Network Working Group
INTERNET-DRAFT

R. R. Stewart
Cisco Systems
Q. Xie
Motorola

expires in six months

November 15,2000

Aggregate Server Access Protocol (ASAP)
<draft-xie-rserpool-asap-01.txt>

Status of This Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC 2026. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Abstract

Aggregate Server Access Protocol (ASAP) in conjunction with ENRP [ENRP] provides a
high availability data transfer mechanism over IP networks. ASAP uses a name-based addressing model which isolates a logical communication endpoint from its IP address(es), thus effectively eliminating the binding between the communication endpoint and its physical IP

address(es) which normally constitutes a single point of failure.

In addition, ASAP defines each logical communication destination as a named group, providing full transparent support for server-pooling and load sharing. It also allows dynamic system scalability - members of a server pool can be added or removed at any time without interrupting the service.

ASAP is designed to take full advantage of the network level redundancy provided by the Stream Transmission Control Protocol (SCTP) [SCTP]. But it can also use other transport protocol like TCP.

The high availability server pooling is gained by combining two parts, namely ASAP and the Endpoint Name Resolution Part (ENRP). ASAP provides the user interface for name to address translation, load sharing management, and fault management. ENRP defines the high availability name translation service.

Table Of Contents

<TBD>

1. Introduction

Aggregate Server Access Protocol (ASAP) in conjunction with ENRP [ENRP] provides a
high availability data transfer mechanism over IP networks. ASAP uses a name-based addressing model which isolates a logical communication endpoint from its IP address(es), thus effectively eliminating the binding between the communication endpoint and its physical IP address(es) which normally constitutes a single point of failure.

When multiple receiver instances exist under the same name, a.k.a, server pool redundancy, ASAP will select one receiver instance, based on the current load sharing policy indicated by the name, and deliver the message to the selected receiver instance.

While delivering the message, ASAP monitors the reachability of the selected destination instance. If it is found unreachable, before notifying the sender the failure, ASAP can automatically select another instance (if one exists) under that name and attempt to deliver the message to that instance. In other words, ASAP is capable of transparent fail-over amongst instances of a server pool if the selected instance is unreachable.

ASAP uses the Endpoint Name Resolution Part (ENRP) to provide a high availability name space. ASAP is responsible for the abstraction of the underlying transport technologies, load distribution management, fault management, as well as the presentation to the upper layer (i.e., the ASAP user) a unified primitive interface.

When SCTP [SCTP] is used as the transport layer protocol, ASAP can seamlessly incorporate the link-layer redundancy provided by the SCTP.

This document defines ASAP portion of the high availability server pool. ASAP depends on the services of a high availiablity name space a.k.a. ENRP.

1.1 Motivation

In this section, we will discuss the motivation for developing ASAP. Our discussion will be focused on the analysis of the inadequateness of two existing technologies, namely CORBA and DNS, in providing solutions to fault-tolerance design of IP distributed applications.

1.1.1 CORBA and Its Limitations

Often referred to as a Distributed Processing Environment (DPE), CORBA was mainly designed to provide location transparency for distributed applications. However, the following limitations may exist when applying CORBA to the design of real time fault-tolerant system:

1) CORBA is traditionally weak in high availability. The recent development of a high availability version of CORBA by OMG is perhaps a step in the right direction towards fixing this weakness. Nevertheless, the maturity, implementablity, and real-time performance of the design is yet to be proven.

   [Editor's Note: the fault tolerance mechanism being developed by OMG for CORBA bears quite some similarities to ASAP.]

2) CORBA's distribution model encourages an object-based view, i.e., each communication endpoint is normally an object. We consider this kind of granularity too fine to be efficient and effective for designing real-time fault-tolerant applications.

3) CORBA, in general, has a large signature that makes the use of it a challenge in real-time environments. Small devices with limited memory and CPU resource (e.g., H323 or SIP terminals) will find CORBA hard to fit in.

4) CORBA uses TCP as its transport (Note, some effort is currently underway to separate CORBA from its transport). This makes CORBA suffer from the same limitations of TCP in terms of real-time and fault-tolerance performance.

5) CORBA has long lacked easily usable support for the asynchronous communication model, and this may be an issue in many applications. An apparently improved API for asynchronous communication has been added to the CORBA standards only recently, and many, if not most, CORBA implementations still do not support

it. There is as yet insufficient user experience with it to make conclusions regarding this new feature's usability.

1.1.2 DNS and Its Limitations

Undoubtedly DNS is the best-known and proven IP namespace mechanism. However, namespace function alone does NOT provide real time fault-tolerance solution. Other mechanisms and procedures, such as server process failure detection, back-up server control and selection, fast fail-over/switch-over, load balancing, etc., also play crucial roles. These functions are supported by ASAP but not by DNS DNS provides a loose binding where as ASAP and ENRP are designed to provide a tight binding.

As will be further elaborated later in this document, the fault tolerant design for server pools is made up in two parts, namely ASAP and ENRP, where ENRP defines a light-weight yet highly efficient namespace mechanism optimized for building real time fault-tolerant applications. Nevertheless, ASAP does not restrict itself to ENRP for namespace services. In fact, it is not only feasible but also desirable in the future to generalize ASAP design so that the ENRP can provide a generic interface that is capable of inter-working with different namespace, including DNS, at the ASAP implementor's choice.

In the following, we list some limitations of the current DNS namespace capability when compared to that of ENRP:

1) DNS name registration and translation services have been primarily optimized for host names. But we consider namespace services optimized for process names or endpoint names more appropriate and efficient for supporting real time server-level fault-tolerance applications.

2) DNS is primarily passive. It provides a query/response database that allows one to find information, but does NOT provide monitoring of hosts or processes to assure that the host or process associated with that IP address will be alive and able to provide service.

3) DNS has dynamic extensions but is not designed around the dynamic fast changing process address space that is typical to real time distributed applications.

It has also been suggested that ASAP be extended to work with DNS to bridge multiple ASAP planes and provide an "inter-ASAP-Domain" bridging function.

1.2 Definitions

This document uses the following terms:

Operation scope --- the part of the network visible by ENRP.

ASAP Endpoint --- a logical entity in the operation scope which implements the ASAP stack and is capable of sending and receiving messages.

ASAP Node --- a host machine in the network which contains one or more ASAP endpoints.

ASAP Plane or ASAP operational domain --- A relam of tight binding controlled by a set of one or more ENRP deamons. A process or application entitiy registers its name within this "plane" and is peroidically checked for sanity. Note that a plane does not imply geographical locality.

Endpoint name --- the registered tag of a ASAP endpoint, consisting of a NULL terminated ASCII string of fixed length.

Named group --- a group of ASAP endpoints sharing the same endpoint name in the name space.

Endpoint handle --- a logical pointer, consisting of a name and the primary destination transport address to a particular endpoint in a named group.

ENRP client --- a ASAP endpoint using ENRP to obtain name translation and other related services. In this document, the term "ASAP endpoint" is exchangeable with "ENRP client", unless otherwise stated.

ENRP maintenance client --- a ASAP endpoint that has the additional capability of exchanging ENRP maintenance messages with an ENRP server in order to perform certain maintenance functions.

ENRP server --- a server program running on a node that manages the name space collectively with its peer ENRP servers and replies to the service requests from any ENRP client.

Home ENRP server --- the ENRP server to which an endpoint currently belongs. Endpoints normally choose the ENRP server on their local host as the home ENRP server, if one exists. An endpoint shall only have one home ENRP server at any given time, and both the endpoint and the server shall keep track of this master/slave relationship between them.

ENRP server takeover --- the event that a remote ENRP server takes the ownership of all the ENRP endpoints on a node and becomes their home server.

