Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: August 21, 2008

D. Saucez
B. Donnet
O. Bonaventure
Universite catholique de Louvain
February 18, 2008

IDIPS : ISP-Driven Informed Path Selection
draft-saucez-idips-00.txt

Status of this Memo

Copyright Notice

Abstract

This draft describes a simple network-based protocol to facilitate Path Selection and to improve traffic engineering capabilities in multihomed corporate networks. With this protocol, any network device that requires to select a path among a list of different paths asks a Traffic Engineering service called IDIPS (ISP-Driven Informed Path Selection) to obtain an ordered list of the possible paths. The ordering is constructed according to policies and performance

requirements of both the host and network provider.

Table of Contents

1. Introduction

During the last years, we have seen the rising of a set of applications requiring more predictable performances. For instance, current applications such as IPTV and VoD, need higher bandwidth and lower delays. Further, it is now frequent that the content is replicated among a set of servers located anywhere on five continents or event among users themselves (see, for instance, peer-to-peer applications and FTP mirrors). Finally, multihoming is becoming more and more popular.

Today, although many of measurement techniques have been developed within the IPPM working group of the IETF, an application that needs to select a path or a server must implement its own measurement system. Thus, several applications running on the same host or in the same campus will probably perform almost the same kind of measurements

A better solution to this problem would be to develop a service that runs continuously and could be queried by applications requiring performance and path information. This draft introduces this service named IDIPS (for ISP-Driven Informed Path Selection).

The key concept of IDIPS is to move the Path Selection from the end-systems (hosts) to a specific path selection network service that is aware of the host and network provider traffic engineering (TE) requirements and of the use of the network.

IDIPS is based on two entities: the Client and the Server. First, the Client determines its source and destination addresses (or prefixes) and its DSCP [DSCP]. Second, the Server determines all the possible paths from the Client Request and order these paths according to some criteria. The ordering can be a function of the DSCP given by the Client. In addition, the ordering should be a function of some criteria of the network provider like TE requirements or administrative policies.

Like NAROS [NAROS], the principle of IDIPS is that hosts contact the IDIPS service before establishing new flows. IDIPS determines which (source, destination) pair the host should use in order to match its requirements.

The main difference between IDIPS and NAROS is that IDIPS is able to determine the "best" (source, destination) pair while NAROS is only able to determine the "best" source address for a given destination address.

This document specifies the basic IDIPS protocol. The companion

document [IDIPS] gives a motivation for the protocol and presents some possible applications of IDIPS.

The IDIPS protocol is built on extensions like IPv6 [RFC2460].

Section 2 defines the terminology used in this document. Section 3 gives an overview of the protocol. Section 4 determines the generic structure of the protocol and Section 5 defines precisely the structure and content of every IDIPS message. Section 6 presents some data types used in the protocol. Finally, Section 8 considers protocol security issues.

2. Terminology

Path Selection:
: Path Selection is a process used to determine the path(s) to use to reach a given destination among all the available paths from a particular source.

Request:
: a Request is a message sent by a Client to a Server to obtain an ordered list of the "best" paths to reach a destination (for some definition of best).

Response:
: a Response is a message sent by a Server to a Client in response to a Request. The Response provides a possible ordering of the potential paths from the request according to some criteria. If no ordering is possible, an Error is sent.

Error:
: an Error is a message sent by a Server to a Client in response to a Request when this Request cannot be process (bad Request or no possible ordering).

Client:
: a Client is an entity that sends Requests to a Server.

Server:
: a Server is an entity that receives Requests from Clients and processes the Requests to provide ordering of paths according to some criteria.

DSCP:
: the Differentiated Services Field Codepoint gives information about the type of the service of the requested/selected path(s).

3. IDIPS Protocol

The IDIPS protocol runs over UDP to allow IDIPS servers to use anycast [RFC1546].

The IDIPS protocol is a message-based protocol with two main types of message. First, the IDIPS_REQUEST message is sent by a Client to a Server. Second, the Server replies with an IDIPS_RESPONSE message. The processing of these messages is explained in Section 5.

IDIPS works with prefixes instead of addresses while most applications work with addresses. To translate an address into a prefix, the prefix address is set to the address and the prefix length is set to the number of bits of the address (i.e., 128 in IPv6 and 32 in IPv4). For example, address 2000::1 is translated in prefix 2000::1/128.

Figure 1 presents a typical topology with multihomed hosts. Host X is connected to the Internet through ISP A and ISP B with routers R1A and R1B respectively. Host Z is connected to the Internet trough ISP B and ISP C with router R2B and router R2C respectively. In such an environment, multiple paths are possible between X and Z. We assume in this example that Shim6 is used to ensure multihoming on hosts X and Z.

