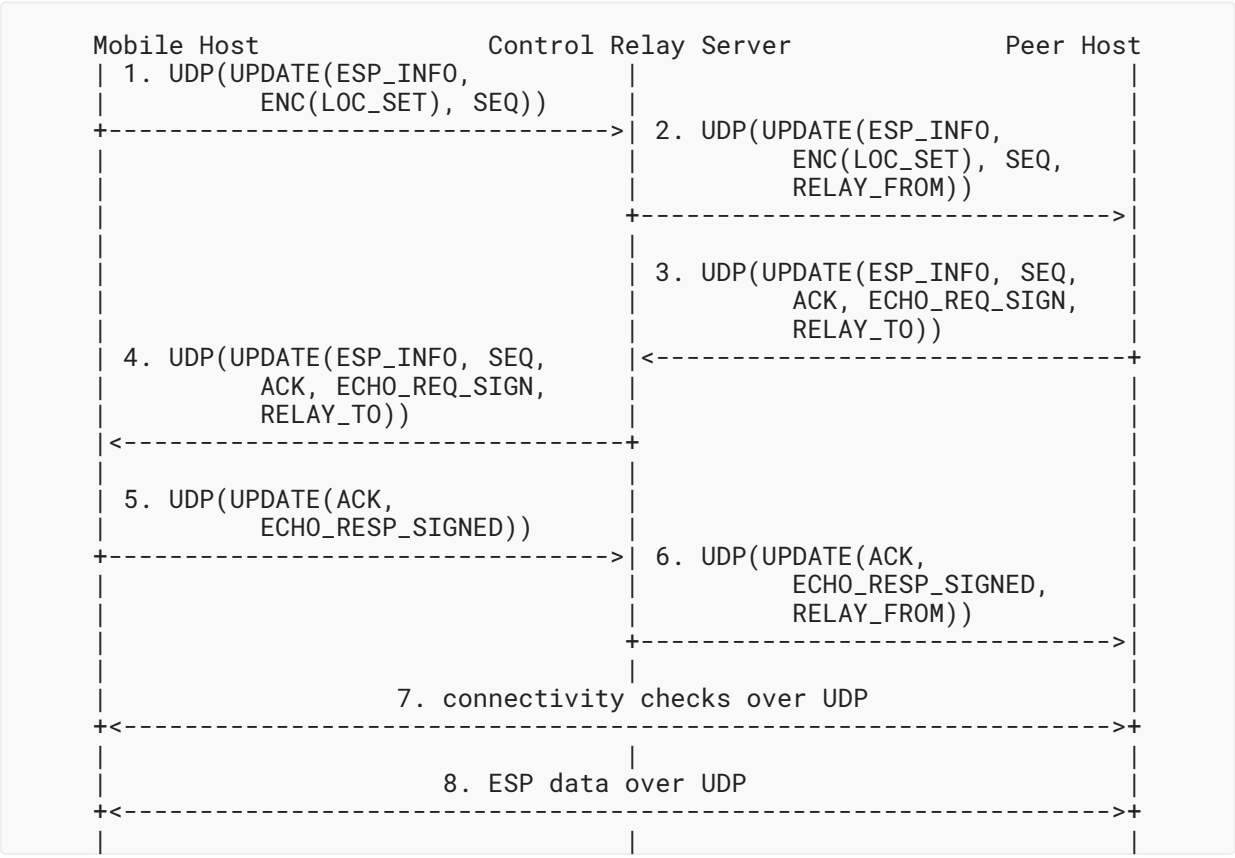

Figure 6: HIP UPDATE Procedure

In step 1, the mobile host has changed location and sends a location update to its peer through the Control Relay Server of the peer. It sends an UPDATE packet with the source HIT belonging to itself and destination HIT belonging to the peer host. In the packet, the source IP address belongs to the mobile host and the destination to the Control Relay Server. The packet contains an ESP_INFO parameter where, in this case, the OLD SPI and NEW SPI parameters both contain the pre-existing incoming SPI. The packet also contains the locators of the mobile host in a LOCATOR_SET parameter, encapsulated inside an ENCRYPTED parameter (see Sections 5.2.18 and 6.5 in [RFC7401] for details on the ENCRYPTED parameter). The packet also contains a SEQ number to be acknowledged by the peer. As specified in [RFC8046], the packet may also include a HOST_ID (for middlebox inspection) and DIFFIE_HELLMAN parameter for rekeying.

In step 2, the Control Relay Server receives the UPDATE packet and forwards it to the peer host (i.e., Control Relay Client). The Control Relay Server rewrites the destination IP address and appends a RELAY_FROM parameter to the message.

In step 3, the peer host receives the UPDATE packet, processes it, and responds with another UPDATE message. The message is destined to the HIT of the mobile host and to the IP address of the Control Relay Server. The message includes an ESP_INFO parameter where, in this case, the OLD SPI and NEW SPI parameters both contain the pre-existing incoming SPI. The peer includes a new SEQ and ECHO_REQUEST_SIGNED parameter to be acknowledged by the mobile host. The message acknowledges the SEQ parameter of the earlier message with an ACK parameter. The RELAY_TO parameter specifies the address of the mobile host where the Control Relay Server should forward the message.

In step 4, the Control Relay Server receives the message, rewrites the destination IP address and UDP port according to the RELAY_TO parameter, and then forwards the modified message to the mobile host.

In step 5, the mobile host receives the UPDATE packet and processes it. The mobile host concludes the handover procedure by acknowledging the received SEQ parameter with an ACK parameter and the ECHO_REQUEST_SIGNED parameter with an ECHO_RESPONSE_SIGNED parameter. The mobile host sends the packet to the HIT of the peer and to the address of the HIP relay. The mobile host can start connectivity checks after this packet.

In step 6, the HIP relay receives the UPDATE packet and forwards it to the peer host (i.e., Relay Client). The HIP relay rewrites the destination IP address and port, and then appends a RELAY_FROM parameter to the message. When the peer host receives this concluding UPDATE packet, it can initiate the connectivity checks.

In step 7, the two hosts test for connectivity across NATs according to procedures described in Section 4.6. The original Initiator of the communications is the controlling host and the original Responder is the controlled host.

In step 8, the connectivity checks are successfully completed and the controlling host has nominated one address pair to be used. The hosts set up security associations to deliver the application payload.

It is worth noting that the Control and Data Relay Client do not have to reregister for the related services after a handover. However, if a Data Relay Client has registered a relayed address candidate from a Data Relay Server, the Data Relay Client **MUST** reactivate the address before the connectivity checks by sending an UPDATE message containing the PEER_PERMISSION parameter as described in Section 4.12.1. Otherwise, the Data Relay Server drops ESP packets sent to the relayed address.

In the so-called "double jump" or simultaneous mobility scenario, both peers change their location simultaneously. In such a case, both peers trigger the procedure described earlier in this section at the same time. In other words, both of the communicating hosts send an UPDATE packet carrying locators at the same time or with some delay. When the locators are exchanged almost simultaneously (reliably via Control Relay Servers), the two hosts can continue with connectivity checks after both have completed separately the steps in Figure 6. The problematic case occurs when one of the hosts (referred to here as host "M") moves later during the connectivity checks. In such a case, host M sends a locator to the peer, which is in the middle of connectivity checks. Upon receiving the UPDATE message, the peer responds with an UPDATE with ECHO_REQ_SIGN as described in step 3 in Figure 6. Upon receiving the valid response from host M as described in step 6, the peer host **MUST** restart the connectivity checks with host M. This way, both hosts start the connectivity checks roughly in a synchronized way. It is also important that the peer host does not restart the connectivity checks until step 6 is successfully completed, because the UPDATE message carrying locators in step 1 could be replayed by an attacker.

4.10. NAT Keepalives

To prevent NAT states from expiring, communicating hosts **MUST** send periodic keepalives to other hosts with which they have established a HIP association every 15 seconds (the so-called Tr value in ICE). Other values **MAY** be used, but a Tr value smaller than 15 seconds **MUST NOT** be used. Both a Control/Data Relay Client and Control/Data Relay Server, as well as two peers employing UDP-ENCAPSULATION or ICE-HIP-UDP mode, **SHOULD** send HIP NOTIFY packets unless they have exchanged some other traffic over the used UDP ports. However, the Data Relay Client and Data Relay Server **MUST** employ only HIP NOTIFY packets in order to keep the server-reflexive candidates alive. The keepalive message encoding format is defined in Section 5.3. If the base exchange or mobility handover procedure occurs during an extremely slow path, a host (with a HIP association with the peer) **MAY** also send HIP NOTIFY packets every 15 seconds to keep the path active with the recipient.

4.11. Closing Procedure

The two-way procedure for closing a HIP association and the related security associations is defined in [RFC7401]. One host initiates the procedure by sending a CLOSE message and the recipient confirms it with CLOSE_ACK. All packets are protected using HMACs and signatures, and the CLOSE messages include an ECHO_REQUEST_SIGNED parameter to protect against replay attacks.

The same procedure for closing HIP associations also applies here, but the messaging occurs using the UDP-encapsulated tunnel that the two hosts employ. A host sending the CLOSE message **SHOULD** first send the message over a direct link. After a number of retransmissions, it **MUST** send over a Control Relay Server of the recipient if one exists. The host receiving the CLOSE message directly without a Control Relay Server **SHOULD** respond directly. If the CLOSE message came via a Control Relay Server, the host **SHOULD** respond using the same Control Relay Server.

4.12. Relaying Considerations

4.12.1. Forwarding Rules and Permissions

The Data Relay Server uses a similar permission model as a TURN server: before the Data Relay Server forwards any ESP data packets from a peer to a Data Relay Client (or the other direction), the client **MUST** set a permission for the peer's address. The permissions also install a forwarding rule for each direction, similar to TURN's channels, based on the Security Parameter Index (SPI) values in the ESP packets.