Caretaker ENRP server --- The ENRP server on a remote node which takes ownership of all endpoints on the local node because of the absence of an active local server.

ENRP client channel --- the communication channel through which a ASAP endpoint requests for ENRP service. The ENRP client channel is usually defined by the transport address of the home server and a well known port number.

ENRP server channel --- defined by a well known multicast IP address and a well known port number. All ENRP servers in an operation scope can communicate with one another through this channel. Endpoints are also allowed to communicate on this channel occasionally.

ENRP name domain --- defined by the combination of the ENRP client channel and the ENRP server channel in the operation scope.

Network Byte Order: Most significant byte first, a.k.a Big Endian.

PU --- A server pool user, i.e. one who uses the server pool.

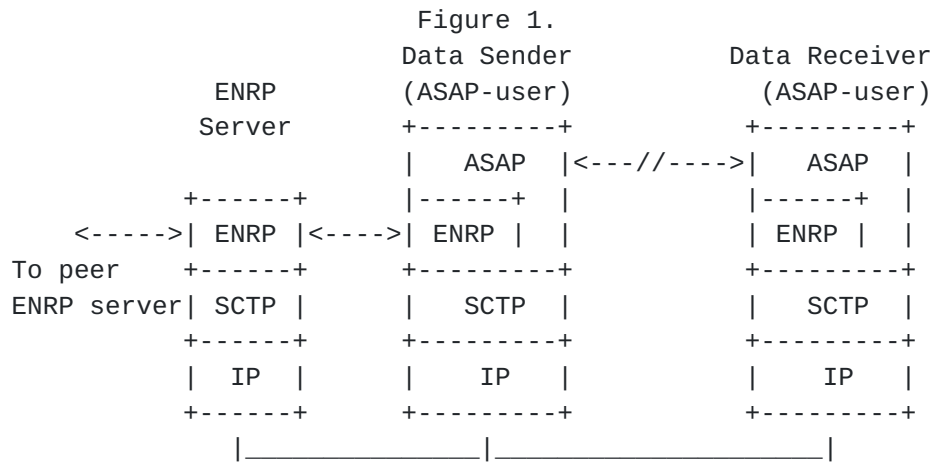
1.3 Protocol overview

At startup, each endpoint in a ASAP operational domain registers its name in the name space. Here a name is defined as a NULL terminated ASCII string of fixed length.

When sending a message, the sender addresses the receiver endpoint by its name and passes the message to its ASAP layer. The ASAP layer, with the help from the ENRP name space daemon server(s), translates the name to a valid transport address (or a list of transport addresses if the receiver is multi-homed) and sends the message to the receiving endpoint.

The following diagram illustrates the components of ASAP and their relationships.

Figure 1.



Multiple endpoints can register themselves under the same name. In that case they will be treated as a receiver pool, and ASAP, when sending a message addressed to that name, will use a predefined load-sharing policy to determine which endpoint(s) in the pool to

send (or address) the message.

ASAP design has a high emphasis on seamless support of "server pooling", high availability, dynamic scalability, and close-to-real-time name translation.

In particular, ASAP can be characterized by:

A) Seamless support of "server pooling" --- ASAP allows multiple servers to register under the same name. It also allows servers to be dynamically add to or removed from a server pool without any reconfiguration.

B) Support automatic receiver "fail-over" --- when the chosen message receiver fails, ASAP, with pre-stated permission from its upper layer, can automatically re-direct the message to an alternative server under the same name if one exists.

C) Transaction management by nickname or "association handle" --- this is to allow a continuous transaction or session consisting of multiple interactions be held between a client endpoint and and one particular server in a server pool.

D) Fully distributed name space --- For achieving a high degree of fault tolerance and operation efficiency, the ENRP daemons which provide name translation service and the name space data are distributed across the operational scope of the network.

   ENRP daemon servers can be added to or removed from the operation scope dynamically, without interrupting the current name translation service.

   For example, a node may be originally configured to operate without a local ENRP server. When the load condition changes, one can start a new ENRP server on that node to increase the operation capacity. The new ENRP server will automatically integrate itself with the existing ENRP server(s) in the scope.

   Similarly, when an ENRP server becomes unavailable for service (e.g., being intentionally shutdown, or suffered failure), its ASAP clients will be automatically taken-over by a remote ENRP server and continuously have ENRP services provided.

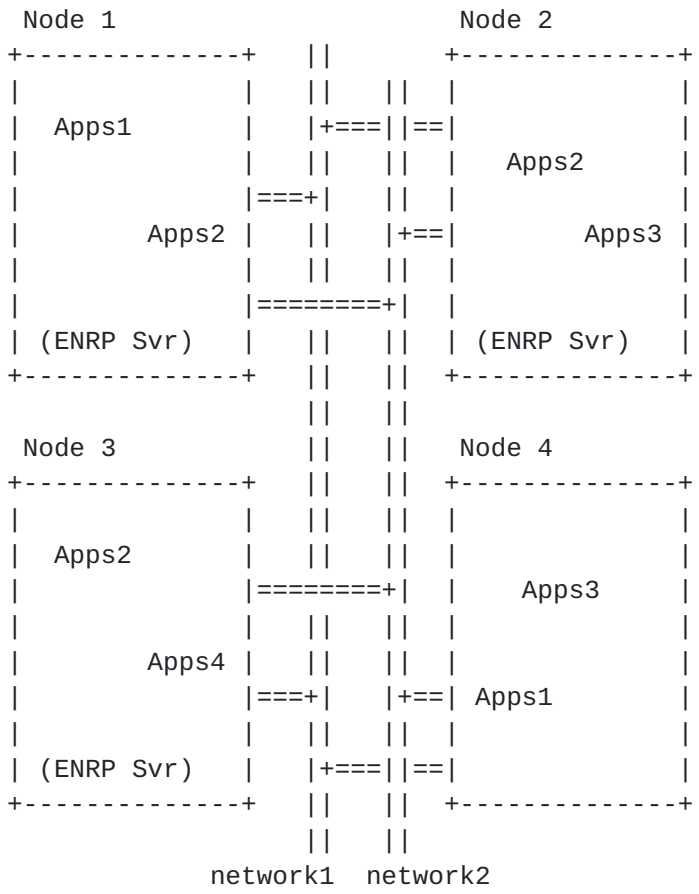
   Note: Details on ENRP can be found in [ENRP].

E) Network failure detection and automatic recovery --- In the case when a major network failure breaks the operation scope into isolated communication islands, the name translation service will survive and continue inside each island so long as there is one or more ENRP servers present in that island. Endpoints inside each island will still continue to be able to communicate with each other. Moreover, when the network recovers, the isolated ENRP servers will

re-discover each other and re-integrate the name space back into its original form.

Figure 2. shows an example of distributed applications operating in a scope that is connected by a pair of redundant networks.

Figure 2.



In this example, there are four nodes in the scope, as shown in Figure 2. **On Node 1, Node 2, and Node 3, there is an ENRP server running. On** each of the nodes, there are also some applications running. Each application has a registered name in the name space collectively managed by the three ENRP servers. In the example, the registered names are "Apps1", "Apps2", "Apps3", and "Apps4". Some of the applications (Apps1, Apps2, and Apps3) are distributed as server pools.

When sending messages to each other, the sender application simply addresses the recipient by its registered name. The ASAP layer inside the sender application will query its home ENRP server to obtain the transport address(es) and load control information of the recipient before sending the message out.

Also note in the example, there is no ENRP server on Node 4. But the applications on Node 4 will be served by one of the ENRP servers on other nodes.

1.4 Organization of this document

In Chapter 2 we give the details of the ASAP interface, focusing on the communication primitives between the applications above ASAP and ASAP itself, and the communications primitives between ASAP and SCTP (or other transport layer). Also included in this discussion is relevant timers and configurable parameters as appropriate. Chapter 3 details ASAP message formats. Chapter 4 provides settable protocol values.

