Native NAT Traversal Mode for the Host Identity Protocol
draft-ietf-hip-native-nat-traversal-12

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 for all NAT traversal procedures.

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, especially 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.

   Two NAT traversal techniques for HIP are specified in [RFC5770]. One
   of them uses only UDP encapsulation, while the other uses also the
   Interactive Connectivity Establishment (ICE)
   [I-D.ietf-ice-rfc5245bis] protocol, which in turn uses Session
   Traversal Utilities for NAT (STUN) [RFC5389] and Traversal Using
   Relays around NAT (TURN) [RFC5766] protocols to achieve a reliable
   NAT traversal solution.

   The benefit of using ICE and STUN/TURN is that one can re-use the NAT
   traversal infrastructure already available in the Internet, such as
   STUN and TURN servers. Also, some middleboxes may be STUN-aware and
   could be able to do something "smart" when they see STUN being used
   for NAT traversal. However, implementing a full ICE/STUN/TURN
   protocol stack results in a considerable amount of effort and code
   which could be avoided by re-using and extending HIP messages and
   state machines for the same purpose. Thus, this document specifies
   an alternative NAT traversal mode that uses HIP messages instead of
   STUN for the connectivity checks keepalives, and data relaying. This
   document also specifies how mobility management works in the context
   of NAT traversal, which was missing from [RFC5770].

2. Terminology

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

This document borrows terminology from [RFC5770], [RFC7401],
[I-D.ietf-hip-rfc5206-bis], [RFC4423], [I-D.ietf-ice-rfc5245bis], and
[RFC5389].   The following terms recur in the text:

HIP relay server:
   A host that forwards any kind of HIP control packets between the
   Initiator and the Responder.

HIP data relay:
   A host that forwards HIP data packets, such as Encapsulating
   Security Payload (ESP) [RFC7402], between two hosts.

Registered host:
   A host that has registered for a relaying service with a HIP data
   relay.

Locator:
   As defined in [I-D.ietf-hip-rfc5206-bis]: "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.

ICE offer:
   The Initiator's LOCATOR_SET parameter in a HIP I2 control packet.
   Corresponds to the ICE offer parameter, but is HIP specific.

ICE answer:
   The Responder's LOCATOR_SET parameter in a HIP R2 control packet.
   Corresponds to the ICE answer parameter, but is HIP specific.

HIP connectivity checks:
   In order to obtain a non-relayed communication path, two
   communicating HIP hosts try to "punch holes" through their NAT

boxes using this mechanism. Similar to the ICE connectivity
checks, but implemented using HIP return routability checks.

Controlling host :
   The controlling host nominates the candidate pair to be used with
   the controlled host.

Controlled host :
   The controlled host waits for the controlling to nominate an
   address candidate pair.

Checklist:
   A list of address candidate pairs that need to be tested for
   connectivity.

Transport address:
   Transport layer port and the corresponding IPv4/v6 address.

Candidate:
   A transport address that is a potential point of contact for
   receiving data.

Host candidate:
   A candidate obtained by binding to a specific port from an IP
   address on the host.

Server reflexive candidate:
   A translated transport address of a host as observed by a HIP
   relay server or a STUN/TURN server.

Peer reflexive candidate:
   A translated transport address of a host as observed by its peer.

Relayed candidate:
   A transport address that exists on a HIP data relay. Packets that
   arrive at this address are relayed towards the registered host.

[3](). Overview of Operation

```
                                +-------+
                                | HIP   |
           +--------+           | Relay |           +--------+
           | Data   |           +-------+           | Data   |
           | Relay  |          /         \          | Relay  |
           +--------+         /           \          +--------+
                             /             \
                            /               \
                           /                 \
                          /  <- Signaling ->  \
                         /                     \
                  +-------+                 +-------+
                  | NAT   |                 | NAT   |
                  +-------+                 +-------+
                   /                             \
                  /                               \
           +-------+                         +-------+
           | Init- |                         | Resp- |
           | iator |                         | onder |
           +-------+                         +-------+
```
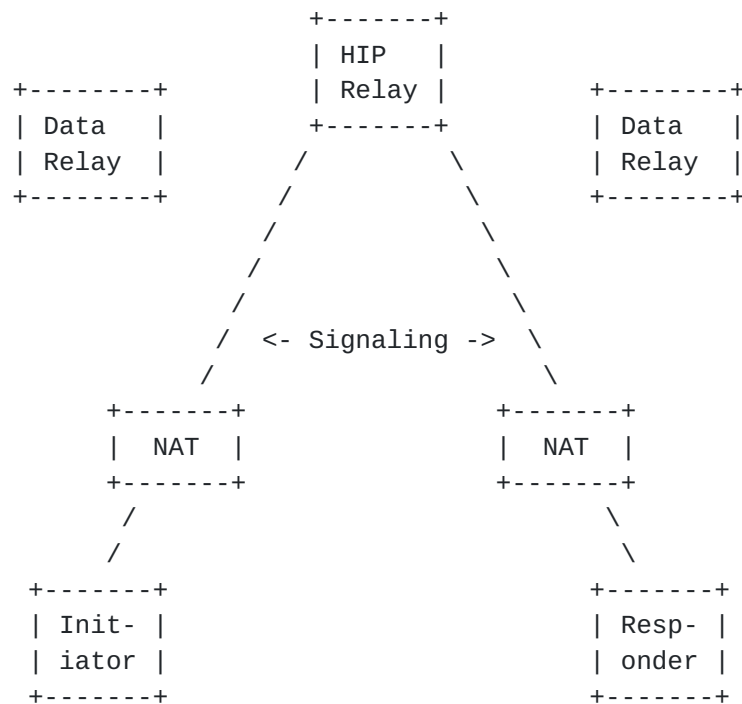
                 Figure 1: Example Network Configuration

   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,
   the Responder must be registered with a HIP relay server reachable on
   the public Internet, and we assume, as a starting point, that the
   Initiator knows both the Responder's Host Identity Tag (HIT) and the
   address of one of its relay servers (how the Initiator learns of the
   Responder's relay server is outside of the scope of this document,
   but may be through DNS or another name service).

   The first steps are for both the Initiator and Responder to register
   with a relay server (need not be the same one) and gather a set of
   address candidates. The hosts may use HIP relay servers (or even
   STUN or TURN 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 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, which follows closely the ICE
   [I-D.ietf-ice-rfc5245bis] protocol.

   Once the base exchange is completed, two HIP hosts have established a
   working communication session (for signaling) via a HIP relay server,
   but the hosts still have to find a better path, preferably without a
   HIP data relay, 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 HIP data relay server.
At this point, also the HIP signaling can be sent over the same
address/port pair, and is demultiplexed 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 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, HIP data relay,
is specified. Also, it should be noted that HIP version 2 [RFC7401]
(instead of [RFC5201] used in [RFC5770]) is expected to be used with
this NAT traversal mode.

4.1. Relay Registration

In order for two hosts to communicate over NATted environments, they
need a reliable way to exchange information. HIP relay servers as
defined in [RFC5770] support relaying of HIP control plane traffic
over UDP in NATted environments. A HIP relay server forwards HIP
control packets between the Initiator and the Responder.