Problem statement

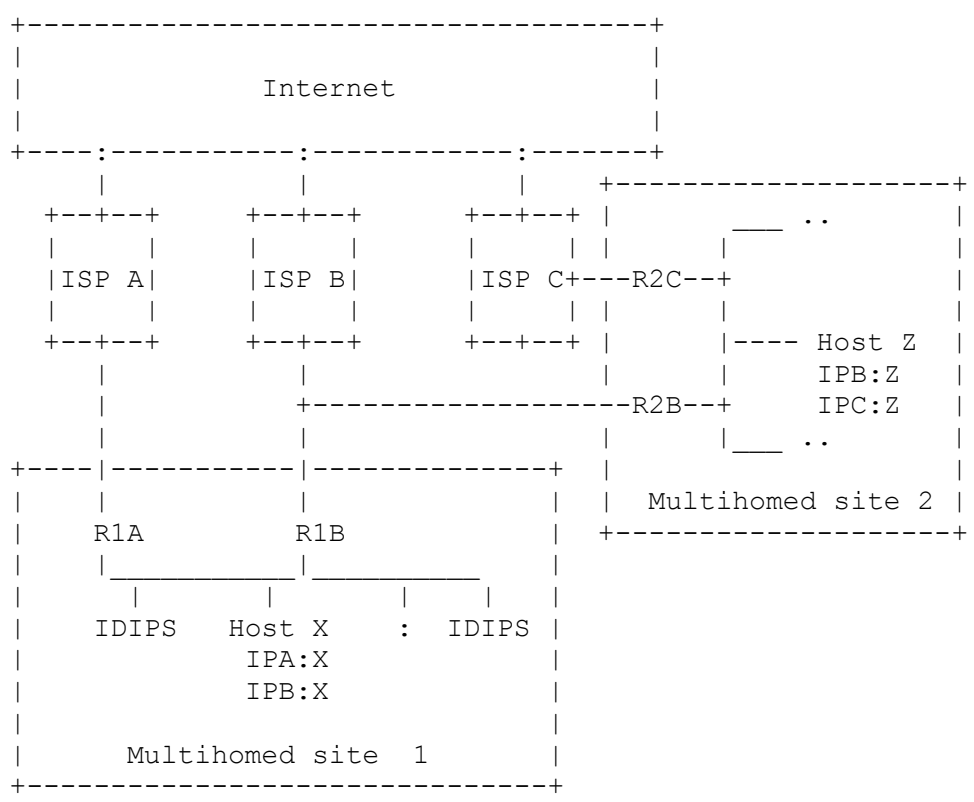

Figure 1

For instance, X wants to access a specific content available on a multihomed host. In this example, we observe that X has multiple choices with potentially different performances (e.g., bandwidth, cost, latency, etc). To determine the best path, X sends to its IDIPS service a Request with all its source addresses (i.e., IPA: X,IPB:X) and all the possible destination addresses (i.e., IPB:Z,IPC:Z). The Server replies with a list of pairs (source,destination) ordered such that the first entry corresponds to the most promising path.

Figure 2 depicts a high-level view of the protocol for this query/ response protocol.

Message exchanges between a client and an IDIPS server

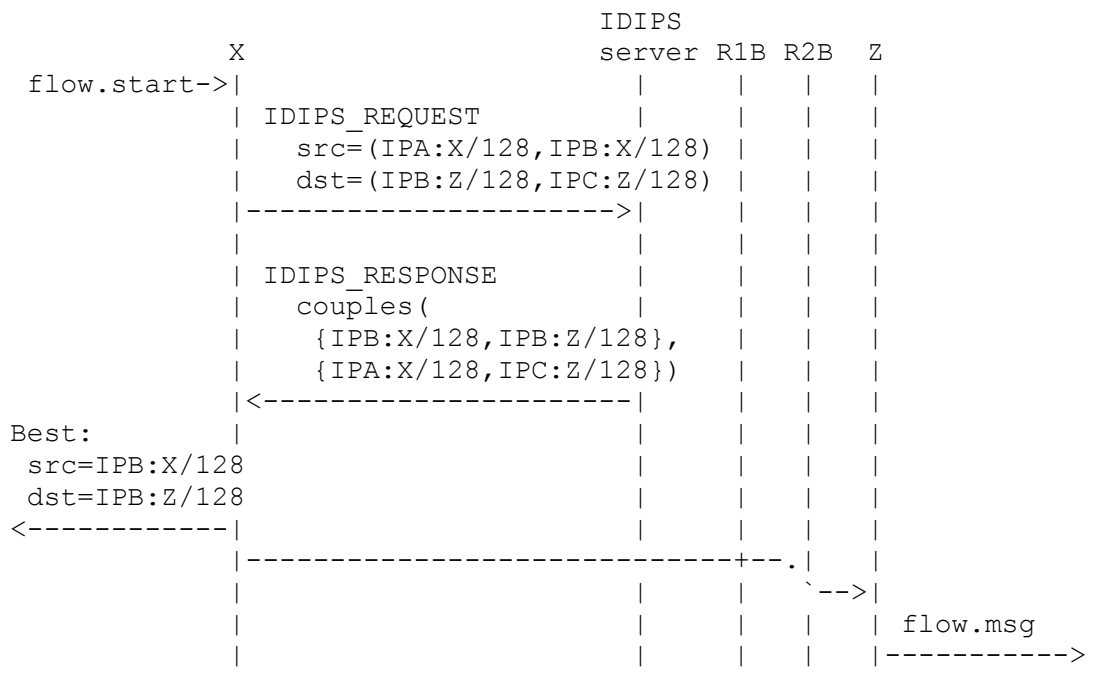

Figure 2

4. IDIPS Packet Format

IDIPS messages are composed of one or more extensions. Every extension contains the type of the next extension.

The size of the IDIPS extensions MUST be a multiple of 4 bytes.

IDIPS messages are formatted as shown on Figure 3.