1.5 Scope of ASAP

The requirements for high availability and scalability do not imply requirements on shared state and data. ASAP does not provide transaction failover. If a host fails during processing of a transaction this transaction may be lost. Some services may provide a way to handle the failure, but this is not guaranteed. ASAP MAY provide hooks to assist an application in building a mechanism to share state but ASAP in itself will NOT share any state.

1.5.1 Extent of the name space

The scope of the ASAP/ENRP is NOT Internet wide. The namespace is neither hierarchical nor arbitrarily large like DNS. We propose a flat peer-to-peer model. Pools of servers will exist in different administrative domains. For example, suppose I want to use ASAP/ENRP. First, the PU will use DNS to contact an ENRP server. Suppose a PU in North America and wish to contact the server pool in Japan instead of North America. The PU would use DNS to get the IP address of the Japanese server pool domain, that is, the address of an ENRP server('s) in Japan.

2. The ASAP Interfaces

This chapter will focus primarily on the primitives and notifications that form the interface between the ASAP-user and the ASAP and that between ASAP and its lower layer transport protocol (e.g., SCTP).

Appropriate timers and recovery actions for failure detection and management are also discussed.

2.1 ASAP User <-> ASAP Layer: Primitives and Notifications

A ASAP user passes primitives to the ASAP sub-layer to request certain actions. Upon the completion of those actions or upon the detection of

certain events, the ASAP will notify the ASAP-user.

2.1.1 Registration.Request Primitive

Format: registration.request(endpointName)

where the endpointName parameter contains a NULL terminated ASCII string of fixed length.

The ASAP user invokes this primitive to add itself to the name space. The ASAP user must register its name with the ENRP server by using this primitive before other ASAP endpoints in the name space can send message to this ASAP user by name or by handle (see Sections 2.1.5.1? and 2.1.5.2??).

In response to the registration primitive, the ASAP layer will send a REGISTRATION message to the home ENRP server (See section 3.?), and start a T2-registration timer.

If the T2-registration timer expires before receiving a REGISTRATION_RESPONSE message, or a SEND.FAILURE notification is received from the SCTP layer, the ASAP layer shall start the Server Hunt procedure (see Section 2.5.4??) in an attempt to get service from a remote ENRP server.

2.1.2 Deregistration.Request Primitive

Format: deregistration.request()

The ASAP user invokes this primitive to remove itself from the name space. This should be used as a part of the graceful shutdown process by the application.

A DEREGISTRATION message will be sent by ASAP layer to the home ENRP server (see Section 3.2?).

2.1.3 Cache.Populate.Request Primitive

Format: cache.populate.request(destinationAddress, typeOfAddress)

If the address type is a name and local name translation cache exists, the ASAP layer should initiate a mapping information query on the name and update it local cache when the response comes back from the ENRP server.

2.1.4 Cache.Purge.Request Primitive

Format: cache.purge.request(destinationAddress, typeOfAddress)

If the address type is a name and local name translation cache exists, the ASAP layer should remove the mapping information on the name from its local cache.

2.1.5 Data.Send.Request Primitive

Format: data.send(destinationAddress, typeOfAddress, message,
sizeOfMessage, Options);

This primitive requests ASAP to send a message to some specified receiver within the name space.

Depending on the address type used for the send request, the sender's ASAP layer may perform address translation and receiver selection before sending the message out.

The data.send primitive can take the following forms of address type:

2.1.5.1 Send by Name

In this case the destinationAddress and typeOfAddress together indicates a name.

This is the simplest form of data.request primitive. By default, this directs ASAP to send the message to one of the endpoints in the named group.

Before sending the message out to the named group, the sender's ASAP layer must first perform a name to address translation. It may also need to perform receiver selection if multiple endpoints exist in the group.

If the sender's ASAP implementation does not support local cache of the translation information or if it does not have the translation information on that named group in its local cache, it will transmit a request for the name mapping information to the ENRP server, and hold the outbound message in queue while waiting for the response from the ENRP server (any further send request to this named group before the ENRP server responds should also be queued).

Once the necessary mapping information arrives from the ENRP server, the sender's ASAP will:

A) map the name into a list of transport addresses of the destination endpoint,

B) If multiple endpoints exist in that named group, ASAP will choose one of them and transmit the message to it. In that case, the

choice of the receiver is made by ASAP layer of the sender based on the server pooling policy as discussed in section 2.????.

C) if no association exists towards the destination, establish a new SCTP association,

NOTE: if the underlying SCTP implementation supports implicit association setup, this step is not needed (see [??]).

D) send out the queued message to the SCTP association using the SEND primitive (see [???]), and,

E) if the local cache is implemented, append/update the local cache with the mapping information received in the ENRP server's response. Also, record the local SCTP association id if a new association was created.

For more on the ENRP server request procedures see [ENRP].

Optionally, the ASAP layer of the sender may return the SCTP association handle of the selected receiver endpoint to the application after sending the message. This handle can then be used dfor future transmissions to that receiver endpoint (see Section 2.??).

Section ??? defines the fail-over procedures for cases where the selected endpoint is found unreachable.

2.1.5.1.1 Receiver Endpoint Selection

Each time a ASAP user sends a message to a named group that contains more than one endpoint, the sender's ASAP layer must select one of the endpoints in the named group as the receiver of the current message. The selection is done according to the current server pooling policy of the named group to which the message is sent.

Note, no selection is needed if the ASAP_SEND_TOALL option is set (see Section 2.??).

When joining a named group, along with its registration each endpoint specifies its preferred server pooling policy for receiving messages sent to this named group. But only the server pooling policy specified by the first endpoint joining the named group will become the current server pooling policy of the group.

Moreover, together with the server pooling policy, each endpoint can also specify a Policy Value for itself at the registration time. The meaning of the policy value depends on the current server pooling policy of the group. An endpoint can also change its policy value whenever it desires, see Section 3.?? for details.

Note, if this first endpoint removes itself from the named group (e.g., by de-registration from the name space) and the remaining endpoints have specified conflicting server pooling policies at their corresponding registrations, it is implementation specific to determine the new current server pooling policy.

Four basic server pooling policies are defined in ASAP, namely the Round Robin, Least Used, Least Used Degrading and weighted round robin. The following sections describes each of these policies.

2.1.5.1.2 Round Robin Policy

When a ASAP endpoint sends messages by name and Round-Robin is the current policy of that named group, the ASAP layer of the sender will select the receiver for each outbound message by round-Robining through all the registered endpoints in that named group, in an attempt to achieve an even distribution of outbound messages. Note that in a large server pool, the ENRP deamon may NOT send back all server elements to the ASAP client. In this case the client or PU will be performing a round robin policy on a subset of the entire server pool.

2.1.5.1.4 Least Used Policy

When the destination named group is under the Least Used server pooling policy, the ASAP layer of the message sender will select the registered endpoint that has the lowest policy value in the group as the receiver of the current message. If more than one registered endpoint from the group share the same lowest policy value, the selection will be done round Robin amongst those registered endpoints.

It is important to note that this policy means that the same registered endpoint will be always selected as the message receiver by the sender until the load control information of the name is updated and changed in the local cache of the sender (see section 2.??).

2.1.5.1.5 Least Used with Degradation Policy

This policy is the same as the Least Used policy with the exception that, each time the endpoint with the lowest policy value is selected from the group as the receiver of the current message, its policy value is incremented, and thus it may no longer be the lowest value in the group.

This provides a degradation of the policy towards round Robin policy over time. As with the Least Used policy, every local cache update at the sender will bring the policy back to Least Used with Degradation.