To guarantee also 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 [TBD by IANA: 3]). This
service may be coupled with the HIP 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
[I-D.ietf-hip-rfc5203-bis]). 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 relay 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" [I-D.ietf-hip-rfc5203-bis].

A HIP relay server MUST silently drop packets to a HIP relay client
that has not previously registered with the HIP relay. The
registration process follows the generic registration extensions
defined in [I-D.ietf-hip-rfc5203-bis]. 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
relay 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.

```
  HIP                                                            HIP
  Relay                                                   [Data] Relay
  Client                                                       Server
  |   1. UDP(I1)                                                   |
  +-------------------------------------------------------------->|
  |                                                               |
  |   2. UDP(R1(REG_INFO(RELAY_UDP_HIP,[RELAY_UDP_ESP])))         |
  |<--------------------------------------------------------------+
  |                                                               |
  |   3. UDP(I2(REG_REQ(RELAY_UDP_HIP),[RELAY_UDP_ESP])))        |
  +-------------------------------------------------------------->|
  |                                                               |
  |   4. UDP(R2(REG_RES(RELAY_UDP_HIP,[RELAY_UDP_ESP]), REG_FROM, |
  |            [RELAYED_ADDRESS]))                                 |
  |<--------------------------------------------------------------+
  |                                                               |
```

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. It is
RECOMMENDED that the Initiator select a random 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 HIP relay server listen to
incoming connections at UDP port 10500. If some other port number is
used, it needs to be known by potential Initiators.

In step 2, the HIP 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 supports also relaying of ESP
traffic over UDP, it includes also Registration type value
RELAY_UDP_ESP.

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

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

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

The registered host MUST maintain an active HIP association with the
data relay 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 registered host refreshing the
registration, the data relay MUST stop relaying packets for that host
and close the corresponding UDP port (unless other registered hosts
are still using it).

The data relay MAY use the same relayed address and port for multiple
registered hosts, but since this can cause problems with stateful
firewalls (see Section 6.5) it is NOT RECOMMENDED.

4.2. Transport Address Candidate Gathering

A host needs to gather a set of address candidates before contacting
a non-relay host. 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 HIP relay server from the
REG_FROM parameter.

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 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 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 relay registration as a server reflexive
candidate. In this case, no further candidate gathering is needed.

If a host has more than one network interface, additional server
reflexive candidates can be discovered by sending registration
requests with Registration Type CANDIDATE_DISCOVERY (value [TBD by
IANA: 4]) from each of the interfaces to a HIP relay server. When a
HIP relay server receives a registration request with
CANDIDATE_DISCOVERY type, it MUST add a REG_FROM parameter,
containing the same information as if this was a relay registration,
to the response. This request type SHOULD NOT create any state at
the HIP relay server.

Gathering of candidates MAY also be performed as specified in
Section 4.2 of [RFC5770] if STUN servers are available, or if the
host has just a single interface and no TURN or HIP data relay
servers are available.

4.3. NAT Traversal Mode Negotiation

This section describes the usage of a new non-critical parameter
type. The presence of the 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 does not support or is not willing to use these
extensions.

With registration with a HIP relay, it is usually sufficient to use
the UDP-ENCAPSULATION mode of NAT traversal since the relay is
assumed to be in public address space. Thus, the relay 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 HIP relay could
be located between the hosts, but omitted here for simplicity.

```
  Initiator                                                  Responder
   | 1. UDP(I1)                                                    |
   +------------------------------------------------------------->|
   |                                                              |
   | 2. UDP(R1(.., NAT_TRAVERSAL_MODE(ICE-HIP-UDP), ..))          |
   |<------------------------------------------------------------+
   |                                                              |
   | 3. UDP(I2(.., NAT_TRAVERSAL_MODE(ICE-HIP-UDP), LOC_SET, ..)) |
   +------------------------------------------------------------->|
   |                                                              |
   | 4. UDP(R2(.., LOC_SET, ..))                                  |
   |<------------------------------------------------------------+
   |                                                              |
```
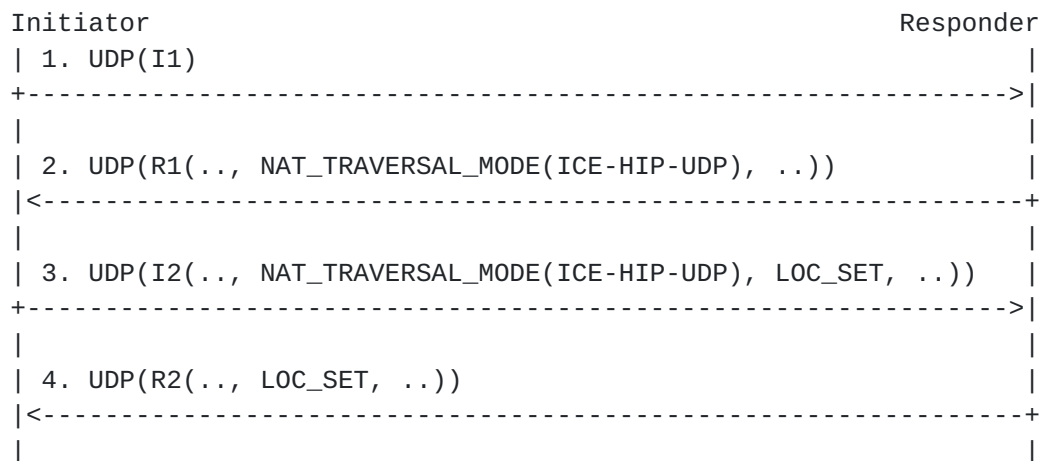
Figure 3: Negotiation of NAT Traversal Mode

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 [TBD by IANA: 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 LOC_SET). The locators in I2 are
the "ICE offer".

In step 4, the Responder concludes the base exchange with an R2
packet. If the Initiator chose ICE NAT traversal mode, the Responder
includes a LOCATOR_SET parameter in the R2 packet. 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 [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 out of the two offered values.

The minimum Ta value SHOULD be configurable, and if no value is configured, a value of 500 ms MUST be used. Guidelines for selecting a Ta value are given in Appendix A. Hosts SHOULD NOT use values smaller than 20 ms for the minimum Ta, since such values may not work well with some NATs, as explained in [I-D.ietf-ice-rfc5245bis]. 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 500 ms MUST be used as that host's minimum value.