IDIPS message

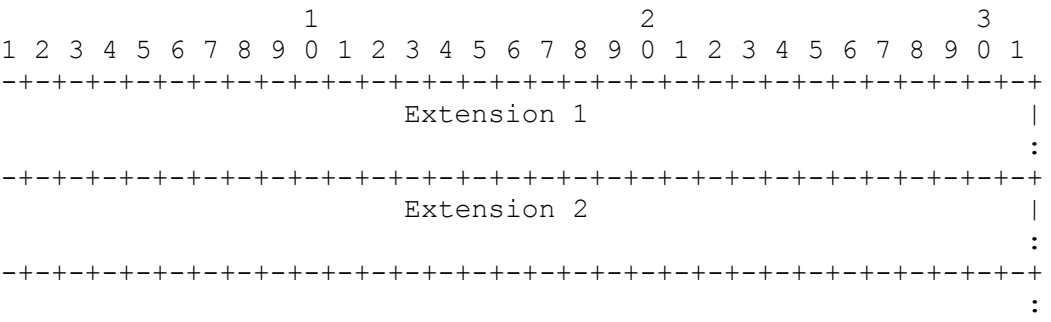

Figure 3

4.1. IDIPS Extension Format

Extensions are divided in two parts: an Extension header and an Extension payload.

Extensions MUST have an header. The payload is optional. The optional payload immediately follows the header.

The structure of the Extensions header is the following:

Extension header structure

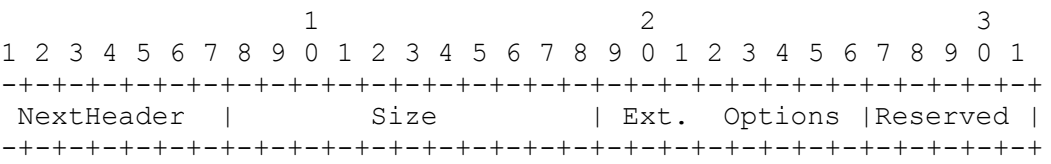

Figure 4

NextHeader:
: 8-bits selector. Identifies the type of the header immediately following the current extension. The types specified in this document are listed in Section 4.2.

Size:
: 11-bits unsigned integer. Specifies the payload extension length. The size is the number of 32-bits words of the payload.

Ext. Options:
: 8-bits. Extension Options is reserved for the extension. This field COULD be used freely to give information about the extension payload. If the extension does not use Extension Options, all bits MUST be set to 0.

Reserved:
: 5-bits. This field is reserved for future usage. All bits MUST be set to 0 in transmission and ignored on reception.

4.2. Extension Types

This section lists IDIPS extension type values. Current allocations are:

Extension types

Type	Description	Direction
0	NONE	
1	IDIPS_REQUEST	Client --> Server
2	IDIPS_RESPONSE	Client <-- Server
3	IDIPS_ERROR	Client <-- Server
255	IDIPS_HEADER	Client <--> Server

Figure 5

- NONE: no extension. This type is used to indicate the end of the message.
- IDIPS_REQUEST: request for the paths ordering according to the payload.
- IDIPS_RESPONSE: ordered paths list for a given request (IDIPS_REQUEST).
- IDIPS_ERROR: information about the error that occurred. Error notification is optional. A server CAN fail silently.
- IDIPS_HEADER: header of any IDIPS message.

5. IDIPS Extensions

5.1. IDIPS_HEADER Extension

Each IDIPS message MUST begin with the IDIPS_HEADER extension. This extension provides basic information about the message.

IDIPS_HEADER extension

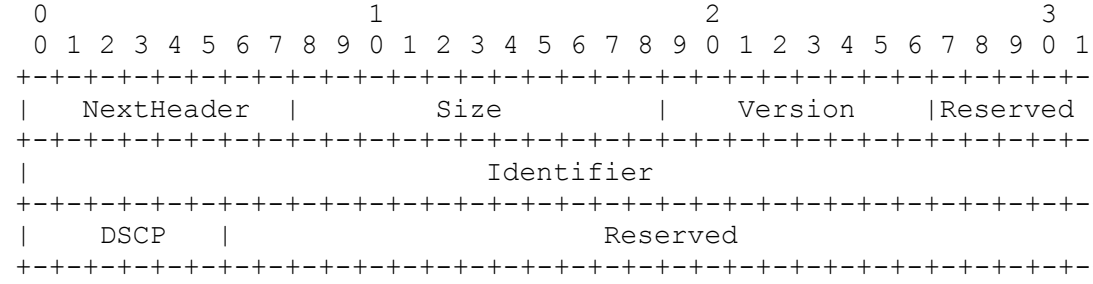

Figure 6

5.1.1. IDIPS_HEADER Extension Header

Version:
: 8-bits selector. The Extension Option field determines the protocol version number. This specification defines version 1.

5.1.2. IDIPS_HEADER Extension Payload

Identifier:
: 32-bits unsigned integer. Identifies the IDIPS message. The Identifier CAN be identical for different Clients and MAY be unique for a given Client for a more or less long time. The Identifier MUST be the same for each retransmission of the same message. The Identifier of an answer to an IDIPS message MUST be equal to the Identifier of the Request.

DSCP:
: 6-bits selector. Differentiates the service. Indicates the DSCP that must be used when forwarding packets. Default is B'000000' (i.e., best effort). See [DSCP] for the possible values and their meaning.

Reserved:
: 24-bits. Reserved for a future usage. All bits MUST be set to 0.

5.2. IDIPS_REQUEST Extension

To construct an IDIPS_REQUEST, the Client determines all the possible source prefixes, all the possible destination prefixes, and optionally the DSCP for the flow it wants to optimize. If the Client only supports addresses, the IDIPS layer MUST translate all the addresses into prefixes as explained in Section 3.

When an IDIPS_REQUEST is received by a Server, the Server SHOULD construct all the possible paths form the sources and destinations of the Request and order them. This ordering depends on the Path Selection algorithm and on the DSCP if it is different from B'000000'. If successful, the ordering of the paths is returned to the Client inside an IDIPS_RESPONSE. Otherwise, an IDIPS_ERROR SHOULD be sent to the Client. The Server CAN silently drop Requests if the load is too high.