2.1.5.2 Send by Handle

In this case the destinationAddress and typeOfAddress together indicates a ASAP endpoint handle.

This requests the ASAP layer to deliver the message to the endpoint identified by the handle.

The handle should contains the name and the primary destination transport address of the destination endpoint.

The ASAP layer shall use the address to identify the SCTP association (or to setup a new one if necessary) and then invoke the SEND primitive to send the message out.

If local cache is supported and the mapping information for the name found in the handle is not available in the local cache, the sender's ASAP layer should, after sending the message, also transmit a request for the name mapping information to the ENRP server. Once the necessary mapping information arrives from the ENRP server, the sender's ASAP will update its local cache with the newly received mapping information for that name.

Section ??? defines the fail-over procedures for cases where the endpoint pointed to by the handle is found unreachable.

Optionally, the ASAP layer may return the actual handle to which the message was sent (this may be different from the handle specified when the primitive is invoked, due to the possibility of automatic fail-over).

2.1.5.3 Send by Transport Address

In this case the destinationAddress and typeOfAddress together indicates an SCTP transport address.

This directs the sender's ASAP layer to send the message out to the specified transport address.

No endpoint fail-over is support when this form of send request is used.

2.1.5.4 Options

The Options parameter passed in the various forms of the above data.request primitive gives directions to the sender's ASAP layer on special handling of the message delivery.

Options can be grouped as follows:

- endpoint fail-over (allowed, or prohibited),
- whether to send to one endpoint or to the whole named group,
- whether to send to the same receiver endpoint last sent to within the named group, and
- options passed to the SCTP transport protocol.

The complete list of Options is as follows:

ASAP_USE_DEFAULT: 0x0000

Use default setting.

ASAP_SEND_FAILOVER: 0x0001

Enables endpoint fail-over on this message. In case where the first selected receiver endpoint or the endpoint pointed to by the handle is found unreachable, this option allows the sender's ASAP layer to re-select an alternate receiver endpoint from the same named group if one exists, and silently re-send the message to this newly selected endpoint.

Endpoint unreachable is normally indicated by the Communication Lost or Send Failure notification from SCTP.

ASAP_SEND_NO_FAILOVER: 0x0002

This option prohibits the sender's ASAP layer from re-sending the message to any alternate receiver endpoint in case that the first selected receiver endpoint or the endpoint pointed to by the handle is found unreachable. Instead, the sender's ASAP layer shall notify its upper layer about the unreachability with an Error.Indication and any unsent data.

ASAP_SEND_TO_LAST: 0x0004

This option requests the sender's ASAP layer to send the message to the same receiver endpoint in the named group that the previous message was sent to.

ASAP_SEND_TOALL: 0x0008

When sending by name, this option directs the sender's ASAP layer to send a copy of the message to all the endpoints, except for the sender itself, in that named group.

ASAP_SEND_TOSELF: 0x0010.

This option only applies in combination with ASAP_SEND_TOALL option. It permits the sender's ASAP layer also deliver a copy of the message to itself (i.e., loopback).

ASAP_SCTP_BUNDLE: 0x0100

This option allows the local SCTP transport layer to bundle the outbound messages whenever possible into bigger datagrams before transmitting them onto the network.

ASAP_SCTP_NO_BUNDLE: 0x0200

This option disallows the local SCTP transport layer to bundle outbound messages.

ASAP_SCTP_HB_ON: 0x0400

This option instructs the local SCTP transport layer to turn on heartbeat on the SCTP association indicated by the destinationAddress parameter.

ASAP_SCTP_HB_OFF: 0x0800

This option instructs the local SCTP transport layer to turn off heartbeat on the SCTP association indicated by the destinationAddress parameter.

ASAP_SCTP_UNORDER: 0x1000

This option instructs the SCTP transport layer to send the current message using un-ordered delivery.

2.1.6 Data.Received Notification

Format: data.received(messageReceived, sizeOfMessage, senderAddress, typeOfAddress)

When a new user message is received, the ASAP layer of the receiver uses this notification to pass the message to its upper layer.

Along with the message being passed, the ASAP layer of the receiver should also indicate to its upper layer the message sender's address. The sender's address can be in the form of either an SCTP association id, or a ASAP endpoint handle.

A) If the name translation local cache is implemented at the receiver's ASAP layer, a reverse mapping from the sender's IP address to the endpoint name should be performed and if the mapping is successful, the sender's ASAP endpoint handle should be constructed and passed in the senderAddress field.

B) If there is no local cache or the reverse mapping is not successful, the SCTP association handle should be passed in the senderAddress field.

2.1.7 Error.Report Notification

Format: error.report(destinationAddress, typeOfAddress,
failedMessage, sizeOfMessage)

An error.report should be generated to notify the ASAP user about failed message delivery as well as other abnormalities (see Section 2.2.3 for details).

The destinationAddress and typeOfAddress together indicates to whom the message was originally sent. The address type can be either a ASAP endpoint handle, association id, or a transport address.

The original message (or the first portion of it if the message is too big) and its size should be passed in the failedMessage and sizeOfMessage fields, respectively.

2.2 ASAP Layer <-> SCTP: Primitives and Notifications

This section gives a brief description on some SCTP notifications and primitives that the ASAP layer uses. See Section 9 in [??] for more information on SCTP primitives and notifications.

2.2.1 SCTP SEND Primitive

Basic Format:

```
SEND(association id, buffer address, byte count, options)
-> result
```

The outbound message will be held in the buffer when this primitive is invoked. The ASAP layer shall identify the SCTP association which connects to the intended destination and fill in the 'association id'.

The options field will hold the options destined to the SCTP transport layer (see 2.??).

The returned 'result' can indicate whether there is any local error executing the primitive.

2.2.2 SCTP RECEIVE Primitive

Basic Format:

```
RECEIVE(association id, buffer address, buffer size)
-> byte count
```

This primitive reads the first user message out from SCTP in-queue if there is one available, and put the it into the specified buffer. The

size of the message read, in octets, will also be returned.

2.2.3 SCTP SET.PRIMARY Primitive

Basic Format:

```
SET.PRIMARY(association id, destination transport address)
-> result
```

This can be used to instructs the SCTP to use the specified destination transport address as the new primary destination address for sending messages.

2.2.4 SCTP DATA.ARRIVE Notification

SCTP layer invokes this notification when a user message is successfully received and ready for retrieval. This shall prompt the ASAP layer to invoke the RECEIVE primitive to get the data (see 2.2.2??).

2.2.5 SCTP SEND.FAILURE Notification

If a message can not be delivered to the specified association id for any reason, SCTP will invoke this notification to notify ASAP.

In response, the ASAP shall take the following steps:

A) If the message was originally sent by name or handle and with option ASAP_SEND_FAILOVER set, retransmit the message to an alternate endpoint of the same name if one exists in the named group. The proper server pooling policy shall be followed if more than one alternates exist in the group.

B) If no alternate exists or option ASAP_SEND_FAILOVER is not set when the message was originally sent, generate an Error.report to report the failure to the ASAP user.

2.2.6 SCTP COMMUNICATION.LOST Notification

When SCTP loses communication to an endpoint completely or detects that the remote endpoint has performed an abort or graceful shutdown operation, it invokes this notification to notify ASAP layer.

When handling this notification ASAP shall report this event to its ENRP server via an ENDPOINT_UNREACHABLE message with the severity level set to NORMAL_REPORT (see 3.??).

If local mapping cache is implemented, the ASAP layer should also mark

the endpoint as unreachable in its local cache. And if all the endpoints in that named group are marked as unreachable, the ASAP layer should remove the named group from its local cache.

2.2.7 SCTP NETWORK.STATUS.CHANGE Notification

The SCTP sends this notification to the ASAP layer when the reachability status of a transport address of a specific SCTP association has changed.