Permissions are not required for HIP control packets. However, if a relayed address (as conveyed in the RELAYED_ADDRESS parameter from the Data Relay Server) is selected to be used for data, the Control Relay Client **MUST** send an UPDATE message to the Data Relay Server containing a PEER_PERMISSION parameter (see Section 5.13) with the following information: the UDP port and address for the server-reflexive address, the UDP port and address of the peer, and the inbound and outbound SPIs used for ESP. The packet **MUST** be sent to the same UDP tunnel the Client employed in the base exchange to contact the Server (i.e., not to the port occupied by the server-reflexive candidate). To avoid packet dropping of ESP packets, the Control Relay Client **SHOULD** send the PEER_PERMISSION parameter before connectivity checks both in the case of base exchange and a mobility handover. It is worth noting that the UPDATE message includes a SEQ parameter (as specified in [RFC7401]) that the Data Relay Server must acknowledge, so that the Control Relay Client can resend the message with the PEER_PERMISSION parameter if it gets lost.

When a Data Relay Server receives an UPDATE with a PEER_PERMISSION parameter, it **MUST** check if the sender of the UPDATE is registered for data-relaying service, and drop the UPDATE if the host was not registered. If the host was registered, the Data Relay Server checks if there is a permission with matching information (protocol, addresses, ports, and SPI values). If there is no such permission, a new permission **MUST** be created and its lifetime **MUST** be set to 5 minutes. If an identical permission already existed, it **MUST** be refreshed by setting the lifetime to 5 minutes. A Data Relay Client **SHOULD** refresh permissions 1 minute before the expiration when the permission is still needed.

When a Data Relay Server receives an UPDATE from a registered client but without a PEER_PERMISSION parameter and with a new locator set, the Data Relay Server can assume that the mobile host has changed its location and is thus not reachable in its previous location. In such an event, the Data Relay Server **SHOULD** deactivate the permission and stop relaying data plane traffic to the client.

The relayed address **MUST** be activated with the PEER_PERMISSION parameter both after a base exchange and after a handover procedure with another ICE-HIP-UDP-capable host. Unless activated, the Data Relay Server **MUST** drop all ESP packets. It is worth noting that a Data Relay Client does not have to renew its registration upon a change of location UPDATE, but only when the lifetime of the registration is close to end.

4.12.2. HIP Data Relay and Relaying of Control Packets

When a Data Relay Server accepts to relay UDP-encapsulated ESP between a Data Relay Client and its peer, the Data Relay Server opens a UDP port (relayed address) for this purpose as described in Section 4.1. This port can be used for also delivering control packets because connectivity checks also cover the path through the Data Relay Server. If the Data Relay Server receives a UDP-encapsulated HIP control packet on that port, it **MUST** forward the packet to the Data Relay Client and add a RELAY_FROM parameter to the packet as if the Data Relay Server were acting as a Control Relay Server. When the Data Relay Client replies to a control packet with a RELAY_FROM parameter via its Data Relay Server, the Data Relay Client **MUST** add a RELAY_TO parameter containing the peer's address and use the address of its Data Relay Server as the destination address. Further, the Data Relay Server **MUST** send this packet to the peer's address from the relayed address.

If the Data Relay Server receives a UDP packet that is not a HIP control packet to the relayed address, it **MUST** check if it has a permission set for the peer the packet is arriving from (i.e., the sender's address and SPI value matches to an installed permission). If permissions are set, the Data Relay Server **MUST** forward the packet to the Data Relay Client that created the permission. The Data Relay Server **MUST** also implement the similar checks for the reverse direction (i.e., ESP packets from the Data Relay Client to the peer). Packets without a permission **MUST** be dropped silently.

4.12.3. Handling Conflicting SPI Values

From the viewpoint of a host, its remote peers can have overlapping inbound SPI numbers because the IPsec also uses the destination IP address to index the remote peer host. However, a Data Relay Server can represent multiple remote peers, thus masquerading the actual destination. Since a Data Relay Server may have to deal with a multitude of Relay Clients and their peers, a Data Relay Server may experience collisions in the SPI namespace, thus being unable to forward datagrams to the correct destination. Since the SPI space is 32 bits and the SPI values should be random, the probability for a conflicting SPI value is fairly small but could occur on a busy Data Relay Server. The two problematic cases are described in this section.

In the first scenario, the SPI collision problem occurs if two hosts have registered to the same Data Relay Server and a third host initiates base exchange with both of them. Here, the two Responders (i.e., Data Relay Clients) claim the same inbound SPI number with the same Initiator (peer). However, in this case, the Data Relay Server has allocated separate UDP ports for the two Data Relay Clients acting now as Responders (as recommended in Section 7.5). When the third host sends an ESP packet, the Data Relay Server is able to forward the packet to the correct Data Relay Client because the destination UDP port is different for each of the clients.

In the second scenario, an SPI collision may occur when two Initiators run a base exchange to the same Responder (i.e., Data Relay Client), and both of the Initiators claim the same inbound SPI at the Data Relay Server using the PEER_PERMISSION parameter. In this case, the Data Relay Server cannot disambiguate the correct destination of an ESP packet originating from the Data Relay Client because the SPI could belong to either of the peers (and the destination IP and UDP port belonging to the Data Relay Server are not unique either). The recommended way and a contingency plan to solve this issue are described below.

The recommend way to mitigate the problem is as follows. For each new HIP association, a Data Relay Client acting as a Responder **SHOULD** register a new server-reflexive candidate as described in Section 4.2. Similarly, the Data Relay Server **SHOULD NOT** reuse the port numbers as described in Section 7.5. This way, each server-reflexive candidate for the Data Relay Client has a separate UDP port that the Data Relay Server can use to disambiguate packet destinations in case of SPI collisions.

When the Data Relay Client is not registering or failed to register a new relay candidate for a new peer, the Data Relay Client **MUST** follow a contingency plan as follows. Upon receiving an I2 with a colliding SPI, the Data Relay Client acting as the Responder **MUST NOT** include the relayed address candidate in the R2 message because the Data Relay Server would not be able to demultiplex the related ESP packet to the correct Initiator. The same also applies to the handover procedures; the Data Relay Client **MUST NOT** include the relayed address candidate when sending its new locator set in an UPDATE to its peer if it would cause an SPI conflict with another peer.

5. Packet Formats

The following subsections define the parameter and packet encodings for the HIP and ESP packets. All values **MUST** be in network byte order.

It is worth noting that all of the parameters are shown for the sake of completeness even though they are specified already in Legacy ICE-HIP [RFC5770]. New parameters are explicitly described as new.

5.1. HIP Control Packets

Figure 7 illustrates the packet format for UDP-encapsulated HIP. The format is identical to Legacy ICE-HIP [RFC5770].

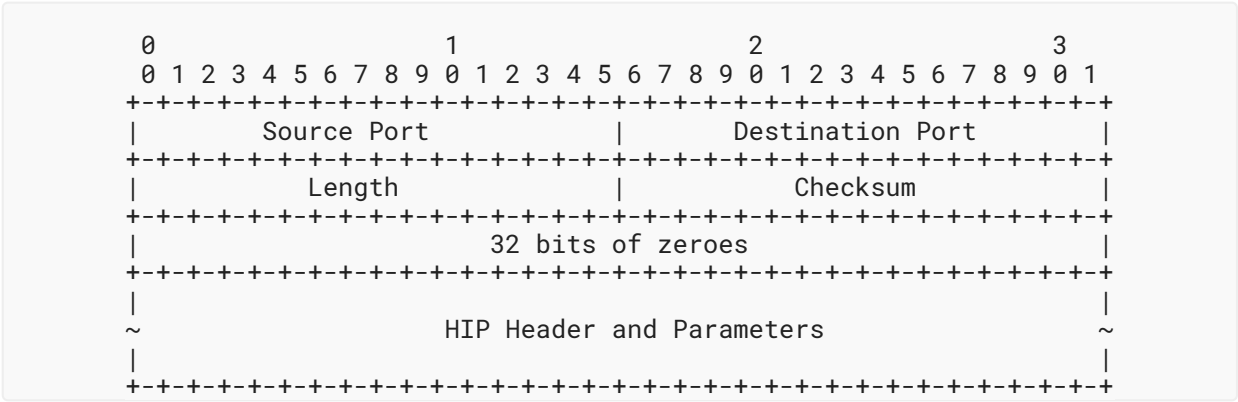

Figure 7: Format of UDP-Encapsulated HIP Control Packets

HIP control packets are encapsulated in UDP packets as defined in Section 2.2 of [RFC3948], "IKE Header Format for Port 4500", except that a different port number is used. Figure 7 illustrates the encapsulation. The UDP header is followed by 32 zero bits that can be used to differentiate HIP control packets from ESP packets. The HIP header and parameters follow the conventions of [RFC7401] with the exception that the HIP header checksum **MUST** be zero. The HIP header checksum is zero for two reasons. First, the UDP header already contains a checksum. Second, the checksum definition in [RFC7401] includes the IP addresses in the checksum calculation. The NATs that are unaware of HIP cannot recompute the HIP checksum after changing IP addresses.

A Control/Data Relay Server or a non-relay Responder **SHOULD** listen at UDP port 10500 for incoming UDP-encapsulated HIP control packets. If some other port number is used, it needs to be known by potential Initiators.

UDP encapsulation of HIP packets reduces the Maximum Transmission Unit (MTU) size of the control plane by 12 bytes (8-byte UDP header plus 4-byte zero SPI marker), and the data plane by 8 bytes. Additional HIP relay parameters, such as RELAY_HMAC, RELAY_UDP_HIP, RELAY_UDP_ESP, etc., further increase the size of certain HIP packets. In regard to MTU, the following aspects need to be considered in an implementation:

- A HIP host **SHOULD** implement ICMP message handling to support Path MTU Discovery (PMTUD) as described in [RFC1191] and [RFC8201].
- Reliance on IP fragmentation is unlikely to be a viable strategy through NATs. If ICMP MTU discovery is not working, MTU-related path black holes may occur.
- A mitigation strategy is to constrain the MTU, especially for virtual interfaces, to expected safe MTU values, e.g., 1400 bytes for the underlying interfaces that support 1500 bytes MTU.
- Further extensions to this specification may define a HIP-based mechanism to find a working path MTU without unnecessary constraining that size using Packetization Layer Path MTU Discovery for Datagram Transports [RFC8899]. For instance, such a mechanism could be implemented between a HIP Relay Client and HIP Relay Server.
- It is worth noting that further HIP extensions can trim off 8 bytes in the ESP header by negotiating implicit initialization vector (IV) support in the ESP_TRANSFORM parameter as described in [RFC8750].