4.5. Base Exchange via HIP Relay Server

This section describes how the Initiator and Responder perform a base exchange through a HIP relay server. Connectivity pacing (denoted as TA_P here) was described in Section 4.4 and is neither repeated here. Similarly, the NAT traversal mode negotiation process (denoted as NAT_TM in the example) was described in Section 4.3 and is neither repeated here. If a relay 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 IPv4 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 HIP relay server. We assume that the Responder (i.e. a HIP relay client) has already registered to the HIP relay server. The Initiator may have also registered to another (or the same relay server), but the base exchange will traverse always through the relay of the Responder.

```
   Initiator                       HIP relay             Responder
    | 1. UDP(I1)                     |                         |
    +------------------------------->| 2. UDP(I1(RELAY_FROM))  |
    |                                 +------------------------------->|
    |                                |                         |
    |                                | 3. UDP(R1(RELAY_TO, NAT_TM,  |
    |                                |         TA_P))          |
    | 4. UDP(R1(RELAY_TO, NAT_TM,    |<-------------------------------+
    |        TA_P))                  |                         |
    |<-------------------------------+                         |
    |                                |                         |
    | 5. UDP(I2(LOC_SET, NAT_TM,     |                         |
    |        TA_P))                  |                         |
    +------------------------------->| 6. UDP(I2(LOC_SET, RELAY_FROM, |
    |                                |           NAT_TM, TA_P))  |
    |                                 +------------------------------->|
    |                                |                         |
    |                                | 7. UDP(R2(LOC_SET, RELAY_TO))  |
    | 8. UDP(R2(LOC_SET, RELAY_TO))  |<-------------------------------+
    |<-------------------------------+                         |
    |                                |                         |
```
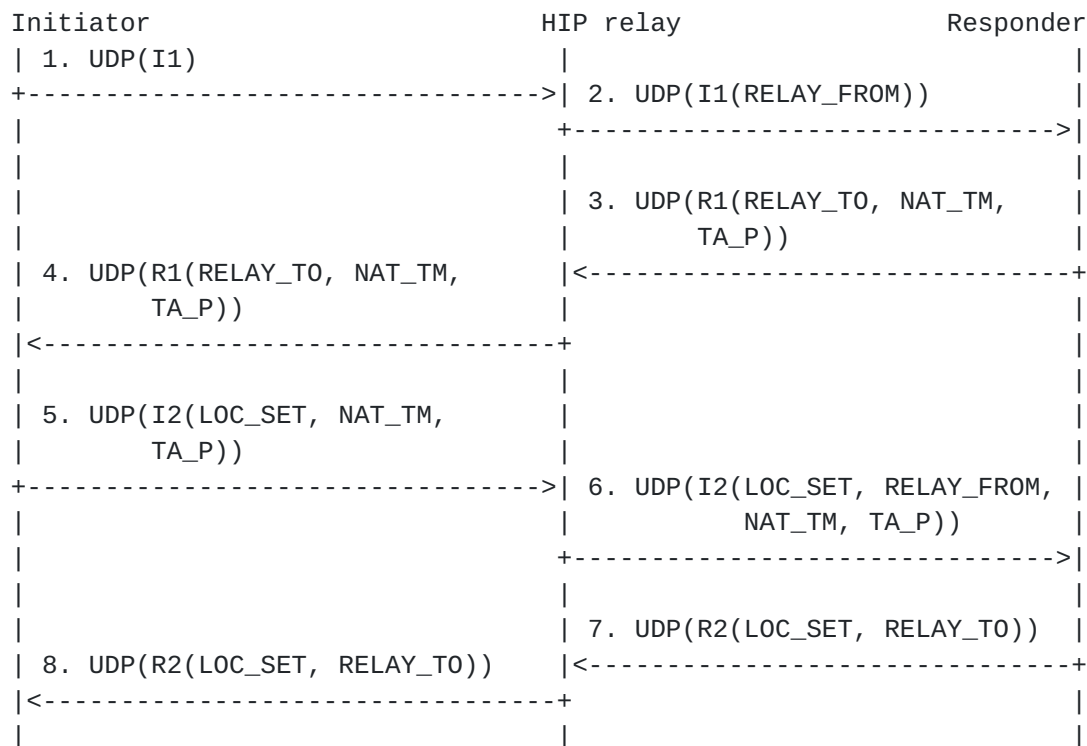
               Figure 4: Base Exchange via a HIP Relay Server

   In step 1 of Figure 4, the Initiator sends an I1 packet over UDP via
   the 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 HIP relay over UDP.

   In step 2, the HIP relay server receives the I1 packet. If the
   destination HIT belongs to a registered Responder, the relay
   processes the packet. Otherwise, the relay MUST drop the packet
   silently. The relay appends a RELAY_FROM parameter to the I1 packet,
   which contains the transport source address and port of the I1 as
   observed by the relay. The relay protects the I1 packet with
   RELAY_HMAC as described in [I-D.ietf-hip-rfc5204-bis], except that
   the parameter type is different (see Section 5.8). The relay changes
   the source and destination ports and IP addresses of the packet to
   match the values the Responder used when registering to the relay,
   i.e., the reverse of the R2 used in the registration. The relay 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

[I-D.ietf-hip-rfc5204-bis] 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 relay receives the R1 packet. The relay drops the
packet silently if the source HIT belongs to an unregistered host.
The relay MAY verify the signature of the R1 packet and drop it if
the signature is invalid. Otherwise, the relay rewrites the source
address and port, and changes the destination address and port to
match RELAY_TO information. Finally, the relay recalculates
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 that lists all the HIP
candidates (ICE 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.

In step 6, the relay receives the I2 packet. The relay appends a
RELAY_FROM and a RELAY_HMAC to the I2 packet similarly as 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]. 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 that lists all the HIP candidates (ICE answer)
of the Responder. The RELAY_TO parameter is protected by the HMAC.

In step 8, the relay processes the R2 as described in step 4. The
relay 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 HIP relay servers
(including the one that is being used for the initial signaling) in
the LOCATOR_SET parameter in I2 and R2 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" and they MUST NOT be used as HIP candidates. If the HIP
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.

4.6. Connectivity Checks

When the Initiator and Responder complete the base exchange through
the HIP relay, both of them employ the IP address of the relay as the
destination address for the packets. This address MUST NOT be used
as a destination for ESP traffic unless the HIP relay supports also
ESP data relaying. 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.

The connectivity checks follow the ICE methodology [MMUSIC-ICE], but
UDP encapsulated HIP control messages are used instead of ICE
messages. Only normal connectivity checks can be used because
aggressive connectivity checks are deprecated. The Initiator MUST
take the role of controlling host and the Responder acts as the
controlled host. The protocol follows standard HIP UPDATE sending
and processing rules as defined in section 6.11 and 6.12 in
[RFC7401], but some new parameters are introduced:
CANDIDATE_PRIORITY, MAPPED_ADDR and NOMINATE.