If the local mapping cache is supported, the ASAP layer, upon reception of this notification, should look up the information of this endpoint in its local cache and record the reachability change.

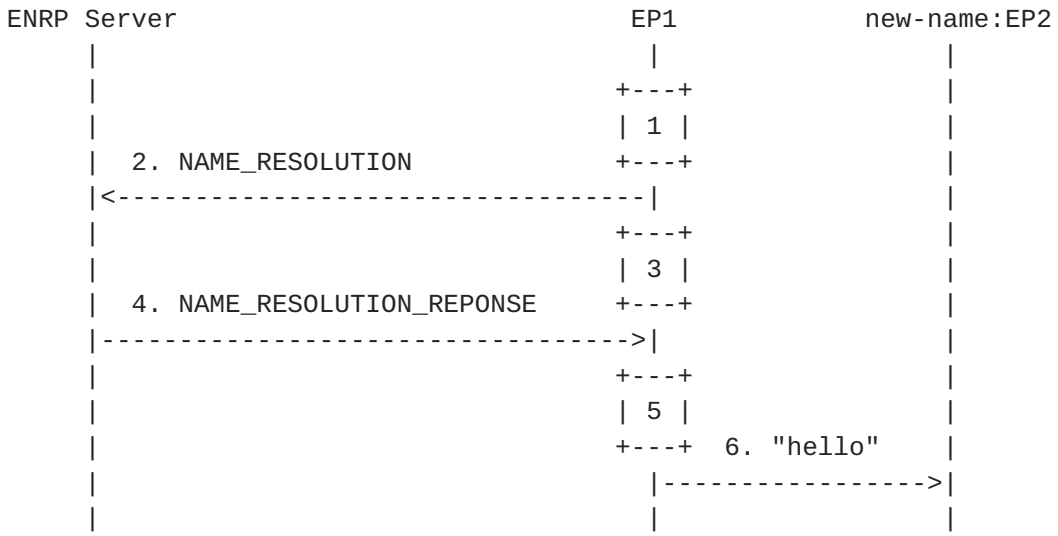
If the address in question becomes unreachable and is the primary address of the association, the ASAP layer MAY also elect a new primary for this association by invoking the SET.PRIMARY primitive (Section 2.2.3??).

If the local cache is not support or the reverse look up does not succeed, ASAP takes no action.

2.4 Examples

2.4.1 Send to an Unknown Name

This example shows the event sequence when the user of ASAP endpoint 1 (EP1) sends message "hello world" to a name which is not in the local mapping cache (assuming local caching is supported).



1) The user at EP1 invokes:

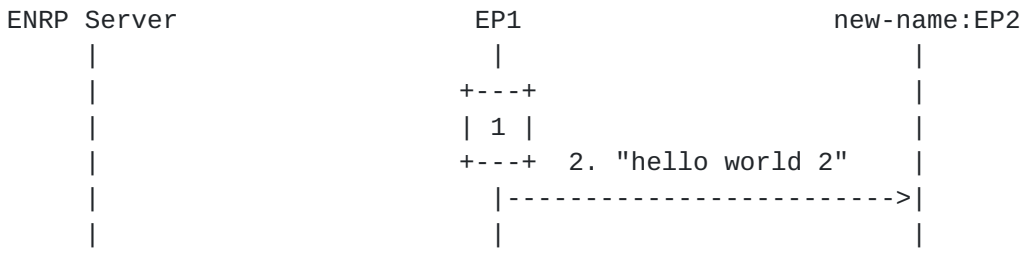
data.send("new-name", name-type, "hello", 5 0);

The ASAP layer, in response, looks up the name "new-name" in its local cache but fails to find it.

2) The ASAP layer of EP1 queues the message, and sends a NAME_RESOLUTION request to the ENRP server asking for all information about "new-name".
3) A T1-ENRPrequest timer is started while the ASAP layer is waiting for the response from the ENRP server.
4) The ENRP Server responds to the query with a NAME_RESOLUTION_REPONSE message that contains all the information about "new-name".
5) ASAP at EP1 cancels the T1-ENRPrequest timer and populate its local cache with information on "new-name".
6) Based on the server pooling policy of "new-name", ASAP at EP1 selects the receiver (EP2), sets up, if necessary, an SCTP association towards EP2 (explicitly or implicitly), and send out "hello" message.

2.4.2 Send to a Cached Name

This shows the event sequence when the ASAP user at EP1 sends another message to the "new-name".



1) The user at EP1 invokes:

data.request("new-name", name-type, "hello world 2", 13, 0);

The ASAP layer, in response, looks up the name "new-name" in its local cache and find the mapping information.

2) Based on the server pooling policy of "new-name", ASAP at EP1 selects the receiver (assume EP2 is selected again), and send out "hello world 2" message (assume the SCTP association is already set up).

2.5 Handle ASAP to ENRP Communication Failures

Three types of failure may occur when the ASAP layer at an endpoint tries to communicate with the ENRP server:

A) SCTP send failure
B) T1-ENRPrequest timer expiration
C) Registration failure

Registration failure is discussed in section 2.1.1??.

2.5.1 SCTP Send Failure

This indicates that the SCTP layer failed to deliver a message sent to the ENRP server. In other words, the ENRP server is currently unreachable.

In such a case, the ASAP layer should not re-send the failed message. Instead, it should discard the failed message and start the ENRP server hunt procedure as described in Section 3.1.2.3??.

2.5.2 T1-ENRPrequest Timer Expiration

When a T1-ENRPrequest timer expires, the ASAP should re-send the original request to the ENRP server and re-start the T1-ENRPrequest timer. In parallel, a SERVER_HUNT message should be issued per Section 3.1.2.3??.

This should be repeated up to 'max-request-retransmit' times. After that, an Error.Report notification should be generated to inform the ASAP user and the ENRP request message associated with the timer should be discarded.

2.5.3 Handle ENDPOINT_KEEP_ALIVE Messages

At times, a ASAP endpoint may receive ENDPOINT_KEEP_ALIVE messages (see section 3.1.2.1?) from the ENRP server. This message requires no response and should be silently discarded by the ASAP layer. However, each time when an ENDPOINT_KEEP_ALIVE is received, the endpoint should update its home ENRP server to the sender of the latest Keep Alive message.

3. Message Summary

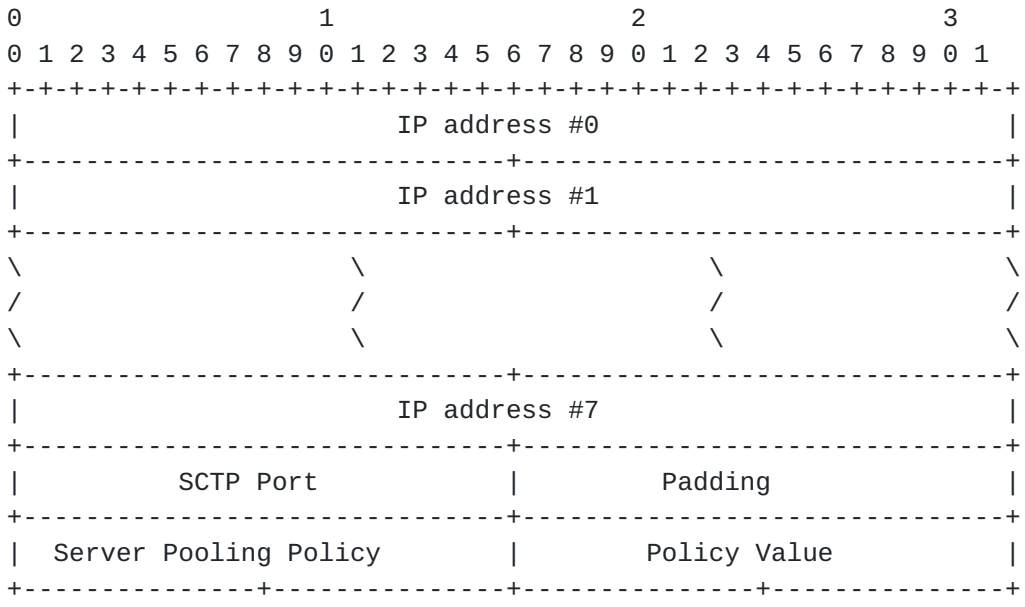
All messages as well as their fields described below shall be in Network Byte Order during transmission. For fields with a length bigger than 4 octets, a number in a pair of parentheses may follow the filed name to indicate the length of the field in number of octets.