5.2. Connectivity Checks

HIP connectivity checks are HIP UPDATE packets. The format is specified in [RFC7401].

5.3. Keepalives

The **RECOMMENDED** encoding format for keepalives is HIP NOTIFY packets as specified in [RFC7401] with the Notify message type field set to NAT_KEEPALIVE (16385) and with an empty Notification data field. It is worth noting that the sending of such a HIP NOTIFY message **SHOULD** be omitted if the host is sending some other traffic (HIP or ESP) to the peer host over the related UDP tunnel during the Tr period. For instance, the host **MAY** actively send ICMPv6 requests (or respond with an ICMPv6 response) inside the ESP tunnel to test the health of the associated IPsec security association. Alternatively, the host **MAY** use UPDATE packets as a substitute. A minimal UPDATE packet would consist of a SEQ and a single ECHO_REQ_SIGN parameter, and a more complex one would involve rekeying procedures as specified in Section 6.8 of [RFC7402]. It is worth noting that a host actively sending periodic UPDATE packets to a busy server may increase the computational load of the server since it has to verify HMACs and signatures in UPDATE messages.

5.4. NAT Traversal Mode Parameter

The format of the NAT traversal mode parameter is defined in Legacy ICE-HIP [RFC5770] but repeated here for completeness. The format of the NAT_TRAVERSAL_MODE parameter is similar to the format of the ESP_TRANSFORM parameter in [RFC7402] and is shown in Figure 8. The Native ICE-HIP extension specified in this document defines the new NAT traversal mode identifier for ICE-HIP-UDP and reuses the UDP-ENCAPSULATION mode from Legacy ICE-HIP [RFC5770]. The identifier named RESERVED is reserved for future use. Future specifications may define more traversal modes.

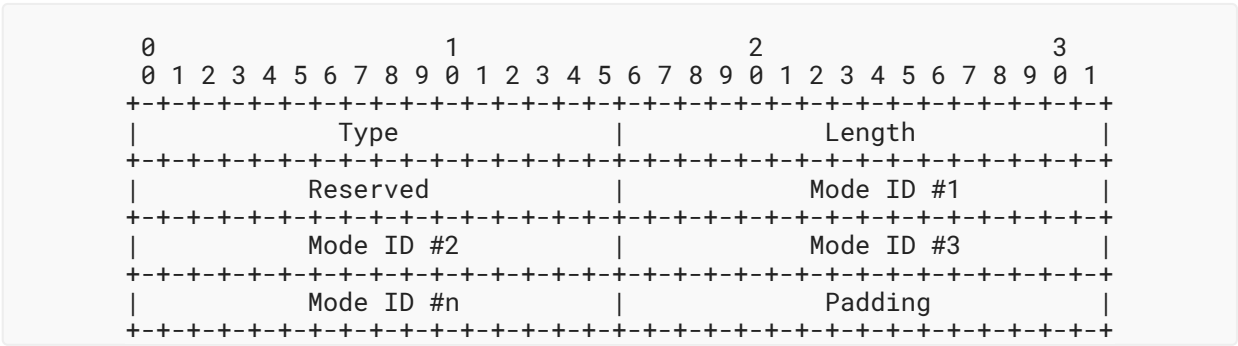

Figure 8: Format of the NAT_TRAVERSAL_MODE Parameter

Type: 608

Length: Length in octets, excluding Type, Length, and Padding

Reserved: Zero when sent, ignored when received

Mode ID: Defines the proposed or selected NAT traversal mode(s)

The following NAT traversal mode IDs are defined:

ID name	Value
RESERVED	0
UDP-ENCAPSULATION	1
ICE-STUN-UDP	2
ICE-HIP-UDP	3

Table 1: NAT Traversal Mode IDs

The sender of a NAT_TRAVERSAL_MODE parameter **MUST** make sure that there are no more than six (6) Mode IDs in one NAT_TRAVERSAL_MODE parameter. Conversely, a recipient **MUST** be prepared to handle received NAT traversal mode parameters that contain more than six Mode IDs by accepting the first six Mode IDs and dropping the rest. The limited number of Mode IDs sets the maximum size of the NAT_TRAVERSAL_MODE parameter. The modes **MUST** be in preference order, most preferred mode(s) first.

Implementations conforming to this specification **MUST** implement UDP-ENCAPSULATION and **SHOULD** implement ICE-HIP-UDP modes.

5.5. Connectivity Check Transaction Pacing Parameter

The TRANSACTION_PACING parameter is defined in [RFC5770] but repeated in Figure 9 for completeness. It contains only the connectivity check pacing value, expressed in milliseconds, as a 32-bit unsigned integer.

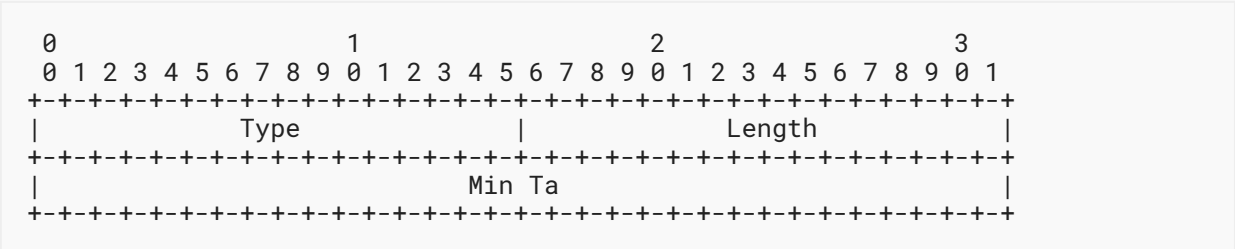

Figure 9: Format of the TRANSACTION_PACING Parameter

Type: 610

Length: 4

Min Ta: The minimum connectivity check transaction pacing value the host would use (in milliseconds)

5.6. Relay and Registration Parameters

The format of the REG_FROM, RELAY_FROM, and RELAY_TO parameters is shown in Figure 10. All parameters are identical except for the type. Of the three, only REG_FROM is covered by the signature.

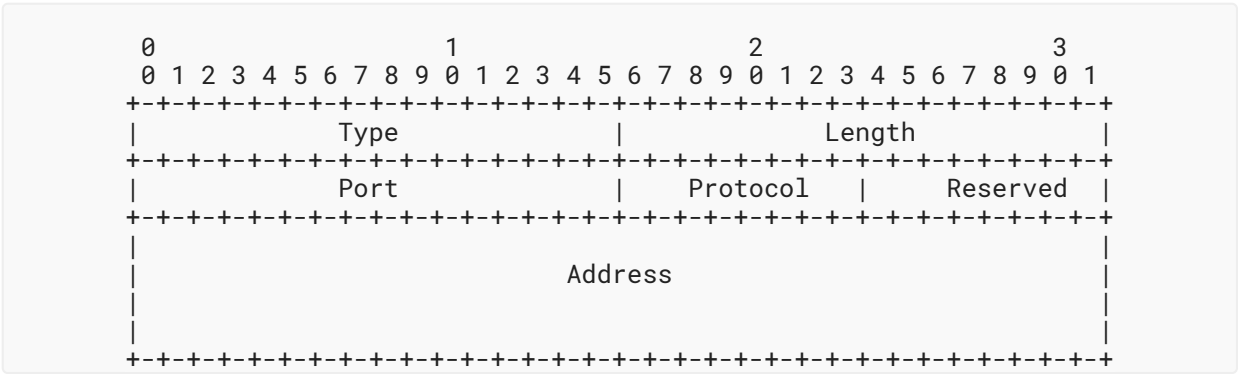

Figure 10: Format of the REG_FROM, RELAY_FROM, and RELAY_TO Parameters

Type: REG_FROM: 950
RELAY_FROM: 63998
RELAY_TO: 64002

Length: 20

Port: Transport port number; zero when plain IP is used

Protocol: IANA-assigned, Internet Protocol number. 17 for UDP; 0 for plain IP

Reserved: Reserved for future use; zero when sent, ignored when received

Address: An IPv6 address or an IPv4 address in "IPv4-mapped IPv6 address" format

REG_FROM contains the transport address and protocol from which the Control Relay Server sees the registration coming. RELAY_FROM contains the address from which the relayed packet was received by the Control Relay Server and the protocol that was used. RELAY_TO contains the same information about the address to which a packet should be forwarded.

5.7. LOCATOR_SET Parameter

This specification reuses the format for UDP-based locators as specified in Legacy ICE-HIP [RFC5770] to be used for communicating the address candidates between two hosts. The generic and NAT-traversal-specific locator parameters are illustrated in Figure 11.