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 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.

```
 Initiator   NAT1                                NAT2       Responder
    |              | 1. UDP(UPDATE(SEQ, CAND_PRIO,      |           |
    |              |          ECHO_REQ_SIGN))           |           |
    |            X<------------------------------------+-----------+
    |              |                                    |           |
    | 2. UDP(UPDATE(SEQ, ECHO_REQ_SIGN, CAND_PRIO))     |           |
    +------------+-------------------------------------+----------->|
    |              |                                    |           |
    | 3. UDP(UPDATE(SEQ, ACK, ECHO_REQ_SIGN, ECHO_RESP_SIGN,        |
    |              | MAPPED_ADDR))                      |           |
    |<-----------+-------------------------------------+-----------+
    |              |                                    |           |
    | 4. UDP(UPDATE(ACK, ECHO_RESP_SIGN, MAPPED_ADDR)) |           |
    +------------+-------------------------------------+----------->+
    |              |                                    |           |
    | 5. Other connectivity checks using UPDATE over UDP            |
    <------------+-------------------------------------+----------->
    |              |                                    |           |
    | 6. UDP(UPDATE(SEQ, ECHO_REQ_SIGN, CAND_PRIO, NOMINATE))       |
    +------------+-------------------------------------+----------->|
    |              |                                    |           |
    | 7. UDP(UPDATE(SEQ, ACK, ECHO_REQ_SIGN, ECHO_RESP_SIGN,        |
    |          NOMINATE))                               |           |
    |<-----------+-------------------------------------+-----------+
    |              |                                    |           |
    | 8. UDP(UPDATE(ACK, ECHO_RESP_SIGN))              |           |
    +------------+-------------------------------------+----------->+
    |              |                                    |           |
    | 9. ESP data traffic over UDP                     |           |
    +<-----------+-------------------------------------+----------->+
    |              |                                    |           |
```

                    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 the Responder did and the message traverses
   successfully the NAT boxes. The message includes the same parameters
   as 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_REQUEST_SIGNED (denoted ECHO_REQUEST_SIGN) parameter. The
Responder includes also a MAPPED_ADDRESS parameter that contains the
transport address of the Initiator as observed by the Responder (i.e.
peer reflexive candidate). The Initiator should acknowledge the
message from the Responder, so the Responder also includes its own
SEQ in the message and its own echo request for additional security.

In step 4, the Initiator receives the message from the Responder and
builds a corresponding response that concludes connectivity checks.
Since the previous message from the Responder included a new SEQ and
ECHO_REQUEST_SIGN parameters, the Initiator includes the
corresponding ACK and ECHO_RESPONSE_SIGN parameters. Before sending,
it also includes a MAPPED_ADDR parameter describing the peer
reflexive candidate.

In step 5, the Initiator and Responder test the remaining address
candidates (if any).

In step 6, 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_SIGN, CANDIDATE_PRIORITY and the
NOMINATE parameter.

In step 7, the Responder receives the message with 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_SIGN parameters acknowledge the SEQ and
ECHO_REQUEST_SIGN parameters from previous message from the
Initiator. The Responder includes SEQ and ECHO_REQUEST_SIGN
parameters in order to receive an acknowledgment from the Responder.

In step 8, the Initiator completes the candidate nomination process
by confirming the message reception to the Responder. In the
confirmation message, the ACK and ECHO_RESPONSE_SIGN parameters
correspond to the SEQ and ECHO_REQUEST_SIGN parameters in the message
sent by the Responder in the previous step.

In step 9, 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, the host MUST activate the

address before connectivity checks by sending an UPDATE message
containing PEER_PERMISSION parameter as described in Section 4.12.1.
Otherwise, the relay drops ESP packets using the relayed address.

4.6.2. Rules for Connectivity Checks

All of the connectivity check packets MUST be protected with HMACs
and signatures (even though the illustrations omitted them for
simplicity). To provide strong replay protection, for each pair of
address candidates, both the Initiator and Responder MUST send a send
a nonce to each other for signing using the ECHO_REQUEST_SIGNED
parameter (that then has to be echoed back by the recipient).
Similarly, the SEQ parameter enforces the the recipient to
acknowledge a received message. Effectively these two mechanisms
combined result in a secure three way packet exchange that tests both
sides for return routability.

[RFC7401] states that UPDATE packets have to include either a SEQ or
ACK parameter (or both). According to the RFC, 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 lost or arrive out of order. From the viewpoint of the
recipient, this is not a problem since the the recipient will just
"blindly" acknowledge the SEQ. However, the sender needs to be
prepared for lost sequence identifiers and ACKs 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 checks SHOULD
be sent with the same sequence identifier in the SEQ parameter. Some
of tested address candidates will never produce a working address
pair, and thus may cause retransmissions. Upon successful nomination
an address pair, a host MAY immediately stop sending such
retransmissions.

The packet flow illustrations are missing a scenario where both the
Initiator and Responder send simultaneously connectivity checks to
each other using the same address candidates, and the NATs at both
sides let the packets pass. From the viewpoint of NAT penetration,
this results in a bit more unnecessary packet exchanges, but both
ends SHOULD nevertheless complete the three way connectivity check
process they initiated.

The connectivity check messages MUST be paced by the 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 500 ms MUST be used.

As defined in [RFC5770], both hosts MUST form a priority ordered checklist and start 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 MUST be calculated as follows:

    RTO = MAX (500ms, 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 [I-D.ietf-ice-rfc5245bis] if there is only one checklist.

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.

If the connectivity checks 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 Alternatives

4.7.1. Minimal NAT Traversal Support

If the Responder has a fixed and publicly reachable IPv4 address and does not employ a HIP relay, the explicit NAT traversal mode negotiation MAY be omitted, and 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 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 implicitly. This way of
operation is still subject to NAT timeouts, and the hosts MUST employ
NAT keepalives as defined in section Section 4.10.

4.7.2. Base Exchange without Connectivity Checks

It is possible to run a base exchange without any connectivity checks
as defined in section 4.8 in [RFC5770]. The procedure is applicable
also 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 set-up 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 HIP relay server, it MUST also
include a LOCATOR_SET parameter in R1 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 1). If there is no such locator in R1, the source
address of R1 is used 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. However, the
Responder SHOULD NOT send any ESP to the Initiator's address before
it has received data from the Initiator, as specified in Sections
4.4.3. and 6.9 of [RFC7401] and in Sections 3.2.9 and 5.4 of
[I-D.ietf-hip-rfc5206-bis].

Since an I2 packet with UDP-ENCAPSULATION NAT traversal mode selected
MUST NOT be sent via a relay, 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 HIP 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 section 4.9 in [RFC5770].
The procedure is applicable also 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
any relays, 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
retransmits of the original R1 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
[I-D.ietf-hip-rfc5204-bis].

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 of sending control packets after the base
exchange described in section 5.10 in [RFC5770] apply also here, so
they are repeated here for completeness.

After the base exchange, the 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 the
HIP relay server. When a host does not get acknowledgments, e.g., to
an UPDATE or CLOSE packet after a timeout based on local policies,
the host SHOULD resend the packet through the relay, if it was listed
in the LOCATOR_SET parameter in the base exchange.

If control packets are sent through a HIP relay server, the host
registered with the relay MUST utilize the RELAY_TO parameter as in
the base exchange. The HIP relay server SHOULD forward HIP packets
to the registered hosts and forward packets from a registered host to
the address in the RELAY_TO parameter. The relay MUST add a
RELAY_FROM parameter to the control packets it relays to the
registered hosts.

If the HIP relay server is not willing or able to relay a HIP packet,
it MAY notify the sender of the packet with 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 [I-D.ietf-hip-rfc5206-bis] define
handover procedures without NATs. In this section, we define how two
hosts interact 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.

We assume 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.