3.1 Endpoint Entry

This field is used to represent a ASAP endpoint and the associated

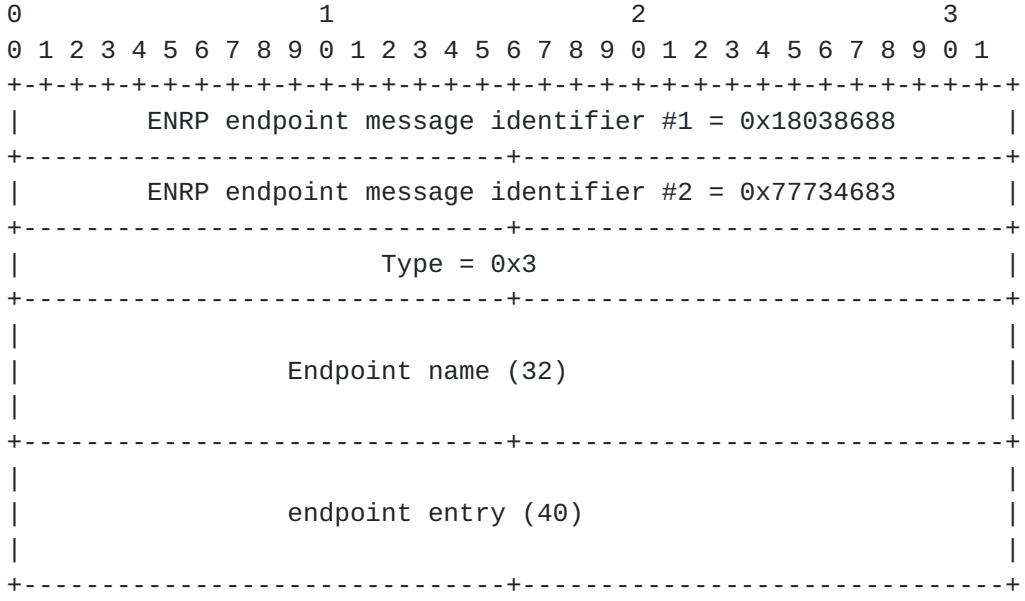
information, such as its transport address(es), load control, and other operational status information.

The field is defined to support endpoint with up to 8 different transport addresses.



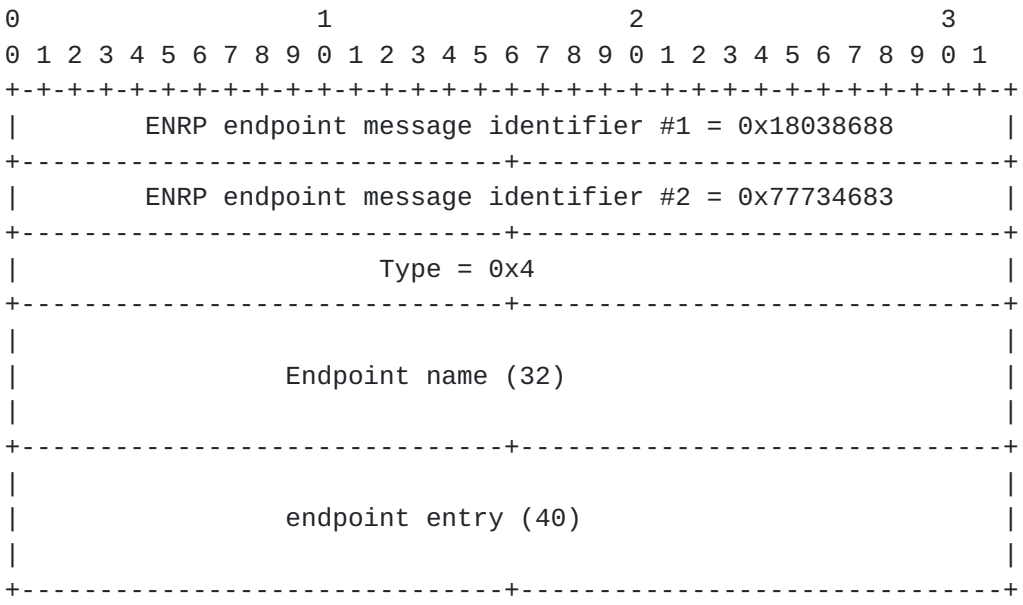
The size of the endpoint entry is 40 octets.

3.2 REGISTRATION message



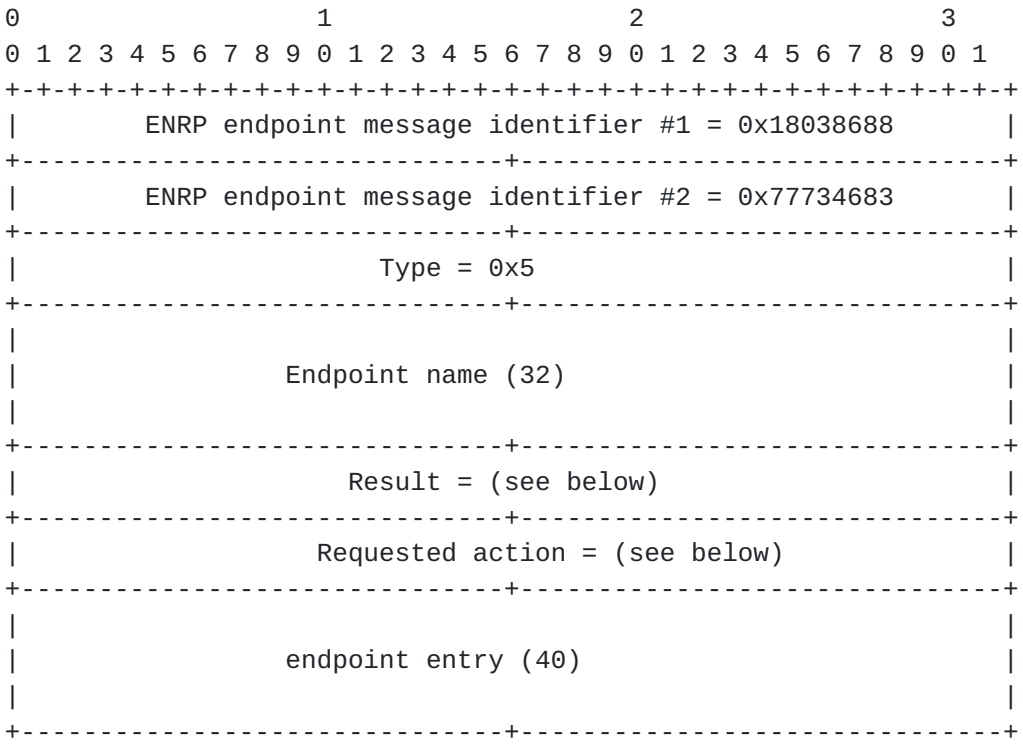
The endpoint name field specifies the name to be registered, that shall be composed of up to 32 characters. The endpoint entry field shall be filled in by the registrant endpoint to declare its transport addresses, server pooling policy and value, and other operation preferences.