IDIPS_REQUEST Extension

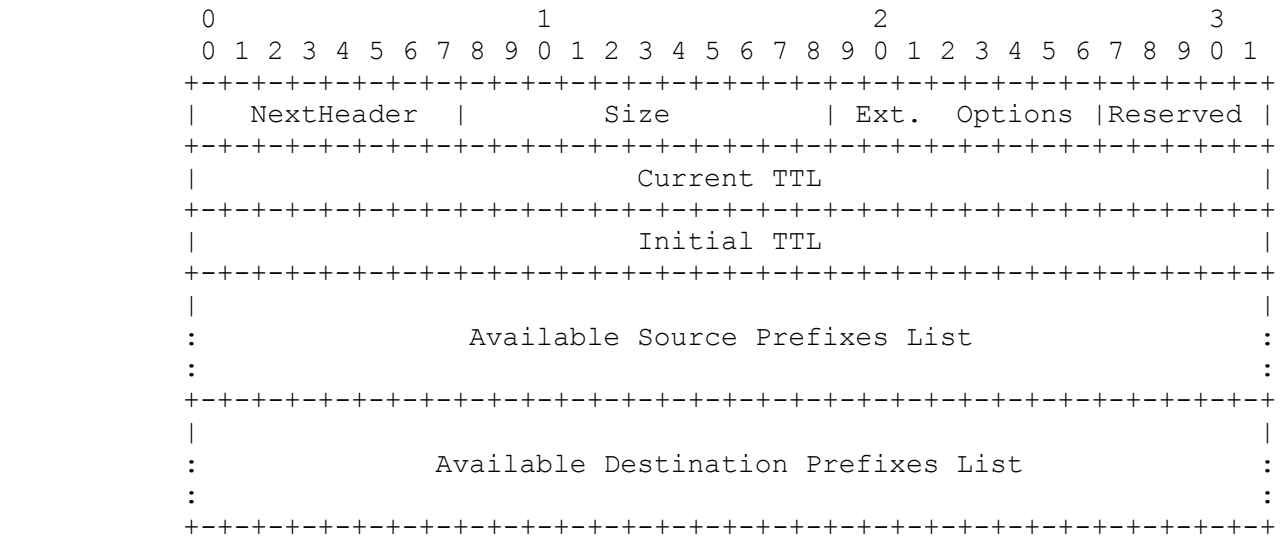

Figure 7

5.2.1. IDIPS_REQUEST Extension Header

Ext. Options:
8-bits. Not used, all bits MUST be set to 0 on transmission and ignored on reception.

5.2.2. IDIPS_HEADER Extension Payload

Current TTL:
: 32-bits unsigned integer. Indicates the current Time-To-Live (TTL) of the request. If this is a new request, the current TTL is set to 0. Otherwise, the current TTL is set to the time remaining before the expiration of the last response for this request. If the previous response has expired, the TTL is set to 0. The time is expressed in seconds.

Initial TTL:
: 32-bits unsigned integer. Indicates the TTL given by the IDIPS Server for the previous Request with the same parameters. If this is a new request, the initial TTL is set to 0. The time is expressed in seconds.

Available Source Prefixes List
: variable length IDIPS list. Indicates the list of source prefixes that the IDIPS Server SHOULD use in its Path Selection process. The structure of the list is described in Section 6.4.

Available Destination Prefixes List
: variable length IDIPS list. Indicates the list of destinations prefixes that the IDIPS Server SHOULD use in its decision process. The structure of the list is described in Section 6.4.

5.3. IDIPS_RESPONSE Extension

An IDIPS_RESPONSE is the normal reply from a Server to a Client for an IDIPS_REQUEST. The Response is mainly composed of a list of pairs. The first element of a pair MUST be one of the prefixes in the Available Source Prefixes List of the IDIPS_REQUEST or a more generic prefix that includes at least one of these prefixes. The second element of a pair MUST be one of the prefixes in the Available Destination Prefixes List of the IDIPS_REQUEST or a more generic prefix that includes at least one of these prefixes. The DSCP of the Response CAN differ from the one in the Request but MUST be the one used for the ordering. The Identifier of the Response MUST be the same as the Request Identifier.

When receiving an IDIPS_RESPONSE, a Client determines the Request associated to the Response. The Client is free to implement the processing of the Response but MUST reject any invalid Response (e.g., bad identifier) and SHOULD use a pair with the highest priority for the flow that triggered the Request. As the Best Pairs List is ordered by priority, a good way to implement it SHOULD be to

take the first pair in the list and if it does not work the second, and so on until a valid pair is used.

If a Client does not receive an answer for a Request after a given time, it CAN send the same Request again. We recommend to wait 3 seconds before retransmission and make at most 3 retransmissions. Servers CAN cache the Response of a specific Request to reduce processing while retransmission. When caching is supported, the responses MAY be cached for at least (number of maximum retransmission * maximum time between retransmission) unit of time.

If no Response is received or if the response is invalid or is an Error or if there is no valid path within returned list, the client SHOULD select the path according to the RFC3484 [RFC3484] for IPv6 or at random for IPv4.