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 HIP relay belongs to the peer
host, i.e., the peer host is a relay client for the HIP relay. It is
worth noting that the figure corresponds to figure 3 in Figure 6, 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 in the figure in detail.

```
Mobile Host                        HIP relay                 Peer Host
| 1. UDP(UPDATE(ESP_INFO,          |                           |
|         LOC_SET, SEQ))           |                           |
+--------------------------------->| 2. UDP(UPDATE(ESP_INFO,   |
|                                  |         LOC_SET, SEQ,     |
|                                  |         RELAY_FROM))      |
|                                  +-------------------------------->|
|                                  |                           |
|                                  | 3. UDP(UPDATE(ESP_INFO, SEQ,   |
|                                  |         ACK, ECHO_REQ_SIGN))   |
| 4. UDP(UPDATE(ESP_INFO, SEQ,     |<--------------------------------+
|         ACK, ECHO_REQ_SIGN,      |                           |
|         RELAY_TO))               |                           |
|<--------------------------------+                            |
|                                  |                           |
| 5. UDP(UPDATE(ACK,               |                           |
|         ECHO_RESP_SIGN))         |                           |
+--------------------------------->| 6. UDP(UPDATE(ACK,        |
|                                  |         ECHO_RESP_SIGN,   |
|                                  |         RELAY_FROM))      |
|                                  +-------------------------------->|
|                                  |                           |
|                7. connectivity checks over UDP               |
+<------------------------------------------------------------->+
|                                  |                           |
|                  8. ESP data over UDP                        |
+<------------------------------------------------------------->+
|                                  |                           |
```
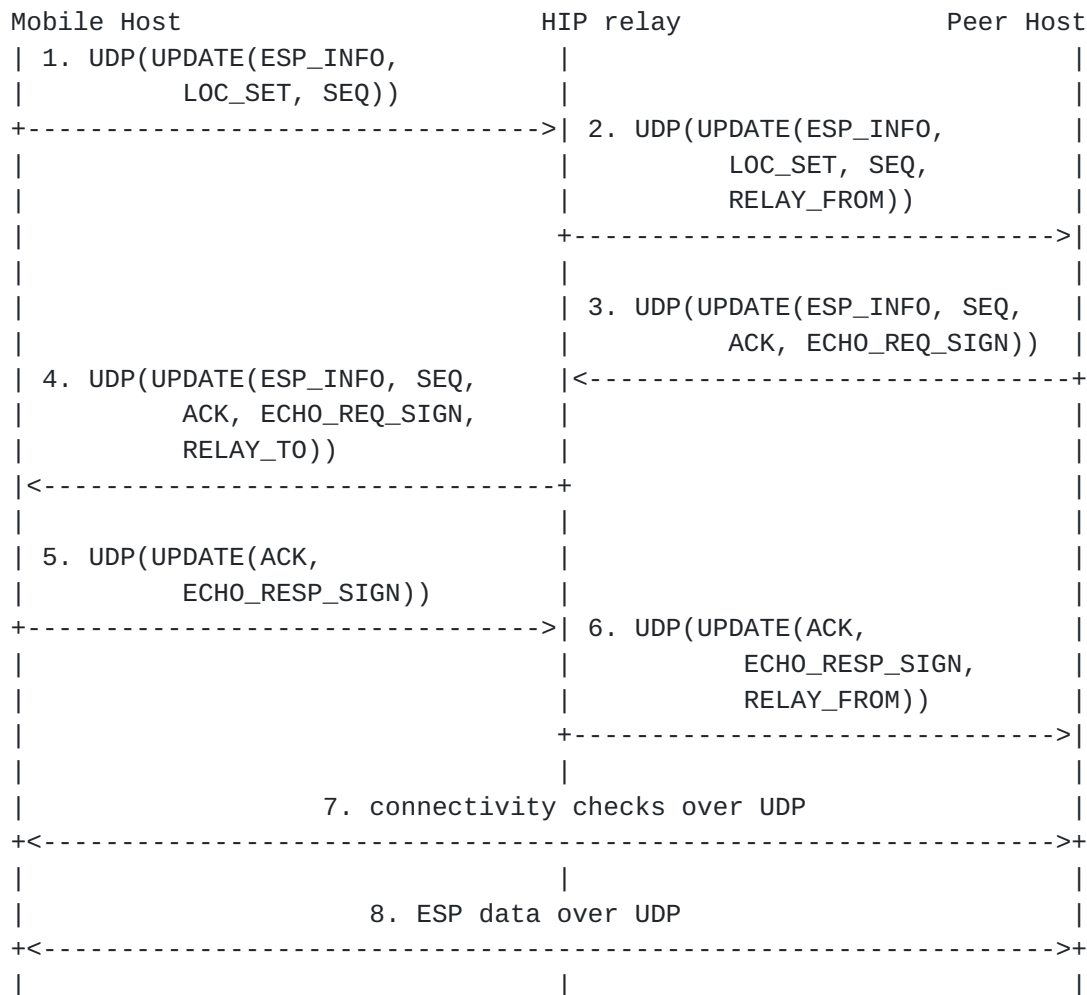
                    Figure 6: HIP UPDATE procedure

In step 1, the mobile host has changed location and sends a location
update to its peer through the HIP relay of the peer. It sends an
UPDATE packet with 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 HIP relay. 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. The packet contains also a SEQ number to be
acknowledged by the peer. As specified in
[I-D.ietf-hip-rfc5206-bis], the packet may also include a HOST_ID
(for middlebox inspection) and DIFFIE_HELLMAN parameter for rekeying.

In step 2, 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 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 mobile host and to the IP address of the HIP relay. 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_SIGN parameters to
be acknowledged by the mobile host. The message acknowledges the SEQ
parameter of the earlier message with an ACK parameter.

In step 4, the HIP relay receives the message, rewrites the
destination IP address, appends an RELAY_TO parameter and forwards
the modified message to the mobile host.

In step 5, the mobile host receives the UPDATE packet from the peer
and processes it. It concludes the information exchange by
acknowledging the received SEQ parameter with an ACK parameter and
the ECHO_REQUEST_SIGN parameter with ECHO_RESPONSE_SIGN parameter.
The mobile host delivers 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, 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 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 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.

If either host has registered a relayed address candidate from a data
relay, the host MUST reactivate the address before connectivity
checks by sending an UPDATE message containing PEER_PERMISSION
parameter as described in Section 4.12.1. Otherwise, the relay drops
ESP packets using the relayed address.

4.10. NAT Keepalives

To prevent NAT states from expiring, communicating hosts send
periodic keepalives to other hosts that they have established a host
associating with. If a registered host has not sent any data or
control messages to its HIP or data relay for 15 seconds, it MUST
send a HIP NOTIFY packet to the relay. Likewise, if a host has not
sent any data to another host it has established a host association
in the ICE-HIP_UDP mode, it MUST send either a HIP NOTIFY packet or
an ICMPv6 echo request inside the related ESP tunnel. HIP relay
servers MAY refrain from sending keepalives if it's known that they
are not behind a middlebox that requires keepalives. If the base
exchange or mobility handover procedure occurs during an extremely
slow path, a host MAY also send HIP notify packets every 15 seconds
to keep to path active.

4.11. Closing Procedure

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

The same procedure for closing HIP associations applies also 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 HIP relay of the recipient if one exists. The
host receiving the CLOSE message directly without a relay SHOULD
respond directly. The the CLOSE message came via a relay, it SHOULD
respond using the same relay.