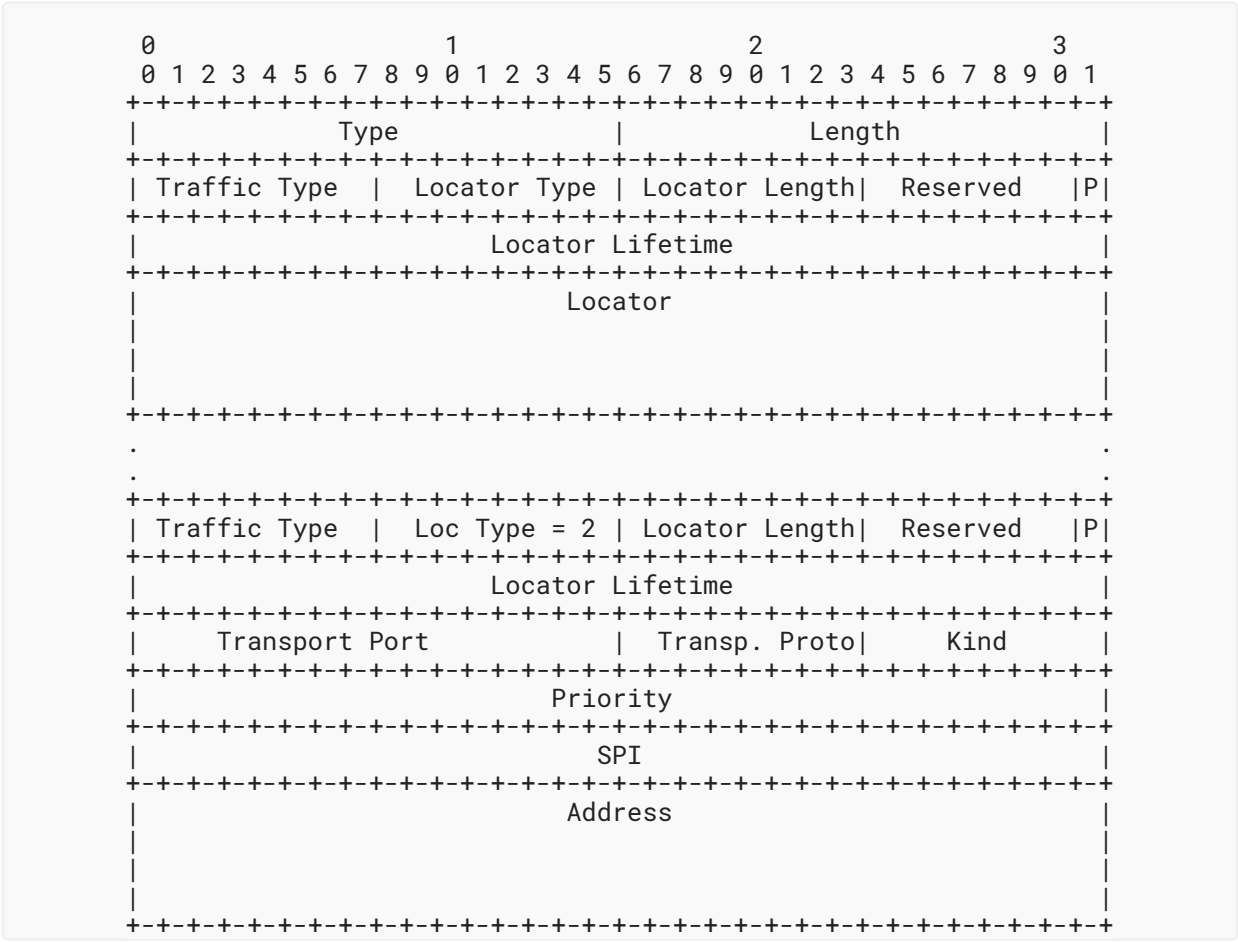

Figure 11: LOCATOR_SET Parameter

The individual fields in the LOCATOR_SET parameter are described in Table 2.

Field	Value(s)	Purpose
Type	193	Parameter type
Length	Variable	Length in octets, excluding Type and Length fields and padding
Traffic Type	0-2	The locator for either HIP signaling (1) or ESP (2), or for both (0)
Locator Type	2	"Transport address" locator type
Locator Length	7	Length of the fields after Locator Lifetime in 4-octet units

Field	Value (s)	Purpose
Reserved	0	Reserved for future extensions
Preferred (P) bit	0 or 1	Set to 1 for a Locator in R1 if the Responder can use it for the rest of the base exchange, otherwise set to zero
Locator Lifetime	Variable	Locator lifetime in seconds, see Section 4 of [RFC8046]
Transport Port	Variable	Transport-layer port number
Transport Protocol	Variable	IANA-assigned, transport-layer Internet Protocol number. Currently, only UDP (17) is supported.
Kind	Variable	0 for host, 1 for server reflexive, 2 for peer reflexive (currently unused), or 3 for relayed address
Priority	Variable	Locator's priority as described in [RFC8445]. It is worth noting that while the priority of a single locator candidate is 32 bits, an implementation should a 64-bit integer to calculate the priority of a candidate pair for the ICE priority algorithm.
SPI	Variable	Security Parameter Index (SPI) value that the host expects to see in incoming ESP packets that use this locator
Address	Variable	IPv6 address or an "IPv4-mapped IPv6 address" format IPv4 address [RFC4291]

Table 2: Fields of the LOCATOR_SET Parameter

The LOCATOR parameter **MUST** be encapsulated inside an ENCRYPTED parameter.

5.8. RELAY_HMAC Parameter

As specified in Legacy ICE-HIP [RFC5770], the RELAY_HMAC parameter value has the TLV type 65520. It has the same semantics as RVS_HMAC as specified in Section 4.2.1 of [RFC8004]. Similar to RVS_HMAC, RELAY_HMAC is also keyed with the HIP integrity key (HIP-lg or HIP-gl as specified in Section 6.5 of [RFC7401]), established during the relay registration procedure as described in Section 4.1.

5.9. Registration Types

The REG_INFO, REG_REQ, REG_RESP, and REG_FAILED parameters contain Registration Type [RFC8003] values for Control Relay Server registration. The value for RELAY_UDP_HIP is 2 as specified in Legacy ICE-HIP [RFC5770]. The value for RELAY_UDP_ESP is 3.

5.10. Notify Packet Types

A Control/Data Relay Server and end hosts can use NOTIFY packets to signal different error conditions. The NOTIFY packet types are the same as in Legacy ICE-HIP [RFC5770] except for the two last ones, which are new.

The Notify Packet Types [RFC7401] are shown below. The Notification Data field for the error notifications **SHOULD** contain the HIP header of the rejected packet and **SHOULD** be empty for the CONNECTIVITY_CHECKS_FAILED type.

NOTIFICATION PARAMETER - ERROR TYPES	Value
NO_VALID_NAT_TRAVERSAL_MODE_PARAMETER If a Control Relay Server does not forward a base exchange packet due to a missing NAT traversal mode parameter, or the Initiator selects a NAT traversal mode that the (non-relay) Responder did not expect, the Control Relay Server or the Responder may send back a NOTIFY error packet with this type.	60
CONNECTIVITY_CHECKS_FAILED Used by the end hosts to signal that NAT traversal connectivity checks failed and did not produce a working path.	61
MESSAGE_NOT_RELAYED Used by a Control Relay Server to signal that it was not able or willing to relay a HIP packet.	62
SERVER_REFLEXIVE_CANDIDATE_ALLOCATION_FAILED Used by a Data Relay Server to signal that it was not able or willing to allocate a new server-reflexive candidate for the Data Relay Client.	63
RVS_HMAC_PROHIBITED_WITH_RELAY In the unintended event that a Control Relay Server sends any HIP message with an RVS_HMAC parameter, the Control Relay Client drops the received HIP message and sends a notify message back to the Control Relay Server using this notify type.	64

Table 3: Notify Packet Types

5.11. ESP Data Packets

The format for ESP data packets is identical to Legacy ICE-HIP [RFC5770].

[RFC3948] describes the UDP encapsulation of the IPsec ESP transport and tunnel mode. On the wire, the HIP ESP packets do not differ from the transport mode ESP; thus, the encapsulation of the HIP ESP packets is same as the UDP encapsulation transport mode ESP. However, the (semantic) difference to Bound End-to-End Tunnel (BEET) mode ESP packets used by HIP is that the IP header is not used in BEET integrity protection calculation.

During the HIP base exchange, the two peers exchange parameters that enable them to define a pair of IPsec ESP security associations (SAs) as described in [RFC7402]. When two peers perform a UDP-encapsulated base exchange, they **MUST** define a pair of IPsec SAs that produces UDP-encapsulated ESP data traffic.

The management of encryption/authentication protocols and SPIs is defined in [RFC7402]. The UDP encapsulation format and processing of HIP ESP traffic is described in Section 6.1 of [RFC7402].

5.12. RELAYED_ADDRESS and MAPPED_ADDRESS Parameters

While the type values are new, the format of the RELAYED_ADDRESS and MAPPED_ADDRESS parameters (Figure 12) is identical to REG_FROM, RELAY_FROM, and RELAY_TO parameters. This document specifies only the use of UDP relaying; thus, only protocol 17 is allowed. However, future documents may specify support for other protocols.

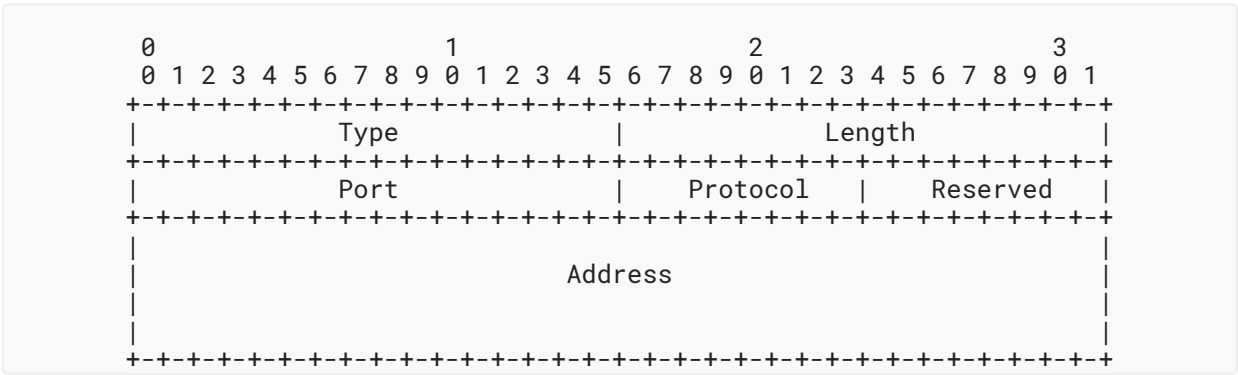

Figure 12: Format of the RELAYED_ADDRESS and MAPPED_ADDRESS Parameters

Type: RELAYED_ADDRESS: 4650
MAPPED_ADDRESS: 4660

Length: 20

Port: The UDP port number

Protocol: IANA-assigned, Internet Protocol number (17 for UDP)

Reserved: Reserved for future use; zero when sent, ignored when received

Address: An IPv6 address or an IPv4 address in "IPv4-mapped IPv6 address" format

5.13. PEER_PERMISSION Parameter

The format of the new PEER_PERMISSION parameter is shown in Figure 13. The parameter is used for setting up and refreshing forwarding rules and the permissions for data packets at the Data Relay Server. The parameter contains one or more sets of Port, Protocol, Address, Outbound SPI (OSPI), and Inbound SPI (ISPI) values. One set defines a rule for one peer address.