3.3 DEREGISTRATION message



The endpoint sending the DEREGISTRATION shall fill in the name and the endpoint entry in order to allow the ENRP server to verify the identity of the endpoint.

3.4 REGISTRATION_RESPONSE message



In response to a REGISTRATION, the 'Requested action' field shall be set to 0x0, and the 'Result' field shall take the following values:

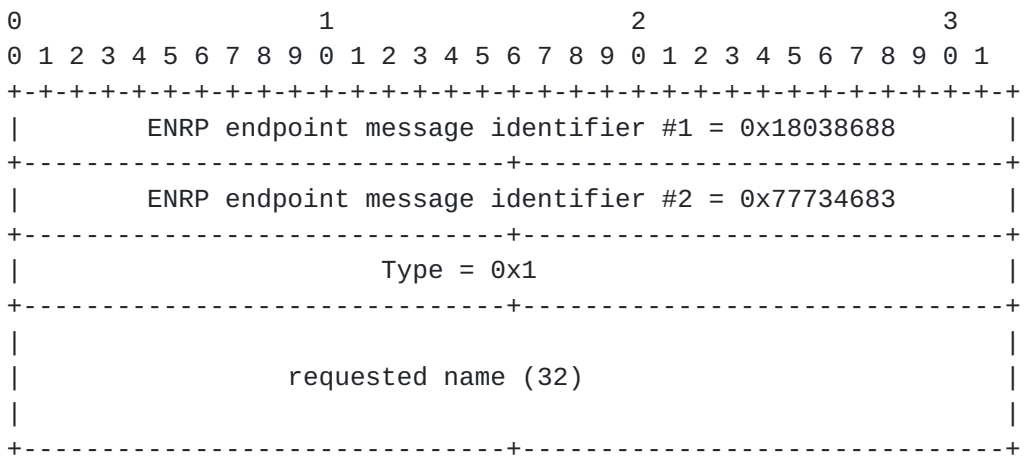
  0x0 -- registration granted
  0x1 -- registration rejected

In response to a DEREGISTRATION, the 'Requested action' field shall be set to 0x1, and the 'Result' field shall take the following values:

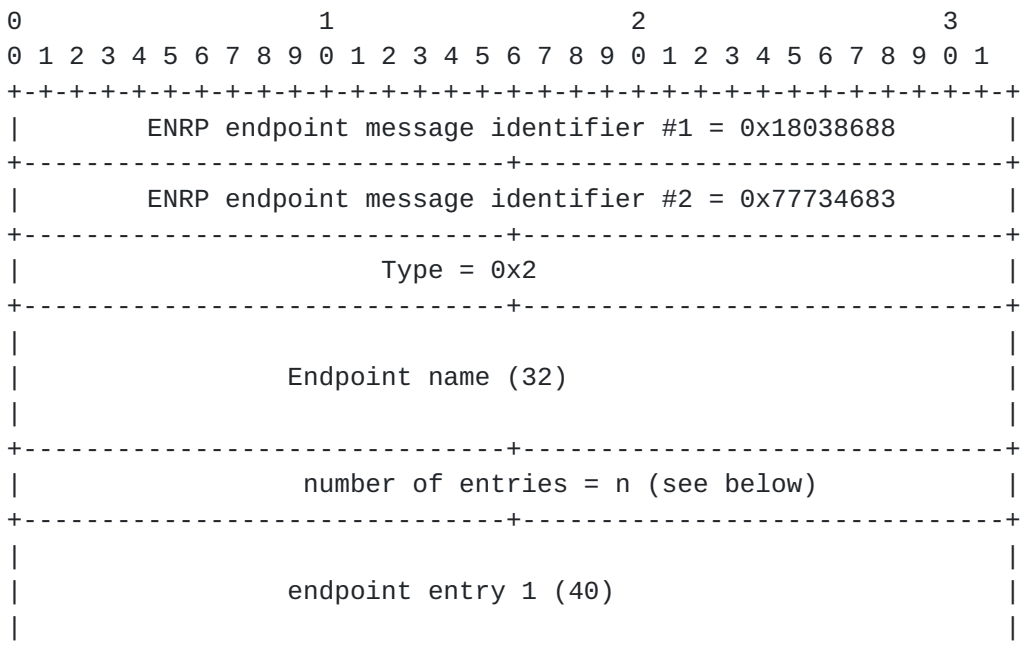
  0x2 -- de-registration granted
  0x3 -- de-registration rejected: endpoint not found
  0x4 -- de-registration rejected: other failures.

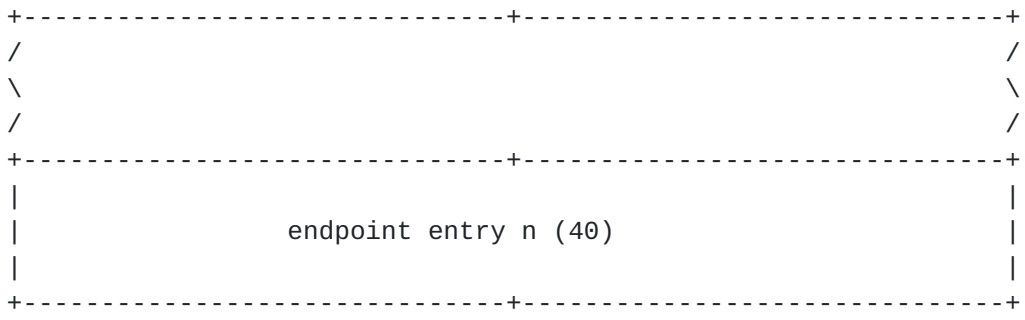
endpoint entry shall be filled in for verification purposes.