IDIPS_RESPONSE Extension

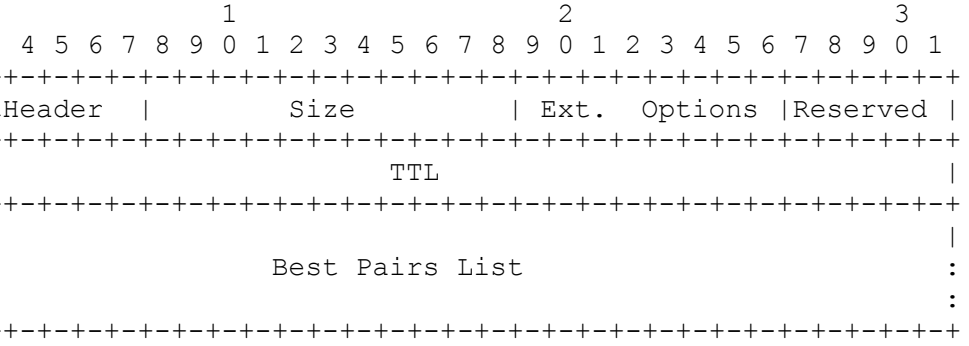

Figure 8

5.3.1. IDIPS_RESPONSE Extension Header

Ext. Options:
8-bits. Not used, all bits MUST be set to 0.

5.3.2. IDIPS_RESPONSE Extension Payload

TTL:
32-bits unsigned integer. Indicates the validity time of the response. Time is expressed in seconds.

Best Pairs List:
variable length IDIPS list. Indicates the list of pairs ordered according to the Path Selection algorithm. The structure of the list is presented in Section 6.3.

5.4. IDIPS_ERROR Extension

IDIPS_ERROR extensions are used to indicate an error. The Server MUST never send another IDIPS message after sending an Error.

An IDIPS_ERROR is always the result of a message reception. The IDIPS_ERROR is sent to the source of the message generating the error.

The Server CAN fail silently.

IDIPS_ERROR Extension

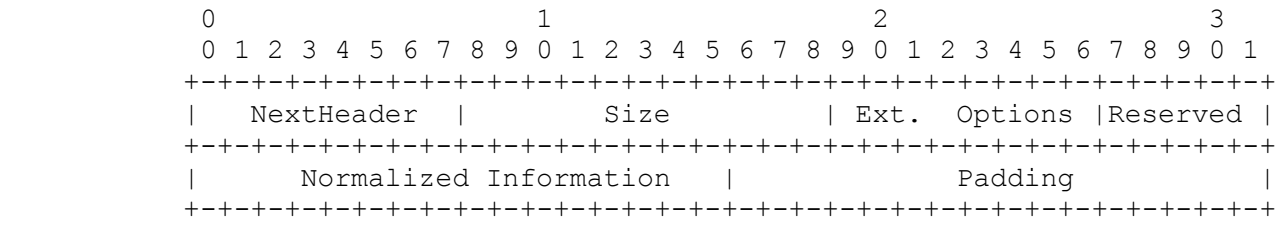

Figure 9

5.4.1. IDIPS_ERROR Extension Header

Ext. Options:
8-bits. Not used, all bits MUST be set to 0.

5.4.2. IDIPS_ERROR Extension Payload

Normalized Information:
16-bits selector. Gives information about the error.

Figure 10 defines the Normalized Information values.

Error type

Description	Value
UNKNOWN_ERROR	0x1
INTERNAL_SERVER_ERROR	0x2
UNSUPPORTED_IDIPS_VERSION	0x3
ILLEGAL_SYNTAX	0x4
BAD_SYNTAX	0x5
NO_ROUTE	0x6
ADMINISTRATIVELY_PROHIBITED	0x7

Figure 10

- UNKNOWN_ERROR is used when an error occurs but does not fit within any other error code.
- INTERNAL_SERVER_ERROR is sent when the Server encounters an internal error that it cannot recover from.
- UNSUPPORTED_IDIPS_VERSION is used when the IDIPS version of the message received is not supported. The IDIPS version is defined in the IDIPS_HEADER Extension Option field.
- ILLEGAL_SYNTAX is used when one or more fields contain an illegal value.
- BAD_SYNTAX is used when the message is badly constructed and triggers a parsing error.
- NO_ROUTE is used when no solution can be found for any prefix in the IDIPS_REQUEST that initiated the error.
- ADMINISTRATIVELY_PROHIBITED is used when there exists a solution but it is prohibited for administrative reasons (i.e., restriction policies).

5.5. IDIPS Message Construction

This section presents how to construct the three common IDIPS messages: (i) the Request, (ii) the Response and (iii) the Error.

5.5.1. Requests

Figure 11 shows the format of the Requests.