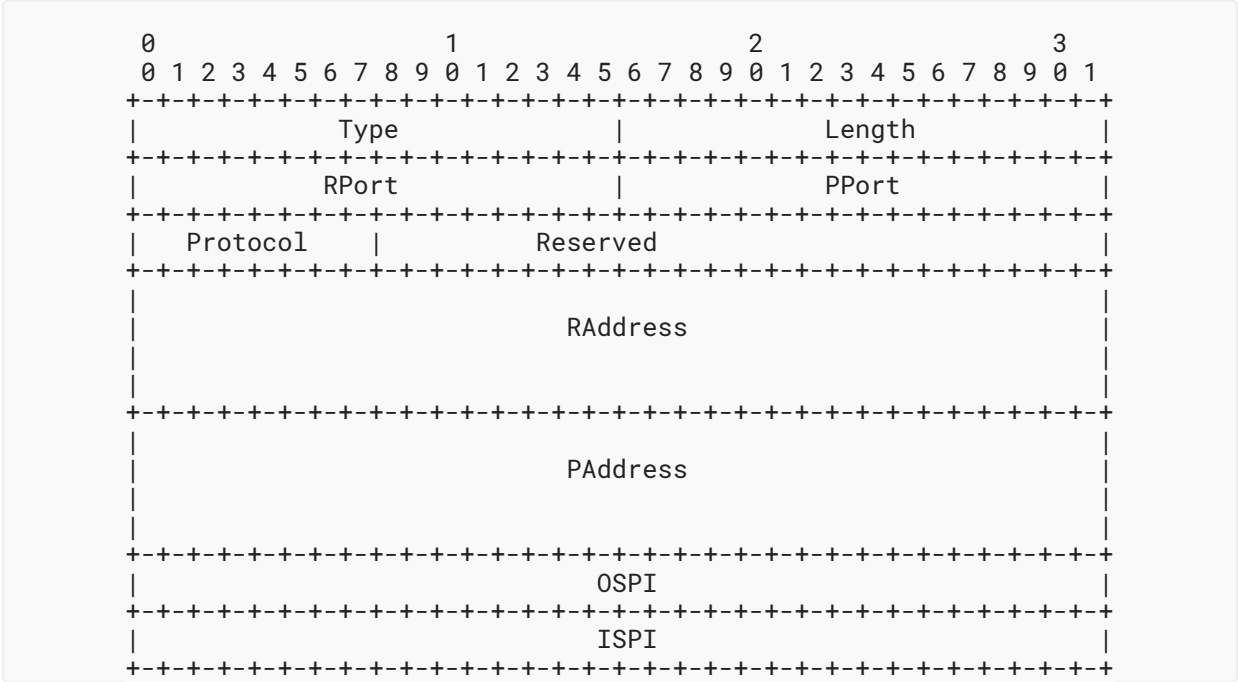

Figure 13: Format of the PEER_PERMISSION Parameter

Type: 4680

Length: 48

RPort: The transport-layer (UDP) port at the Data Relay Server (i.e., the port of the server-reflexive candidate)

PPort: The transport-layer (UDP) port number of the peer

Protocol: IANA-assigned, Internet Protocol number (17 for UDP)

Reserved: Reserved for future use; zero when sent, ignored when received

RAddress: An IPv6 address, or an IPv4 address in "IPv4-mapped IPv6 address" format, of the server-reflexive candidate

PAddress: An IPv6 address, or an IPv4 address in "IPv4-mapped IPv6 address" format, of the peer

OSPI: The outbound SPI value the Data Relay Client is using for the peer

ISPI: The inbound SPI value the Data Relay Client is using for the peer

5.14. HIP Connectivity Check Packets

The connectivity request messages are HIP UPDATE packets containing a new CANDIDATE_PRIORITY parameter (Figure 14). Response UPDATE packets contain a MAPPED_ADDRESS parameter (Figure 12).

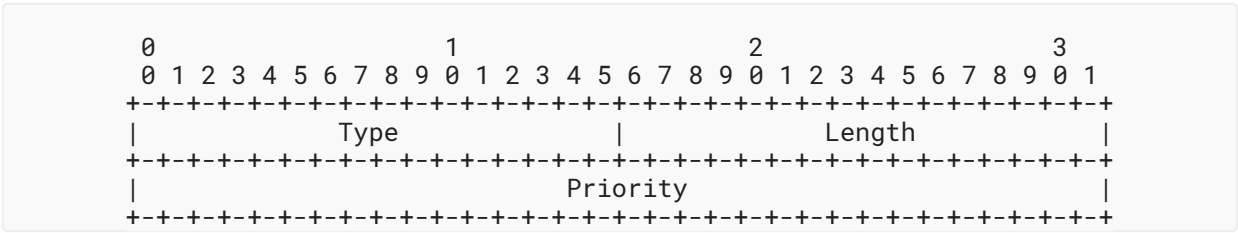

Figure 14: Format of the CANDIDATE_PRIORITY Parameter

Type: 4700

Length: 4

Priority: The priority of a (potential) peer-reflexive candidate

5.15. NOMINATE Parameter

Figure 15 shows the NOMINATE parameter that is used to conclude the candidate nomination process.

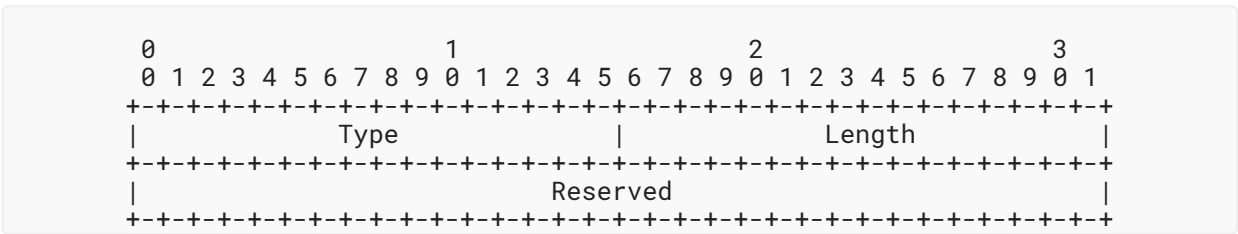

Figure 15: Format of the NOMINATE Parameter

Type: 4710

Length: 4

Reserved: Reserved for future extension purposes

6. IAB Considerations

The ICE specification [RFC8445] discusses "Unilateral Self-Address Fixing" in Section 18. This protocol is based on ICE; thus, the same considerations also apply here.

7. Security Considerations

Since the control plane protocol and Control Relay Server are essentially the same (with some minor differences) in this document as in Legacy ICE-HIP [RFC5770], the same security considerations (in Sections 7.1, 7.2, 7.3, and 7.4) are still valid, but are repeated here for the sake of completeness. New security considerations related to the new Data Relay Server are discussed in Section 7.5, and considerations related to the new connectivity check protocol are discussed in Sections 7.6 and 7.7.

7.1. Privacy Considerations

It is also possible that end users may not want to reveal all locators to each other. For example, tracking the physical location of a multihoming end host may become easier if it reveals all locators to its peer during a base exchange. Also, revealing host addresses exposes information about the local topology that may not be allowed in all corporate environments. For these two local policy reasons, it might be tempting to exclude certain host addresses from the LOCATOR_SET parameter of an end host, but this is **NOT RECOMMENDED**. For instance, such behavior creates non-optimal paths when the hosts are located behind the same NAT. Especially, this could be problematic with a legacy NAT that does not support routing from the private address realm back to itself through the outer address of the NAT. This scenario is referred to as the hairpin problem [RFC5128]. With such a legacy NAT, the only option left would be to use a relayed transport address from a Data Relay Server.

The use of Control and Data Relay Servers can also be useful for privacy purposes. For example, a privacy-concerned Responder may reveal only its Control Relay Server and Relayed candidates to Initiators. This partially protects the Responder against Denial-of-Service (DoS) attacks by allowing the Responder to initiate new connections even if its relays would be unavailable due to a DoS attack.

7.2. Opportunistic Mode

In opportunistic HIP mode (cf. Section 4.1.8 of [RFC7401]), an Initiator sends an I1 without setting the destination HIT of the Responder (i.e., the Control Relay Client). A Control Relay Server **SHOULD** have a unique IP address per the Control Relay Client when the Control Relay Server is serving more than one Control Relay Client and supports opportunistic mode. Otherwise, the Control Relay Server cannot guarantee to deliver the I1 packet to the intended recipient. Future extensions of this document may allow opportunistic mode to be used with non-unique IP addresses to be utilized either as a HIP-level anycast or multicast mechanism. Both of the mentioned cases would require separate registration parameters that the Control Relay Server proposes and the Control Client Server accepts during registration.

7.3. Base Exchange Replay Protection for Control Relay Server

In certain scenarios, it is possible that an attacker, or two attackers, can replay an earlier base exchange through a Control Relay Server by masquerading as the original Initiator and Responder. The attack does not require the attacker(s) to compromise the private key(s) of the attacked host(s). However, for this attack to succeed, the legitimate Responder has to be disconnected from the Control Relay Server.

The Control Relay Server can protect itself against replay attacks by becoming involved in the base exchange by introducing nonces that the end hosts (Initiator and Responder) are required to sign. One way to do this is to add ECHO_REQUEST_M parameters to the R1 and I2 packets as described in [HIP-MIDDLEBOXES] and drop the I2 or R2 packets if the corresponding ECHO_RESPONSE_M parameters are not present.

7.4. Demultiplexing Different HIP Associations

Section 5.1 of [RFC3948] describes a security issue for the UDP encapsulation in the standard IP tunnel mode when two hosts behind different NATs have the same private IP address and initiate communication to the same Responder in the public Internet. The Responder cannot distinguish between two hosts because security associations are based on the same inner IP addresses.

This issue does not exist with the UDP encapsulation of HIP ESP transport format because the Responder uses HITs to distinguish between different Initiators.

7.5. Reuse of Ports at the Data Relay Server

If the Data Relay Server uses the same relayed address and port (as conveyed in the RELAYED_ADDRESS parameter) for multiple Data Relay Clients, it appears to all the peers, and their firewalls, that all the Data Relay Clients are at the same address. Thus, a stateful firewall may allow packets to pass from hosts that would not normally be able to send packets to a peer behind the firewall. Therefore, a Data Relay Server **SHOULD NOT** reuse the port numbers. If port numbers need to be reused, the Data Relay Server **SHOULD** have a sufficiently large pool of port numbers and randomly select ports from the pool to decrease the chances of a Data Relay Client obtaining the same address that another host behind the same firewall is using.