4.12. Relaying Considerations

4.12.1. Forwarding Rules and Permissions

The HIP data relay uses a similar permission model as a TURN server:
before the data relay forwards any ESP data packets from a peer to a
registered host (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) is selected to be used for data, the registered host
MUST send an UPDATE message to the data relay containing a
PEER_PERMISSION parameter (see Section 5.13) with the address of the
peer, and the outbound and inbound SPI values the registered host is
using with this particular peer. To avoid packet dropping of ESP
packets, the registered host 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 must acknowledge, so that the registered host can resend
the message with PEER_PERMISSION parameter if it gets lost.

When a data relay 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 relay checks if there is
a permission with matching information (address, protocol, port 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 registered host SHOULD refresh
permissions 1 minute before the expiration when the permission is
still needed.

The relayed address MUST be activated with the PEER_PERMISSION
parameter both after the base exchange and after a handover. Unless
activated, the data relay MUST drop all ESP packets.

4.12.2. Relaying UDP Encapsulated Control and Data Packets

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

data relay MUST send this packet to the peer's address from the
relayed address.

If the data relay 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 MUST forward the packet to the
registered host that created the permission. The data relay MUST
also implement the similar checks for the reverse direction (i.e.
ESP packets from the registered host to the peer). Packets without a
permission MUST be dropped silently.

4.12.3. Handling Conflicting SPI Values

The inbound SPI values of the registered clients should be unique so
that a data relay can properly demultiplex incoming packets from peer
hosts to the correct registered clients. Vice versa, the inbound
SPIs of the peer hosts should be unique for the same reason. These
two cases are discussed in this section separately.

In first case, the SPI collision problem occurs when two Initiators
run a base exchange to the same Responder (i.e. registered host), and
both the Initiators claim the same inbound SPI. Upon receiving an I2
with a colliding SPI, the Responder MUST not include the relayed
address in the R2 message because the data relay would not be able
demultiplex the related ESP packet to the correct Initiator. Since
the SPI space is 32 bits and the SPI values should be random, the
probability for a conflicting SPI value is fairly small. However, a
registered host with many peers MAY proactively decrease the odds of
a conflict by registering to multiple data relays. The described
collision scenario can be avoided if the Responder delivers a new
relayed address candidate upon SPI collisions. Each relayed address
has a separate UDP port reserved to it, so the relay can demultiplex
properly conflicting SPIs of the Initiators based on the SPI and port
number towards the correct Responder.

In the second case, the SPI collision problems occurs if two hosts
have registered to the same data relay and a third host initiates
base exchange with both of them. In this case, the data relay has
allocated separate UDP ports for the two registered hosts acting now
as Responders. When the Responders send identical SPI values in
their I2 messages via the relay, it can properly demultiplex it to
the correct Responder because the UDP ports are different.

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 most of the parameters are shown for
completeness sake even though they are specified already in
[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 [RFC5770].

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |          Source Port          |       Destination Port        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |            Length             |            Checksum           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                       32 bits of zeroes                       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    ~                    HIP Header and Parameters                  ~
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
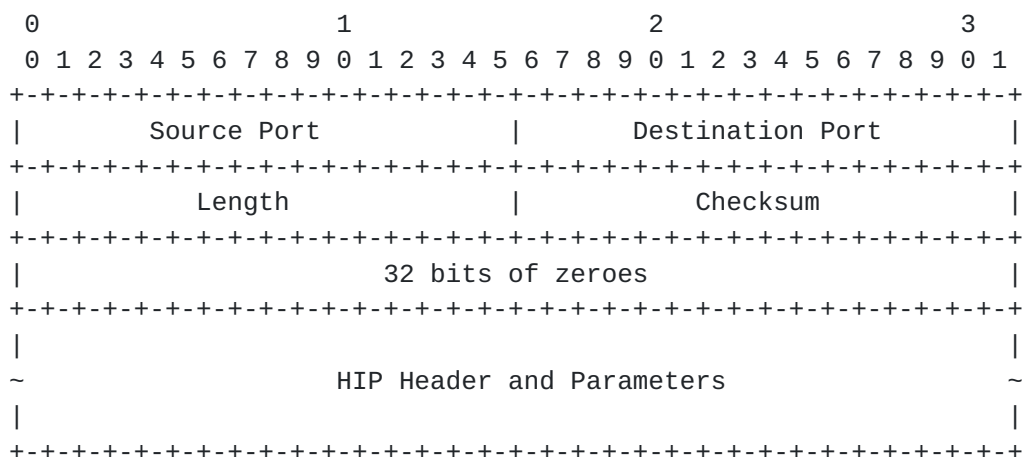
Figure 7: Format of UDP-Encapsulated HIP Control Packets

HIP control packets are encapsulated in UDP packets as defined in
Section 2.2 of [RFC3948], "IKE Header Format for Port 4500", except 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 unaware of HIP cannot recompute the HIP checksum after changing
IP addresses.

A HIP relay server or a Responder without a relay 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.

5.2. Connectivity Checks

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

5.3. Keepalives

The keepalives are either HIP NOTIFY packets as specified in
[RFC7401] or ICMPv6 packets inside the ESP tunnel.

5.4. NAT Traversal Mode Parameter

The format of NAT traversal mode parameter is borrowed from
[RFC5770]. 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. This specification defines traversal mode
identifier for ICE-HIP-UDP. The identifier RESERVED is reserved for
future use. Future specifications may define more traversal modes.

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |             Type              |             Length            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |            Reserved           |           Mode ID #1          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           Mode ID #2          |           Mode ID #3          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           Mode ID #n          |            Padding            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

   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
     ICE-HIP-UDP          3
```

Figure 8: Format of the NAT_TRAVERSAL_MODE Parameter

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.

5.5. Connectivity Check Transaction Pacing Parameter

The TRANSACTION_PACING is a new parameter, and it shown in Figure 9
contains only the connectivity check pacing value, expressed in
milliseconds, as a 32-bit unsigned integer.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type              |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Min Ta                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Type      610
Length    4
Min Ta    the minimum connectivity check transaction pacing
          value the host would use
```
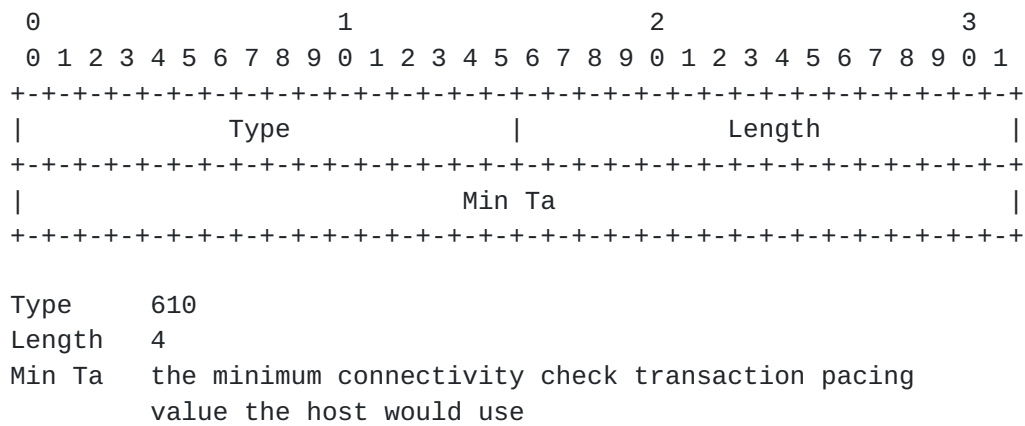
Figure 9: Format of the TRANSACTION_PACING Parameter