Request Message

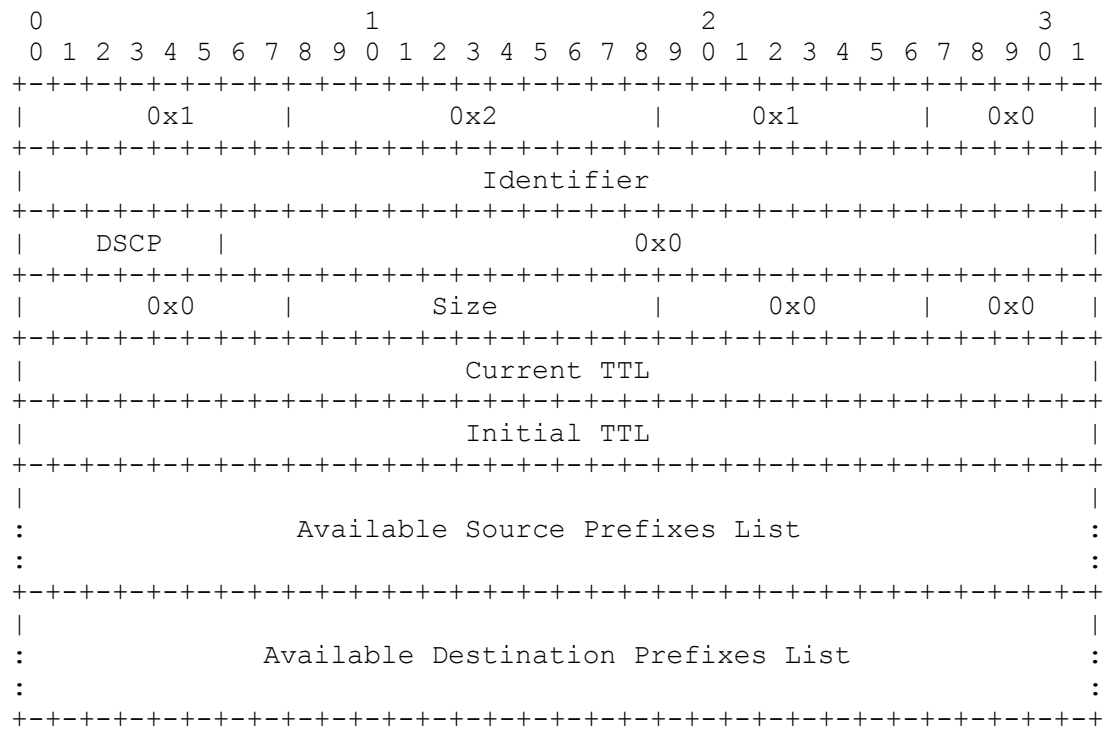

Figure 11

5.5.2. Responses

Figure 12 shows the format of the Responses.

Response Message

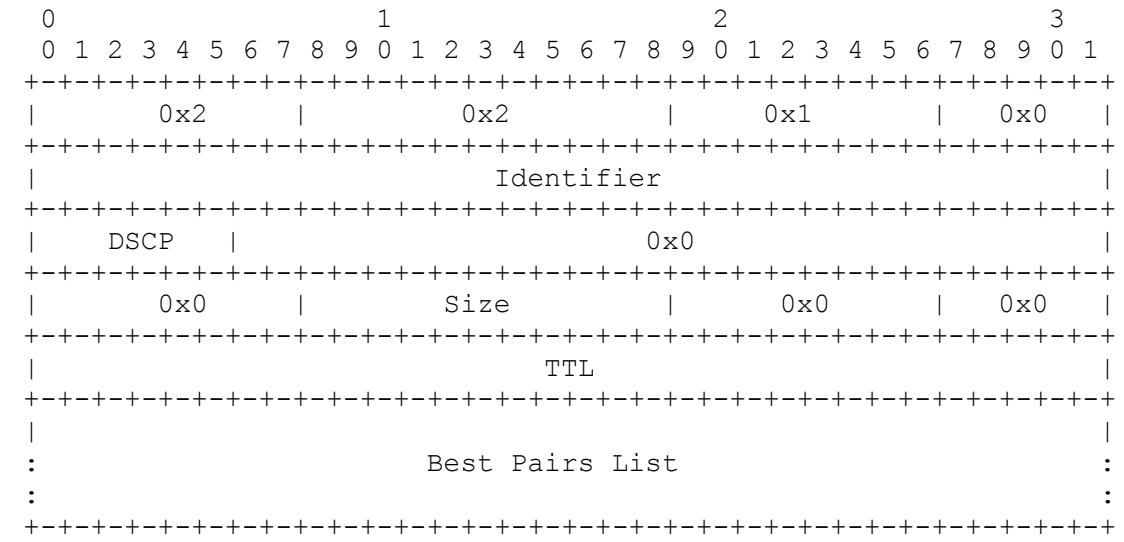

Figure 12

5.5.3. Errors

Figure 13 shows the format of the Errors.

Error Message

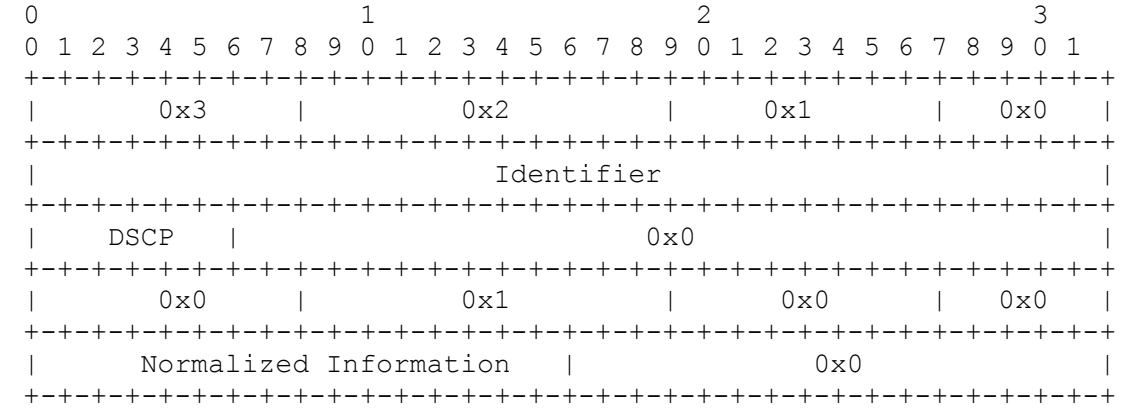

Figure 13

6. Data Structure in IDIPS Messages

6.1. List

A list is a collection of entries having the same type.