7.6. Amplification Attacks

A malicious host may send an invalid list of candidates to its peer that are used for targeting a victim host by flooding it with connectivity checks. To mitigate the attack, this protocol adopts the ICE mechanism to cap the total amount of connectivity checks as defined in Section 4.7.

7.7. Attacks against Connectivity Checks and Candidate Gathering

Section 19.2 of [RFC8445] describes attacks against ICE connectivity checks. HIP bases its control plane security on Diffie-Hellman key exchange, public keys, and Hashed Message Authentication codes, meaning that the mentioned security concerns do not apply to HIP either. The mentioned

section also discusses man-in-the-middle replay attacks that are difficult to prevent. The connectivity checks in this protocol are effectively immune against replay attacks because a connectivity request includes a random nonce that the recipient must sign and send back as a response.

Section 19.3 of [RFC8445] describes attacks on server-reflexive address gathering. Similarly here, if the DNS, a Control Relay Server, or a Data Relay Server has been compromised, not much can be done. However, the case where attackers can inject fake messages (located on a shared network segment like Wi-Fi) does not apply here. HIP messages are integrity and replay protected, so it is not possible to inject fake server-reflexive address candidates.

Section 19.4 of [RFC8445] describes attacks on relayed candidate gathering. Similarly to ICE TURN servers, a Data Relay Server requires an authenticated base exchange that protects relayed address gathering against fake requests and responses. Further, replay attacks are not possible because the HIP base exchange (and also UPDATE procedure) is protected against replay attacks.

7.8. Cross-Protocol Attacks

Section 4.1 explains how a Control Relay Client registers for the RELAY_UDP_HIP service from a Control Relay Server. However, the same server may also offer Rendezvous functionality; thus, a client can register both to a RELAY_UDP_HIP and a RENDEZVOUS (see [RFC8004]) service from the same server. Potentially, this introduces a cross-protocol attack (or actually a "cross-message" attack) because the key material is the same for the Control Relay Service and Rendezvous HMACs. While the problem could be avoided by deriving different keys for the Control Relay Service, a more simple measure was chosen because the exact attack scenario was unclear. Consequently, this section defines a mandatory mitigation mechanism against the cross-protocol attack that works by preventing the simultaneous use of Rendezvous and Control Relay Service in the context of a single HIP Association.

The registration involves three parameters typically delivered sequentially in R1 (REG_INFO parameter), I2 (REG_REQUEST), and R2 (REG_RESPONSE) messages but can also be delivered in UPDATE messages as described in [RFC8003]. The parameters and the modifications to their processing are described below:

REG_INFO: The Control Relay Server advertises its available services using this parameter. RELAY_UDP_HIP and RENDEZVOUS services **MAY** be included in the first advertisement for the HIP association, but subsequent ones **MUST** include only one of them as agreed in earlier registrations (see steps 2 and 3).

REG_REQUEST: The Control Relay Client chooses the services it requires using this parameter. If the Control Relay Server offered both RENDEZVOUS or RELAY_UDP_HIP, the Control Relay Client **MUST** choose only one of them in the REG_REQUEST parameter. Upon choosing one of the two, it persists throughout the lifetime of the HIP association, and the Control Relay Client **MUST NOT** register the other remaining one in a subsequent UPDATE message.

REG_RESPONSE: The Control Relay Server verifies the services requested by the Control Relay Client using this parameter. If the Control Relay Server offered both RENDEZVOUS and RELAY_UDP_HIP service, and the Control Relay Client requested for both of them, the Control Relay Client **MUST** offer only RELAY_UDP_HIP service in the REG_RESPONSE parameter and include a REG_FAILED parameter in the same message, with RENDEZVOUS as the Registration Type and 9 as the Failure Type.

As a further measure against cross-protocol attacks, the Control Relay Client **MUST** drop any HIP message that includes an RVS_HMAC parameter when it originates from a successfully registered Control Relay Server. Upon such an (unintended) event, the Control Relay Client **MUST** send a NOTIFY message with RVS_HMAC_PROHIBITED_WITH_RELAY as the Notify Message Type to the Control Relay Server.

8. IANA Considerations

This section is to be interpreted according to [RFC8126].

This document reuses the same default UDP port number 10500 as specified by Legacy ICE-HIP [RFC5770] for tunneling both HIP control and data plane traffic. The port was registered separately for [RFC5770] to coauthor Ari Keränen originally, but it has been reassigned for IESG control. With the permission of Ari Keränen, the new assignee is the IESG and the contact is <chair@ietf.org>. In addition, IANA has added a reference to this document in the entry for UDP port 10500 in the "Service Name and Transport Protocol Port Number Registry". The selection between Legacy ICE-HIP and Native ICE-HIP mode is negotiated using the NAT_TRAVERSAL_MODE parameter during the base exchange. By default, hosts listen to this port for incoming UDP datagrams and can also use it for sending UDP datagrams. Other ephemeral port numbers are negotiated and utilized dynamically.

IANA has assigned the following values in the HIP "Parameter Types" registry [RFC7401]: 4650 for RELAYED_ADDRESS (length 20), 4660 for MAPPED_ADDRESS (length 20; defined in Section 5.12), 4680 for PEER_PERMISSION (length 48; defined in Section 5.13), 4700 for CANDIDATE_PRIORITY (length 4; defined in Section 5.14), and 4710 for NOMINATE (length 4; defined in Section 5.15).

IANA has assigned the following value in the "HIP NAT Traversal Modes" registry specified in Legacy ICE-HIP [RFC5770]: 3 for ICE-HIP-UDP (defined in Section 5.4).

IANA has assigned the following values in the HIP "Notify Message Types" registry: 16385 for NAT_KEEPALIVE in Section 5.3, 63 for SERVER_REFLEXIVE_CANDIDATE_ALLOCATION_FAILED in Section 5.10, and 64 for RVS_HMAC_PROHIBITED_WITH_RELAY in Section 5.10.

IANA has assigned the following values in the "Registration Types" registry for the HIP Registration Extension [RFC8003]: 3 for RELAY_UDP_ESP (defined in Section 5.9) for allowing registration with a Data Relay Server for ESP-relaying service, and 4 for CANDIDATE_DISCOVERY (defined in Section 4.2) for performing server-reflexive candidate discovery.

IANA has assigned one value in the "Registration Failure Types" registry as defined in Section 7.8. The value is 9, and the Registration Failure Type is "Simultaneous Rendezvous and Control Relay Service usage prohibited".

9. References

9.1. Normative References

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <https://www.rfc-editor.org/info/rfc1191>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <https://www.rfc-editor.org/info/rfc4291>.

[RFC5770] Komu, M., Henderson, T., Tschofenig, H., Melen, J., and A. Keranen, Ed., "Basic Host Identity Protocol (HIP) Extensions for Traversal of Network Address Translators", RFC 5770, DOI 10.17487/RFC5770, April 2010, <https://www.rfc-editor.org/info/rfc5770>.

[RFC7050] Savolainen, T., Korhonen, J., and D. Wing, "Discovery of the IPv6 Prefix Used for IPv6 Address Synthesis", RFC 7050, DOI 10.17487/RFC7050, November 2013, <https://www.rfc-editor.org/info/rfc7050>.

[RFC7401] Moskowitz, R., Ed., Heer, T., Jokela, P., and T. Henderson, "Host Identity Protocol Version 2 (HIPv2)", RFC 7401, DOI 10.17487/RFC7401, April 2015, <https://www.rfc-editor.org/info/rfc7401>.

[RFC7402] Jokela, P., Moskowitz, R., and J. Melen, "Using the Encapsulating Security Payload (ESP) Transport Format with the Host Identity Protocol (HIP)", RFC 7402, DOI 10.17487/RFC7402, April 2015, <https://www.rfc-editor.org/info/rfc7402>.

[RFC8003] Laganier, J. and L. Eggert, "Host Identity Protocol (HIP) Registration Extension", RFC 8003, DOI 10.17487/RFC8003, October 2016, <https://www.rfc-editor.org/info/rfc8003>.

[RFC8004] Laganier, J. and L. Eggert, "Host Identity Protocol (HIP) Rendezvous Extension", RFC 8004, DOI 10.17487/RFC8004, October 2016, <https://www.rfc-editor.org/info/rfc8004>.

[RFC8005] Laganier, J., "Host Identity Protocol (HIP) Domain Name System (DNS) Extension", RFC 8005, DOI 10.17487/RFC8005, October 2016, <https://www.rfc-editor.org/info/rfc8005>.

[RFC8046] Henderson, T., Ed., Vogt, C., and J. Arkko, "Host Mobility with the Host Identity Protocol", RFC 8046, DOI 10.17487/RFC8046, February 2017, <https://www.rfc-editor.org/info/rfc8046>.

[RFC8047] Henderson, T., Ed., Vogt, C., and J. Arkko, "Host Multihoming with the Host Identity Protocol", RFC 8047, DOI 10.17487/RFC8047, February 2017, <https://www.rfc-editor.org/info/rfc8047>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8201] McCann, J., Deering, S., Mogul, J., and R. Hinden, Ed., "Path MTU Discovery for IP version 6", STD 87, RFC 8201, DOI 10.17487/RFC8201, July 2017, <https://www.rfc-editor.org/info/rfc8201>.

[RFC8445] Keranen, A., Holmberg, C., and J. Rosenberg, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal", RFC 8445, DOI 10.17487/RFC8445, July 2018, <https://www.rfc-editor.org/info/rfc8445>.

[RFC8489] Petit-Huguenin, M., Salgueiro, G., Rosenberg, J., Wing, D., Mahy, R., and P. Matthews, "Session Traversal Utilities for NAT (STUN)", RFC 8489, DOI 10.17487/RFC8489, February 2020, <https://www.rfc-editor.org/info/rfc8489>.

[RFC8961] Allman, M., "Requirements for Time-Based Loss Detection", BCP 233, RFC 8961, DOI 10.17487/RFC8961, November 2020, <https://www.rfc-editor.org/info/rfc8961>.