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. REG_FROM is the only parameter covered with the signature.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type              |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Port              |   Protocol    |    Reserved   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                           Address                             |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+


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
```

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

REG_FROM contains the transport address and protocol from which the
HIP relay server sees the registration coming. RELAY_FROM contains
the address from which the relayed packet was received by the 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 specified
in [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.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Type             |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Traffic Type  | Locator Type  | Locator Length|  Reserved   |P|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Locator Lifetime                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Locator                            |
|                                                               |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                                                               .
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Traffic Type  |  Loc Type = 2 | Locator Length|  Reserved   |P|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Locator Lifetime                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Transport Port           | Transp. Proto|     Kind      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Priority                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              SPI                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Address                            |
|                                                               |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 11: LOC_SET Parameter

The individual fields in the LOCATOR_SET parameter are described in
Table 1.

Field	Value(s)	Purpose
Type	193	Parameter type
Length	Variable	Length in octets, excluding Type and Length fields and padding
Traffic Type	0-2	Is the locator for HIP signaling (1), for 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
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
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 or 3 for relayed address
Priority	Variable	Locator's priority as described in [I-D.ietf-ice-rfc5245bis]
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 1: Fields of the LOCATOR_SET Parameter

5.8. RELAY_HMAC Parameter

As specified in [RFC5770], the RELAY_HMAC parameter value has the TLV
type 65520. It has the same semantics as RVS_HMAC
[I-D.ietf-hip-rfc5204-bis].

5.9. Registration Types

The REG_INFO, REG_REQ, REG_RESP, and REG_FAILED parameters contain
Registration Type [I-D.ietf-hip-rfc5203-bis] values for HIP relay
server registration. The value for RELAY_UDP_HIP is 2 as specified
in [RFC5770].

5.10. Notify Packet Types

   A HIP relay server and end-hosts can use NOTIFY packets to signal
   different error conditions. The NOTIFY packet types are the same as
   in [RFC5770].

   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   60

      If a HIP relay server does not forward a base exchange packet due
      to missing NAT traversal mode parameter, or the Initiator selects
      a NAT traversal mode that the Responder did not expect, the relay
      or the Responder may send back a NOTIFY error packet with this
      type.

   CONNECTIVITY_CHECKS_FAILED   61

      Used by the end-hosts to signal that NAT traversal connectivity
      checks failed and did not produce a working path.

   MESSAGE_NOT_RELAYED   62

      Used by a HIP relay server to signal that is was not able or
      willing to relay a HIP packet.

5.11. ESP Data Packets

   The format for ESP data packets is identical to [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, and 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 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, and, thus, only protocol 17 is allowed.
However, future documents may specify support for other protocols.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type              |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Port              |    Protocol   |   Reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                            Address                            |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+


Type      [TBD by IANA;
           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
```

Figure 12: Format of the RELAYED_ADDRESS and MAPPED_ADDRESS
                        Parameters

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. 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.

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |              Type             |             Length            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |             Port             |   Protocol    |   Reserved    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   |                                                               |
   |                           Address                             |
   |                                                               |
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                            OSPI                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                            ISPI                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   |                            ...                                |
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
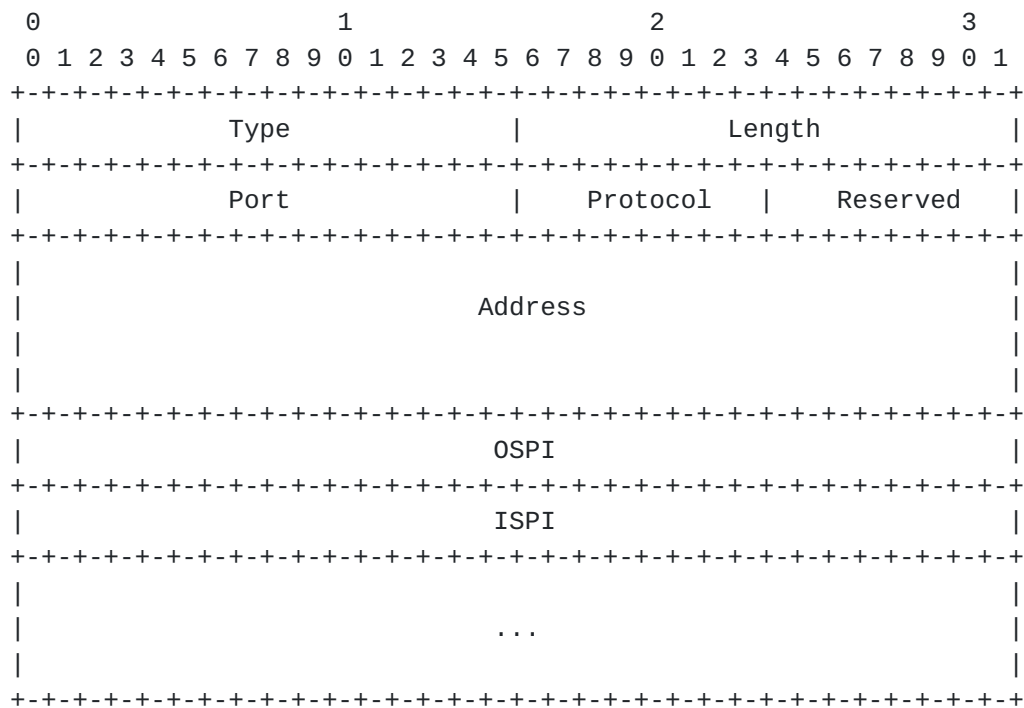
```
Type       [TBD by IANA; 4680]
Length     length in octets, excluding Type and Length
Port       the transport layer (UDP) port number of the peer
Protocol   IANA assigned, Internet Protocol number (17 for UDP)
Reserved   reserved for future use; zero when sent, ignored
           when received
Address    an IPv6 address, or an IPv4 address in "IPv4-Mapped
           IPv6 address" format, of the peer
OSPI       the outbound SPI value the registered host is using for
           the peer with the Address and Port
ISPI       the inbound SPI value the registered host is using for
           the peer with the Address and Port
```

Figure 13: Format of the PEER_PERMISSION Parameter

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).

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |              Type             |             Length            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                           Priority                            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

   Type      [TBD by IANA; 4700]
   Length    4
   Priority  the priority of a (potential) peer reflexive candidate
```

             Figure 14: Format of the CANDIDATE_PRIORITY Parameter

5.15. NOMINATE parameter

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

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |              Type             |             Length            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                           Reserved                            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

   Type      [TBD by IANA; 4710]
   Length    4
   Reserved  Reserved for future extension purposes
```

                Figure 15: Format of the NOMINATE Parameter

6. Security Considerations

   The security considerations are the same as in [RFC5770], but are
   repeated here for completeness sake.