List structure

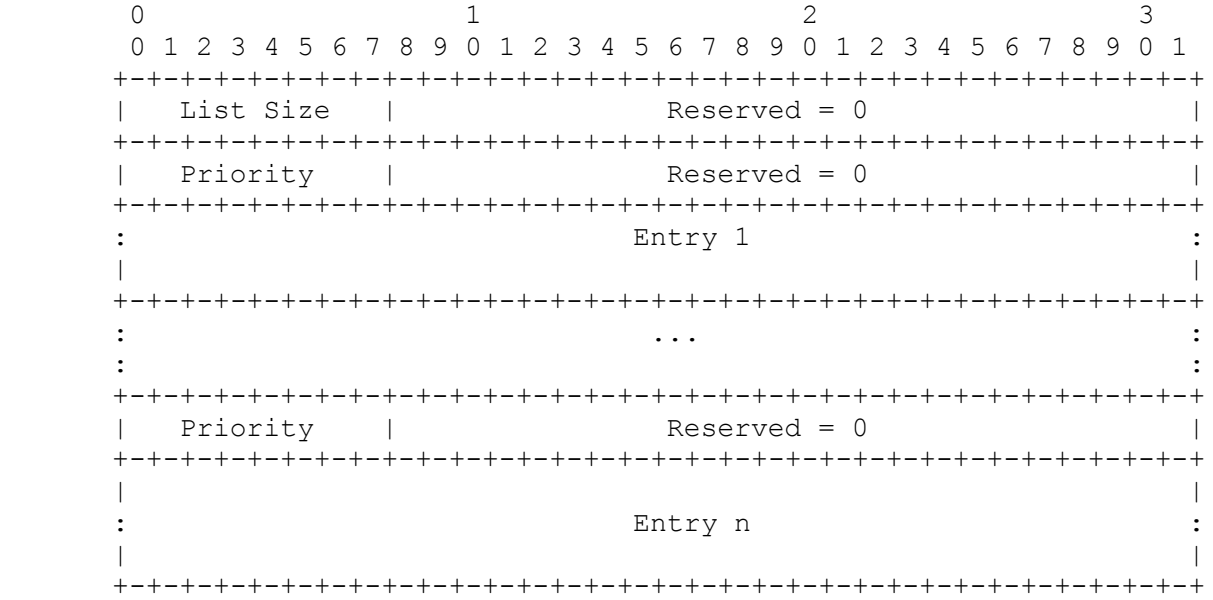

Figure 14

List Size:
8-bits unsigned integer. Indicates the number of entries in the list. A list contains zero, one or more entries. A list contains at most 255 entries.

Reserved:
24-bits. This field is reserved for future usage. All bits MUST be set to 0 in transmission and ignored on reception.

Priority:
8-bits unsigned integer. A priority is assigned to each entry. A value of 0 gives the maximum priority while a value of 255 gives the lowest possible priority. Entries are ordered by Priority in the lists, hence, the i'th entry in a list has always a higher or equal priority than the i'th + 1 entry.

6.2. Prefixes

The prefixes are represented in IDIPS as follows

IDIPS prefixes

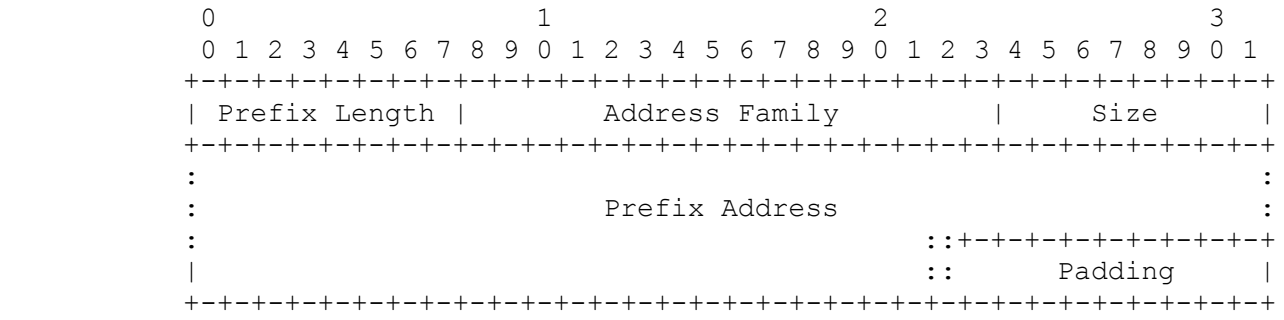

Figure 15

Prefix Length:
8-bits unsigned integer. Indicates the length in bits of the prefix. A Prefix Length of 0 indicates an address.

Address Family:
16-bits selector. Indicates the address family of the prefix. The complete list of address families can be found at [ADDR]. The following table shows the address families that MUST be implemented by any Server and Client:

Addresses families

Family	Description
1	IP (IP version 4)
2	IP6 (IP version 6)

Figure 16

Reserved:
8-bits. This field is reserved for future usage. All bits MUST be set to 0.

Size:
8-bits. Indicates the size in bits of the prefix address.

Prefix Address:
Variable length. Prefix Address is the address of the prefix presented in its standardized binary form.

Padding:
Variable length. Padding is used if the Prefix Address size is not a multiple of 32 bits. All bits MUST be set to 0. The Padding is at most of 31 bits.

NOTE: the overall size of the tuple (Prefix Length, Address Family, Size, Prefix Address, Padding) MUST be a multiple of 32 bits.

For example, the fields of the prefix 130.104.0.0/16 in IPv4 are (16, 1, 130.104.0.0) for Prefix Length, Address Family and Prefix, respectively.

6.3. Prefix Pair

The Prefix Pair structure indicates a pair of prefixes. The first prefix is the source prefix and the second is the destination.