9.2. Informative References

[HIP-MIDDLEBOXES] Heer, T., Hummen, R., Wehrle, K., and M. Komu, "End-Host Authentication for HIP Middleboxes", Work in Progress, Internet-Draft, draft-heer-hip-middle-auth-04, 31 October 2011, <https://tools.ietf.org/html/draft-heer-hip-middle-auth-04>.

[ICE-NONSIP] Rosenberg, J., "Guidelines for Usage of Interactive Connectivity Establishment (ICE) by non Session Initiation Protocol (SIP) Protocols", Work in Progress, Internet-Draft, draft-rosenberg-mmusic-ice-nonsip-01, 14 July 2008, <https://tools.ietf.org/html/draft-rosenberg-mmusic-ice-nonsip-01>.

[RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, DOI 10.17487/RFC2475, December 1998, <https://www.rfc-editor.org/info/rfc2475>.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, DOI 10.17487/RFC3264, June 2002, <https://www.rfc-editor.org/info/rfc3264>.

[RFC3948] Huttunen, A., Swander, B., Volpe, V., DiBurro, L., and M. Stenberg, "UDP Encapsulation of IPsec ESP Packets", RFC 3948, DOI 10.17487/RFC3948, January 2005, <https://www.rfc-editor.org/info/rfc3948>.

[RFC5128] Srisuresh, P., Ford, B., and D. Kegel, "State of Peer-to-Peer (P2P) Communication across Network Address Translators (NATs)", RFC 5128, DOI 10.17487/RFC5128, March 2008, <https://www.rfc-editor.org/info/rfc5128>.

[RFC5207] Stiemerling, M., Quittek, J., and L. Eggert, "NAT and Firewall Traversal Issues of Host Identity Protocol (HIP) Communication", RFC 5207, DOI 10.17487/RFC5207, April 2008, <https://www.rfc-editor.org/info/rfc5207>.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, DOI 10.17487/RFC5245, April 2010, <https://www.rfc-editor.org/info/rfc5245>.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146, April 2011, <https://www.rfc-editor.org/info/rfc6146>.

[RFC6147] Bagnulo, M., Sullivan, A., Matthews, P., and I. van Beijnum, "DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers", RFC 6147, DOI 10.17487/RFC6147, April 2011, <https://www.rfc-editor.org/info/rfc6147>.

[RFC6538] Henderson, T. and A. Gurtov, "The Host Identity Protocol (HIP) Experiment Report", RFC 6538, DOI 10.17487/RFC6538, March 2012, <https://www.rfc-editor.org/info/rfc6538>.

[RFC8656] Reddy, T., Ed., Johnston, A., Ed., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 8656, DOI 10.17487/RFC8656, February 2020, <https://www.rfc-editor.org/info/rfc8656>.

[RFC8750] Migault, D., Guggemos, T., and Y. Nir, "Implicit Initialization Vector (IV) for Counter-Based Ciphers in Encapsulating Security Payload (ESP)", RFC 8750, DOI 10.17487/RFC8750, March 2020, <https://www.rfc-editor.org/info/rfc8750>.

[RFC8899] Fairhurst, G., Jones, T., Tüxen, M., Rüngeler, I., and T. Völker, "Packetization Layer Path MTU Discovery for Datagram Transports", RFC 8899, DOI 10.17487/RFC8899, September 2020, <https://www.rfc-editor.org/info/rfc8899>.

[RFC9068] Moskowitz, R., Ed. and M. Komu, "Host Identity Protocol Architecture", RFC 9063, DOI 10.17487/RFC9068, June 2021, <https://www.rfc-editor.org/info/rfc9068>.

Appendix A. Selecting a Value for Check Pacing

Selecting a suitable value for the connectivity check transaction pacing is essential for the performance of connectivity check-based NAT traversal. The value should not be so small that the checks cause network congestion or overwhelm the NATs. On the other hand, a pacing value that is too high makes the checks last for a long time, thus increasing the connection setup delay.

The Ta value may be configured by the user in environments where the network characteristics are known beforehand. However, if the characteristics are not known, it is recommended that the value is adjusted dynamically. In this case, it is recommended that the hosts estimate the round-trip time (RTT) between them, and they **SHOULD** set the minimum Ta value so that at most a single connectivity check message is sent on every RTT.

One way to estimate the RTT is to use the time that it takes for the Control Relay Server registration exchange to complete; this would give an estimate on the registering host's access link's RTT. Also, the I1/R1 exchange could be used for estimating the RTT, but since the R1 can be cached in the network, or the relaying service can increase the delay notably, this is not recommended. In general, estimating RTT can be difficult and error prone; thus, the guidelines for choosing a Ta value in Section 4.4 **MUST** be followed.

Appendix B. Differences with Respect to ICE

Legacy ICE-HIP reuses the ICE/STUN/TURN protocol stack as it is. The benefits of such as an approach include the reuse of STUN/TURN infrastructure and possibly the reuse of existing software libraries, but there are also drawbacks with the approach. For example, ICE is meant for application-layer protocols, whereas HIP operates at layer 3.5 between transport and network layers. This is particularly problematic because the implementations employ kernel-space IPsec ESP as their data plane: demultiplexing of incoming ESP, HIP, and TURN messages required the capturing of all UDP packets destined to port 10500 to the userspace (due to different, incompatible markers in ESP and STUN), thus causing additional software complexity and an unnecessary latency/throughput bottleneck for the dataplane performance. It is also worth noting that the demultiplexing of STUN packets in the kernel would also incur a performance impact (albeit smaller than with userspace demultiplexing), and secure verification of STUN messages would require communication between the kernel-space STUN detector and HIP daemon typically residing in the userspace (thus again increasing the performance overhead).

Legacy ICE-HIP also involves some other complexities when compared to the approach taken in this document. The relaying of ESP packets via TURN relays was not considered that simple because TURN relays require adding and removing extra TURN framing for the relayed packets. Finally, the developers of the two Legacy ICE-HIP implementations concluded that effort needed for integrating an ICE library into a HIP implementation turned out to be quite a bit higher than initially estimated. Also, the amount of extra code (some 10 kLoC) needed for all the new parsers, state machines, etc., was quite high and by reusing the HIP code, one should be able to do with much less. This should result in smaller binary size, less bugs, and easier debugging.

Consequently, the HIP working group decided to follow ICE methodology but reuse HIP messaging format to achieve the same functionality as ICE; the result of that is this document, which specifies the Native ICE-HIP protocol.

The Native ICE-HIP protocol specified in this document follows the semantics of ICE as close as possible, and most of the differences are syntactical due to the use of a different protocol. In this section, we describe the differences to the ICE protocol.

- ICE operates at the application layer, whereas this protocol operates between transport and network layers, thus hiding the protocol details from the application.
- The STUN protocol is not employed. Instead, Native ICE-HIP reuses the HIP control plane format in order to simplify the demultiplexing of different protocols. For example, the STUN binding response is replaced with a HIP UPDATE message containing an ECHO_REQUEST_SIGNED parameter and the STUN binding response with a HIP UPDATE message containing an ECHO_RESPONSE_SIGNED parameter as defined in Section 4.6. It is worth noting that a drawback of not employing STUN is that discovery of the address candidates requires creating (using HIP base exchange) and maintaining (using HIP UPDATE procedures) state at the Control Relay Client and Control Relay Server. Future extensions to this document may define a stateless, HIP-specific mechanism for an end host to discover its address candidates.
- The TURN protocol is not utilized. Instead, Native ICE-HIP reuses Control Relay Servers for the same purpose.
- ICMP errors may be used in ICE to signal failure. In the Native ICE-HIP protocol, HIP NOTIFY messages are used instead.
- Instead of the ICE username fragment and password mechanism for credentials, Native ICE-HIP uses the HIT, derived from a public key, for the same purpose. The username fragments are "transient host identifiers, bound to a particular session established as part of the candidate exchange" [RFC8445]. Generally in HIP, a local public key and the derived HIT are considered long-term identifiers and invariant across different host associations and different transport-layer flows.
- In ICE, the conflict when two communicating endpoints take the same controlling role is solved using random values (a so-called tie-breaker value). In the Native ICE-HIP protocol, the conflict is solved by the standard HIP base exchange procedure, where the host with the "larger" HIT switches to the Responder role, thus also changing to the controlled role.
- The ICE-CONTROLLED and ICE-CONTROLLING attributes are not included in the connectivity checks.
- The foundation concept is unnecessary in Native ICE-HIP because only a single UDP flow for the IPsec tunnel will be negotiated.
- Frozen candidates are omitted for the same reason the foundation concept is excluded.
- Components are omitted for the same reason the foundation concept is excluded.
- Native ICE-HIP supports only "full ICE" where the two communicating hosts participate actively to the connectivity checks, and the "lite" mode is not supported. This design decision follows the guidelines of ICE, which recommends full ICE implementations. However, it should be noted that a publicly reachable Responder may refuse to negotiate the ICE mode as

described in Section 4.7.2. This would result in a HIP base exchange (as per [RFC7401]) tunneled over UDP, followed by ESP traffic over the same tunnel, without the connectivity check procedures defined in this document (in some sense, this mode corresponds to the case where two ICE lite implementations connect since no connectivity checks are sent).

- As the "ICE lite" is not adopted here and both sides are capable of ICE-HIP-UDP mode (negotiated during the base exchange), default candidates are not employed in Native ICE-HIP.
- If the agent is using Diffserv Codepoint markings [RFC2475] in its media packets, it **SHOULD** apply those same markings to its connectivity checks.
- Unlike in ICE, the addresses are not XORed in the Native ICE-HIP protocol but rather encrypted to avoid middlebox tampering.
- ICE defines Related Address and Port attributes used for diagnostic/SIP purposes, but the Native ICE-HIP protocol does not employ these attributes.
- The minimum RTO is 500 ms in ICE but 1000 ms in the Native ICE-HIP protocol in favor of [RFC8961].

Appendix C. Differences to Base Exchange and UPDATE Procedures

This section gives some design guidance for implementers on how the extensions in this protocol extend and differ from [RFC7401] and [RFC8046].