6.1. Privacy Considerations

   The locators are in plain text format in favor of inspection at HIP-
   aware middleboxes in the future. The current document does not
   specify encrypted versions of LOCATOR_SETs, even though it could be
   beneficial for privacy reasons to avoid disclosing them to
   middleboxes.

   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 reasons, an
end-host may exclude certain host addresses from its LOCATOR_SET
parameter. However, 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 TURN server.

The use of HIP and data relays can be also useful for privacy
purposes. For example, a privacy concerned Responder may reveal only
its HIP relay server and Relayed candidates to Initiators. This same
mechanism also 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.

6.2. Opportunistic Mode

A HIP relay server should have one address per relay client when a
HIP relay is serving more than one relay client and supports
opportunistic mode. Otherwise, it cannot be guaranteed that the HIP
relay server can deliver the I1 packet to the intended recipient.

6.3. Base Exchange Replay Protection for HIP Relay Server

In certain scenarios, it is possible that an attacker, or two
attackers, can replay an earlier base exchange through a HIP 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 Responder has to be disconnected from the HIP relay server.

The relay 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-MIDDLE] and drop the I2 or R2 packets if the
corresponding ECHO_RESPONSE_M parameters are not present.

6.4. Demuxing 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.

6.5. Reuse of Ports at the Data Relay

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

7. IANA Considerations

This section is to be interpreted according to [RFC5226].

This document updates the IANA Registry for HIP Parameter Types
[RFC7401] by assigning new HIP Parameter Type values for the new HIP
Parameters: RELAYED_ADDRESS, MAPPED_ADDRESS (defined in
Section 5.12), and PEER_PERMISSION (defined in Section 5.13).

This document also updates the IANA Registry for HIP NAT traversal
modes [RFC5770] by assigning value for the NAT traversal mode ICE-
HIP-UDP (defined in Section 5.4).

This document defines additional registration types for the HIP
Registration Extension [I-D.ietf-hip-rfc5203-bis] that allow
registering with a HIP relay server for ESP relaying service:
RELAY_UDP_ESP (defined in Section 4.1; and performing server
reflexive candidate discovery: CANDIDATE_DISCOVERY (defined in
Section 4.2).

8. Contributors

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

9. Acknowledgments

10. References

10.1. Normative References

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

[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,
           <http://www.rfc-editor.org/info/rfc7401>.

[I-D.ietf-hip-rfc5203-bis]
           Laganier, J. and L. Eggert, "Host Identity Protocol (HIP)
           Registration Extension", draft-ietf-hip-rfc5203-bis-10
           (work in progress), January 2016.

[I-D.ietf-hip-rfc5204-bis]
           Laganier, J. and L. Eggert, "Host Identity Protocol (HIP)
           Rendezvous Extension", draft-ietf-hip-rfc5204-bis-07 (work
           in progress), December 2015.

[I-D.ietf-hip-rfc5206-bis]
           Henderson, T., Vogt, C., and J. Arkko, "Host Mobility with
           the Host Identity Protocol", draft-ietf-hip-rfc5206-bis-12
           (work in progress), May 2016.

   [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,
              <http://www.rfc-editor.org/info/rfc5770>.

   [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing,
              "Session Traversal Utilities for NAT (STUN)", RFC 5389,
              DOI 10.17487/RFC5389, October 2008,
              <http://www.rfc-editor.org/info/rfc5389>.

   [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,
              <http://www.rfc-editor.org/info/rfc7402>.

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

   [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an
              IANA Considerations Section in RFCs", BCP 26, RFC 5226,
              DOI 10.17487/RFC5226, May 2008,
              <http://www.rfc-editor.org/info/rfc5226>.

   [I-D.ietf-ice-rfc5245bis]
              Keränen, A., Holmberg, C., and J. Rosenberg,
              "Interactive Connectivity Establishment (ICE): A Protocol
              for Network Address Translator (NAT) Traversal", draft-
              ietf-ice-rfc5245bis-03 (work in progress), June 2016.

10.2. Informative References

   [RFC4423] Moskowitz, R. and P. Nikander, "Host Identity Protocol
              (HIP) Architecture", RFC 4423, DOI 10.17487/RFC4423, May
              2006, <http://www.rfc-editor.org/info/rfc4423>.

   [RFC5201] Moskowitz, R., Nikander, P., Jokela, P., Ed., and T.
              Henderson, "Host Identity Protocol", RFC 5201,
              DOI 10.17487/RFC5201, April 2008,
              <http://www.rfc-editor.org/info/rfc5201>.

   [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, <http://www.rfc-editor.org/info/rfc5207>.

[RFC5766]   Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using
            Relays around NAT (TURN): Relay Extensions to Session
            Traversal Utilities for NAT (STUN)", RFC 5766,
            DOI 10.17487/RFC5766, April 2010,
            <http://www.rfc-editor.org/info/rfc5766>.

[MMUSIC-ICE]
            Rosenberg, J., "Guidelines for Usage of Interactive
            Connectivity Establishment (ICE) by non Session Initiation
            Protocol (SIP) Protocols", Work in Progress, July 2008.

[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, <http://www.rfc-editor.org/info/rfc5128>.

[HIP-MIDDLE]
            Heer, T., Wehrle, K., and M. Komu, "End-Host
            Authentication for HIP Middleboxes", Work in Progress,
            February 2009.

[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,
            <http://www.rfc-editor.org/info/rfc3948>.

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's recommended that the hosts
estimate the round-trip time (RTT) between them and set the minimum
Ta value so that only two connectivity check messages are sent on
every RTT.

One way to estimate the RTT is to use the time it takes for the HIP
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, it is not recommended.

Appendix B. Base Exchange through a Rendezvous Server

When the Initiator looks up the information of the Responder from
DNS, it's possible that it discovers a rendezvous server (RVS) record
[I-D.ietf-hip-rfc5204-bis]. In this case, if the Initiator uses NAT
traversal methods described in this document, it MAY use its own HIP
relay server to forward HIP traffic to the rendezvous server. The
Initiator will send the I1 packet using its HIP relay server, which
will then forward it to the RVS server of the Responder. In this
case, the value of the protocol field in the RELAY_TO parameter MUST
be IP since RVS does not support UDP-encapsulated base exchange
packets. The Responder will send the R1 packet directly to the
Initiator's HIP relay server and the following I2 and R2 packets are
also sent directly using the relay.

In case the Initiator is not able to distinguish which records are
RVS address records and which are Responder's address records (e.g.,
if the DNS server did not support HIP extensions), the Initiator
SHOULD first try to contact the Responder directly, without using a
HIP relay server. If none of the addresses are reachable, it MAY try
them out using its own HIP relay server as described above.

Authors' Addresses

Ari Keranen
Ericsson
Hirsalantie 11
02420 Jorvas
Finland

Email: ari.keranen@ericsson.com

Jan Melen
Ericsson
Hirsalantie 11
02420 Jorvas
Finland

Email: jan.melen@ericsson.com

Miika Komu (editor)
Ericsson
Hirsalantie 11
02420 Jorvas
Finland

Email: miika.komu@ericsson.com