Prefix Pair structure is presented below:

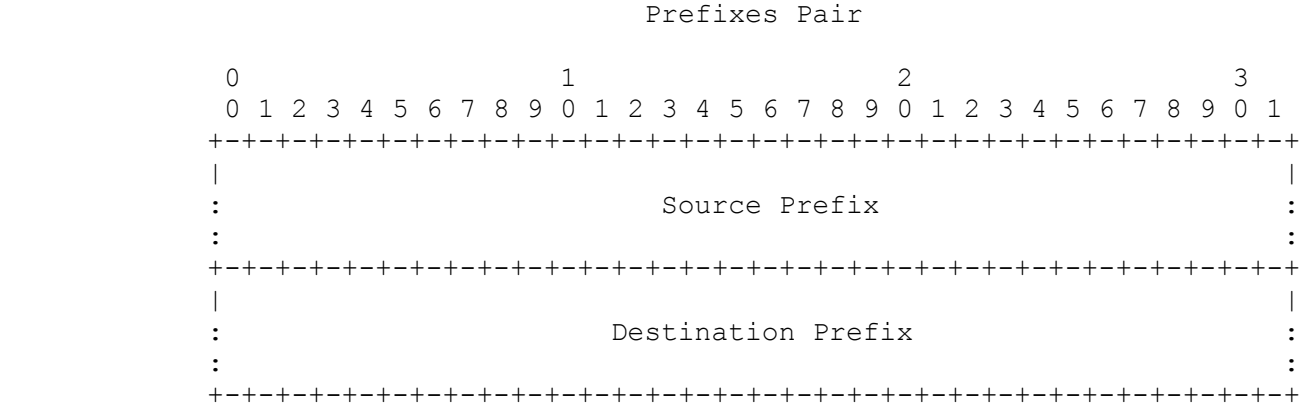

Figure 17

6.4. Prefixes List

Prefixes List is a list (described in Section 6.1) containing one or more prefixes (Section 6.2).

For IPv4 prefixes, the Entry Length field of the list is set to 2 ((32+32)/32). For IPv6 prefixes, the length is set to 5 ((128+ 32)/32).

6.5. Prefixes Pair List

Prefix Pair List is a list (described in Section 6.1) containing zero, one or more Prefixes Pairs (Section 6.3).

For IPv4 pairs, the Entry Length field of the list is set to 4 (((32+ 32)*2)/32). For IPv6 pairs, the length is set to 10 (((128+ 32)*2)/32).

7. IANA

The following considerations will be requested to the IANA:

- a UDP port number for the Server.
- an IPv4 anycast address to reach the service.
- an IPv6 anycast address to reach the service.

8. Security Considerations

The current version of IDIPS assumes a trust relationship between Clients and Servers.

The IDIPS protocol can introduce security concerns as the Server choice can have an impact on the the path followed by the packets of Clients relying on the IDIPS decisions. An attacker could divert a part of the traffic to a given path and thus alter the performances of the network or make eavesdropping.

Extensions and recommendations will be proposed to add authenticity and privacy to the IDIPS protocol.

9. Conclusion

The companion document [IDIPS] proposes an informed path selection service that is able to rank paths based on policy and performance criteria. In this document, we propose IDIPS, a protocol to support such a service.

IDIPS is a simple service that proposes to move the Paths Selection algorithm from Clients to a specific network service able to estimate paths performances according to some criteria in a scalable manner.

The IDIPS service works in anycast and is based on two entities. On

one side, the Clients are normal hosts that want to select a path among a list of multiple possibilities. On the other side, Servers continuously monitor the network to efficiently determine the best paths to use when multiple choices are possible.

10. References

10.1. Normative References

[ADDR] IANA, "Address Family Numbers", February 2007.

[DSCP] IANA, "Differentiated Services Field Codepoints", March 2002.

[RFC1546] Partridge, C., Mendez, T., and W. Milliken, "Host Anycasting Service", RFC 1546, November 1993.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC3484] Draves, R., "Default Address Selection for Internet Protocol version 6 (IPv6)", RFC 3484, February 2003.

10.2. Informative References

[IDIPS] Bonaventure, O., Saucez, D., and B. Donnet, "The case for an informed path selection service", Internet-Draft draft-bonaventure-informed-path-selection-00, February 2008.

[NAROS] de Launois, C. and O. Bonaventure, "NAROS : Host-Centric IPv6 Multihoming with Traffic Engineering", draft-de-launois-multi6-naros-00.txt (work in progress), May 2003.

Appendix A. Acknowledgments

Damien Saucez is supported by the European-founded 027609 Agave project. Benoit Donnet is supported by the European-founded 034819 OneLab project.

The authors would like to gratefully acknowledge people who have contributed discussion and ideas to the making of this proposal: Sebastien Barre, Pierre Francois, Luigi Iannone and Bruno Quoitin.

Authors' Addresses

Damien Saucez
Universite catholique de Louvain
Place Sainte Barbe 2
Louvain-la-Neuve, 1348
Belgium

Email: damien.saucez@uclouvain.be
URI: http://inl.info.ucl.ac.be

Benoit Donnet
Universite catholique de Louvain
Place Sainte Barbe 2
1348
Belgium

Email: benoit.donnet@uclouvain.be
URI: http://inl.info.ucl.ac.be

Olivier Bonaventure
Universite catholique de Louvain
Place Sainte Barbe 2
Louvain-la-Neuve, 1348
Belgium

Email: olivier.bonaventure@uclouvain.be
URI: http://inl.info.ucl.ac.be

Full Copyright Statement

Intellectual Property

Acknowledgment

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