- Both the control and data plane are operated on top of UDP, not directly on IP.
- A minimal implementation would conform only to Sections 4.7.1 or 4.7.2, thus merely tunneling HIP control and data traffic over UDP. The drawback here is that it works only in the limited cases where the Responder has a public address.
- It is worth noting that while a Rendezvous Server [RFC8004] has not been designed to be used in NATed scenarios because it just relays the first I1 packet and does not employ UDP encapsulation, the Control Relay Server forwards all control traffic and, hence, is more suitable in NATed environments. Further, the Data Relay Server guarantees forwarding of data plane traffic also in cases where the NAT traversal procedures fail.
- Registration procedures with a Control/Data Relay Server are similar as with a Rendezvous Server. However, a Control/Data Relay Server has different registration parameters than a Rendezvous Server because it offers a different service. Also, the Control/Data Relay Server also includes a REG_FROM parameter that informs the Control/Data Relay Client about its server-reflexive address. A Data Relay Server also includes a RELAYED_ADDRESS containing the relayed address for the Data Relay Client.
- In [RFC7401], the Initiator and Responder can start to exchange application payload immediately after the base exchange. While exchanging data immediately after a base exchange via a Data Control Relay would also be possible here, we follow the ICE methodology to establish a direct path between two hosts using connectivity checks. This means that there will be some additional delay after the base exchange before application

payload can be transmitted. The same applies for the UPDATE procedure as the connectivity checks introduce some additional delay.

- In HIP without any NAT traversal support, the base exchange acts as an implicit connectivity check, and the mobility and multihoming extensions support explicit connectivity checks. After a base exchange or UPDATE-based connectivity checks, a host can use the associated address pair for transmitting application payload. In this Native ICE-HIP extension, we follow the ICE methodology where one endpoint acting in the controlled role chooses the used address pair also on behalf of the other endpoint acting in the controlled role, which is different from HIP without NAT traversal support. Another difference is that the process of choosing an address pair is explicitly signaled using the nomination packets. The nomination process in this protocol supports only a single address pair, and multihoming extensions are left for further study.
- The UPDATE procedure resembles the mobility extensions defined in [RFC8046]. The first UPDATE message from the mobile host is exactly the same as in the mobility extensions. The second UPDATE message from the peer host and third from the mobile host are different in the sense that they merely acknowledge and conclude the reception of the candidates through the Control Relay Server. In other words, they do not yet test for connectivity (besides reachability through the Control Relay Server) unlike in the mobility extensions. The idea is that the connectivity check procedure follows the ICE specification, which is somewhat different from the HIP mobility extensions.
- The connectivity checks as defined in the mobility extensions [RFC8046] are triggered only by the peer of the mobile host. Since successful NAT traversal requires that both endpoints test connectivity, both the mobile host and its peer host have to test for connectivity. In addition, this protocol also validates the UDP ports; the ports in the connectivity check must match with the response, as required by ICE.
- In HIP mobility extensions [RFC8046], an outbound locator has some associated state: UNVERIFIED means that the locator has not been tested for reachability, ACTIVE means that the address has been verified for reachability and is being used actively, and DEPRECATED means that the locator lifetime has expired. In the subset of ICE specifications used by this protocol, an individual address candidate has only two properties: type and priority. Instead, the actual state in ICE is associated with candidate pairs rather than individual addresses. The subset of ICE specifications utilized by this protocol require the following attributes for a candidate pair: valid bit, nominated bit, base, and the state of the connectivity check. The connectivity checks have the following states: Waiting, In-progress, Succeeded, and Failed. Handling of this state attribute requires some additional logic when compared to the mobility extensions, since the state is associated with a local-remote address pair rather than just a remote address; thus, the mobility and ICE states do not have an unambiguous one-to-one mapping.
- Credit-based authorization as defined in [RFC8046] could be used before candidate nomination has been concluded upon discovering working candidate pairs. However, this may result in the use of asymmetric paths for a short time period in the beginning of communications. Thus, support of credit-based authorization is left for further study.

Appendix D. Multihoming Considerations

This document allows a host to collect address candidates from multiple interfaces but does not support activation and the simultaneous use of multiple address candidates. While multihoming extensions to support functionality similar to that found in [RFC8047] are left for further study and experimentation, we envision here some potential compatibility improvements to support multihoming:

Data Relay Registration: a Data Relay Client acting as an Initiator with another peer host should register a new server-reflexive candidate for each local transport address candidate. A Data Relay Client acting as a Responder should register a new server-reflexive candidate for each {local transport address candidate, new peer host} pair for the reasons described in Section 4.12.3. In both cases, the Data Relay Client should request the additional server-reflexive candidates by sending UPDATE messages originating from each of the local address candidates as described in Section 4.1. As the UPDATE messages are originating from an unknown location from the viewpoint of the Data Relay Server, it must also include an ECHO_REQUEST_SIGNED in the response in order to test for return routability.

Data Relay unregistration: This follows the procedure in Section 4, but the Data Relay Client should unregister using the particular transport address to be unregistered. All transport address pair registrations can be unregistered when no RELAYED_ADDRESS parameter is included.

PEER_PERMISSION parameter: This needs to be extended or an additional parameter is needed to declare the specific local candidate of the Data Relay Client. Alternatively, the use of the PEER_PERMISSION could be used as a wild card to open permissions for a specific peer to all of the candidates of the Data Relay Client.

Connectivity checks: The controlling host should be able to nominate multiple candidates (by repeating step 7 in Figure 5 in Section 4.6 using the additional candidate pairs).

Keepalives: These should be sent for all the nominated candidate pairs. Similarly, the Control/Data Relay Client should send keepalives from its local candidates to its Control/Data Relay Server transport addresses.

Appendix E. DNS Considerations

This section updates Appendix B of [RFC5770], which will be replaced with the mechanism described in this section.

[RFC5770] did not specify how an end host can look up another end host via DNS and initiate a UDP-based HIP base exchange with it, so this section makes an attempt to fill this gap.

[RFC8005] specifies how a HIP end host and its Rendezvous Server is registered to DNS. Essentially, the public key of the end host is stored as a HI record and its Rendezvous Server as an A or AAAA record. This way, the Rendezvous Server can act as an intermediary for the end host and forward packets to it based on the DNS configuration. The Control Relay Server offers similar functionality to the Rendezvous Server, with the difference being that the Control Relay Server forwards all control messages, not just the first I1 message.

Prior to this document, the A and AAAA records in the DNS refer either to the HIP end host itself or a Rendezvous Server [RFC8005], and control and data plane communication with the associated host has been assumed to occur directly over IPv4 or IPv6. However, this specification extends the records to be used for UDP-based communications.

Let us consider the case of a HIP Initiator with the default policy to employ UDP encapsulation and the extensions defined in this document. The Initiator looks up the Fully Qualified Domain Name (FQDN) of a Responder, and retrieves its HI, A, and AAAA records. Since the default policy is to use UDP encapsulation, the Initiator **MUST** send the I1 message over UDP to destination port 10500 (either over IPv4 in the case of an A record or over IPv6 in the case of an AAAA record). It **MAY** send an I1 message both with and without UDP encapsulation in parallel. In the case in which the Initiator receives R1 messages both with and without UDP encapsulation from the Responder, the Initiator **SHOULD** ignore the R1 messages without UDP encapsulation.

The UDP-encapsulated I1 packet could be received by four different types of hosts:

HIP Control Relay Server: In this case, the A/AAAA records refer to a Control Relay Server, which will forward the packet to the corresponding Control Relay Client based on the destination HIT in the I1 packet.

HIP Responder supporting UDP encapsulation: In this case, the A/AAAA records refer to the end host. Assuming the destination HIT belongs to the Responder, the Responder receives and processes the I1 packet according to the negotiated NAT traversal mechanism. The support for the protocol defined in this document, as opposed to the support defined in [RFC5770], is dynamically negotiated during the base exchange. The details are specified in Section 4.3.

HIP Rendezvous Server: This entity is not listening to UDP port 10500, so it will drop the I1 message.

HIP Responder not supporting UDP encapsulation: The targeted end host is not listening to UDP port 10500, so it will drop the I1 message.

The A/AAAA record **MUST NOT** be configured to refer to a Data Relay Server unless the host in question also supports Control Relay Server functionality.

It is also worth noting that SRV records are not employed in this specification. While they could be used for more flexible UDP port selection, they are not suitable for end-host discovery but rather would be more suitable for the discovery of HIP-specific infrastructure. Further extensions to this document may define SRV records for Control and Data Relay Server discovery within a DNS domain.

Acknowledgments

Thanks to Jonathan Rosenberg, Christer Holmberg, and the rest of the MMUSIC WG folks for the excellent work on ICE. The authors would also like to thank Andrei Gurtov, Simon Schuetz, Martin Stiemerling, Lars Eggert, Vivien Schmitt, and Abhinav Pathak for their contributions, and Tobias Heer, Teemu Koponen, Juhana Mattila, Jeffrey M. Ahrenholz, Kristian Slavov, Janne Lindqvist, Pekka Nikander, Lauri Silvennoinen, Jukka Ylitalo, Juha Heinanen, Joakim Koskela, Samu Varjonen, Dan Wing, Tom Henderson, Alex Elsayed, Jani Hautakorpi, Tero Kauppinen, and Timo Simanainen for their comments to [RFC5770] and this document. Thanks to Éric Vyncke, Alvaro Retana, Adam Roach, Ben Campbell, Eric Rescorla, Mirja Kühlewind, Spencer Dawkins, Derek Fawcus, Tianran Zhou, Amanda Barber, Colin Perkins, Roni Even, Alissa Cooper, Carl Wallace, Martin Duke, and Benjamin Kaduk for reviewing this document.

This work has been partially funded by the Cyber Trust Program by Digile/Tekes in Finland.

Contributors

Marcelo Bagnulo, Philip Matthews, and Hannes Tschofenig have contributed to [RFC5770]. This document leans heavily on the work in that RFC.

Authors' Addresses

Ari Keränen
Ericsson
Hirsalantie 11
FI-02420 Jorvas
Finland
Email: ari.keranen@ericsson.com

Jan Melén
Ericsson
Hirsalantie 11
FI-02420 Jorvas
Finland
Email: jan.melen@ericsson.com

Miika Komu (EDITOR)
Ericsson
Hirsalantie 11
FI-02420 Jorvas
Finland
Email: miika.komu@ericsson.com