3.5 NAME_RESOLUTION message

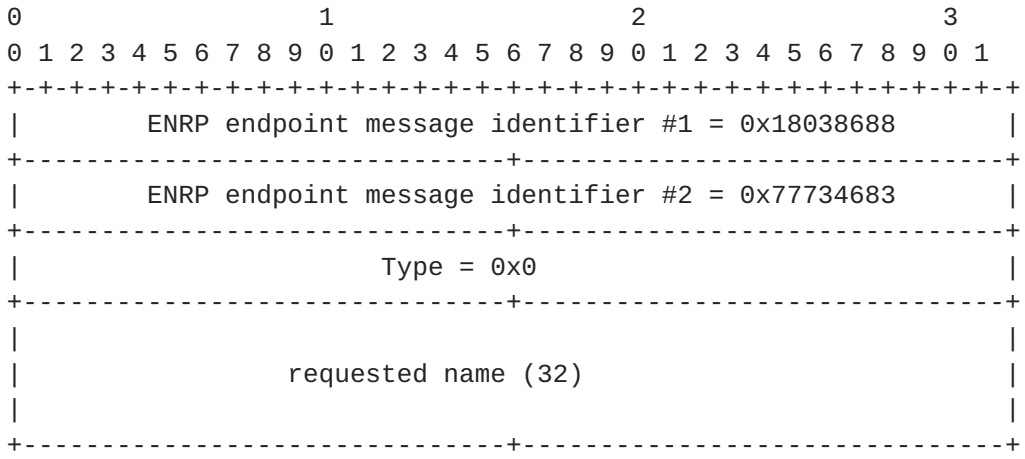


3.6 NAME_RESOLUTION_RESPONSE message

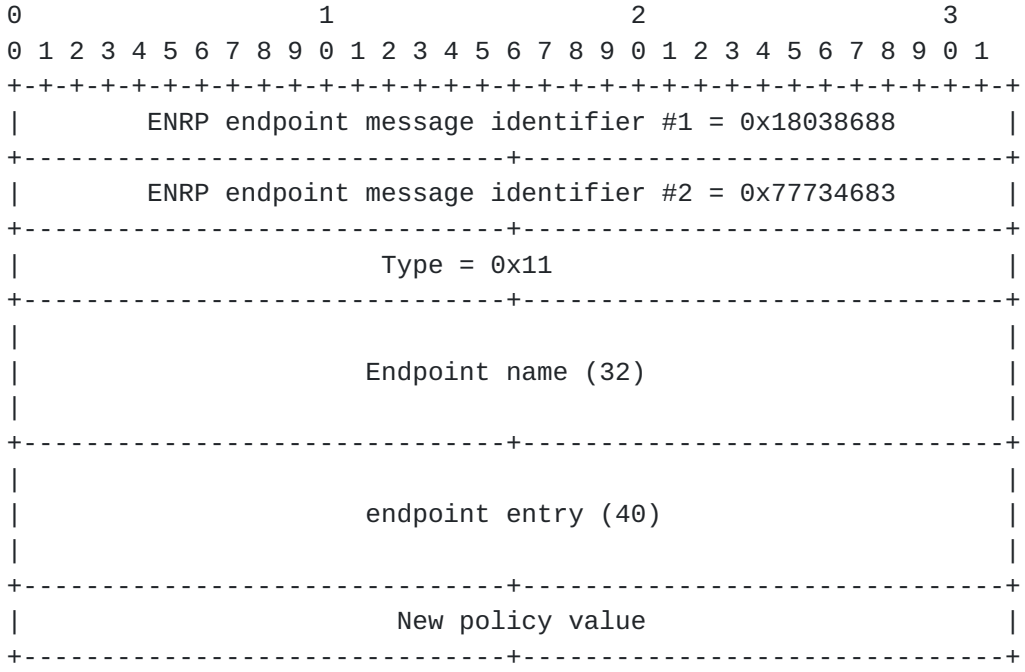




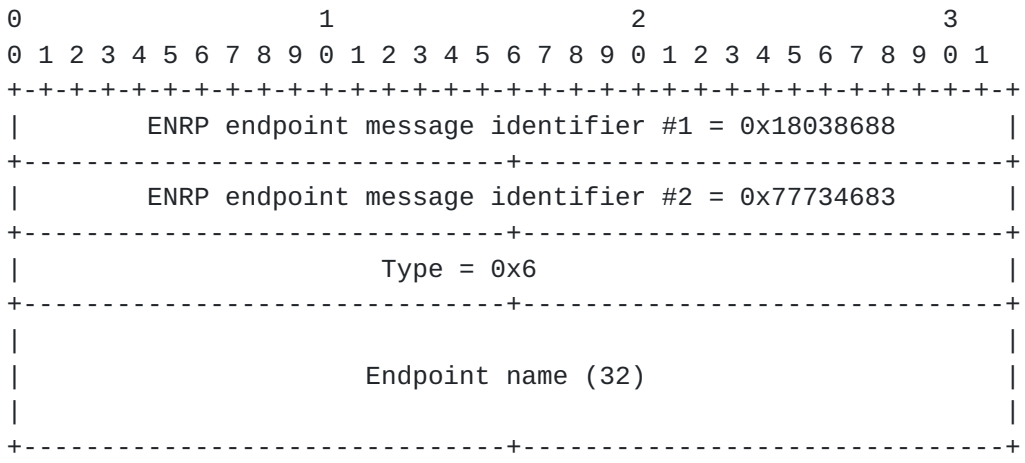
3.7 NAME_UNKNOWN message



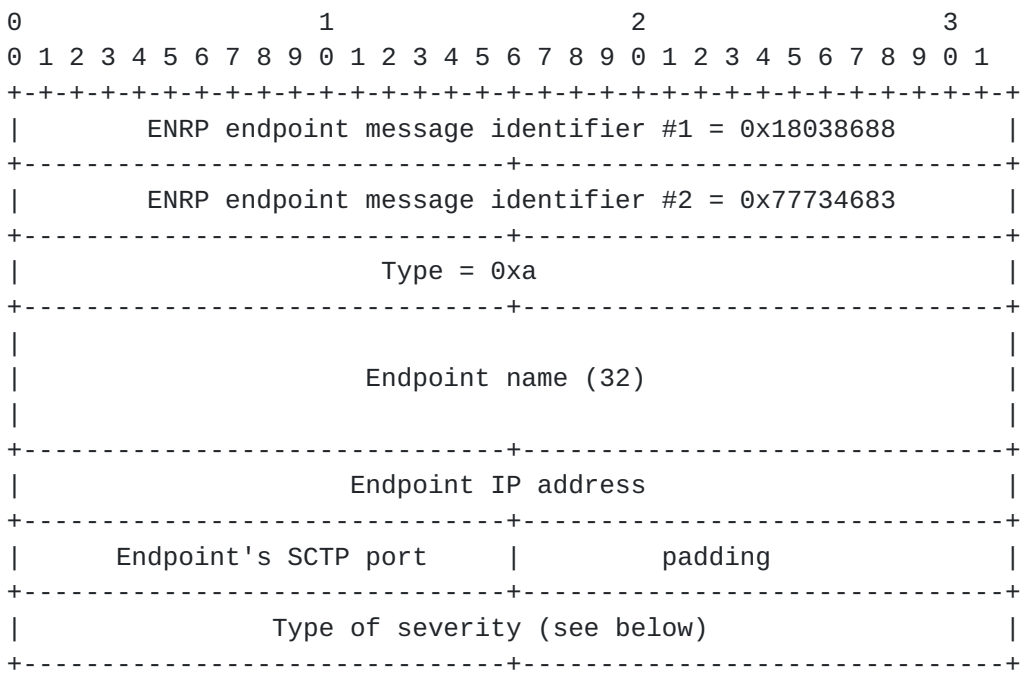
3.8 UPDATE_POLICY_VALUE message



3.9 ENDPOINT_KEEP_ALIVE message



3.10 ENDPOINT_UNREACHABLE message

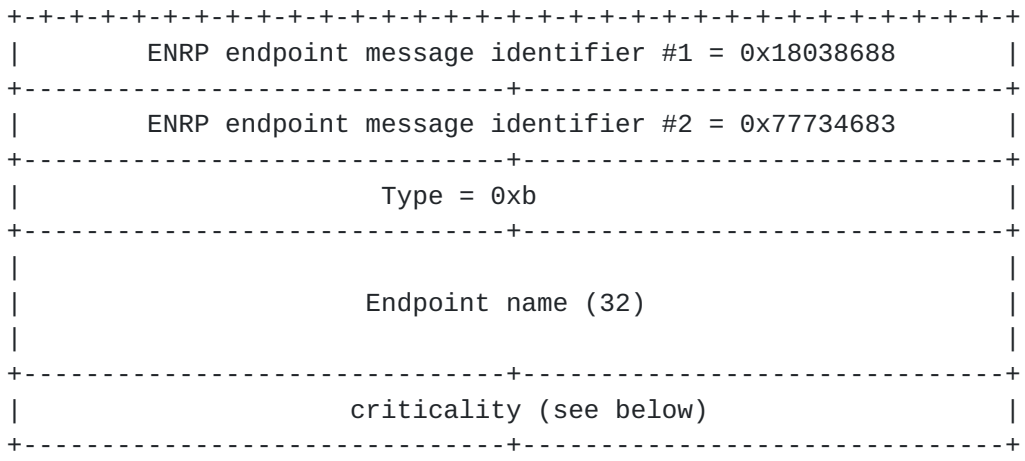


The 'Endpoint name' is not required to be filled in for this message, however the IP address and SCTP port number of the endpoint must be supplied in the message.

'Type of severity' shall take one of the following values:

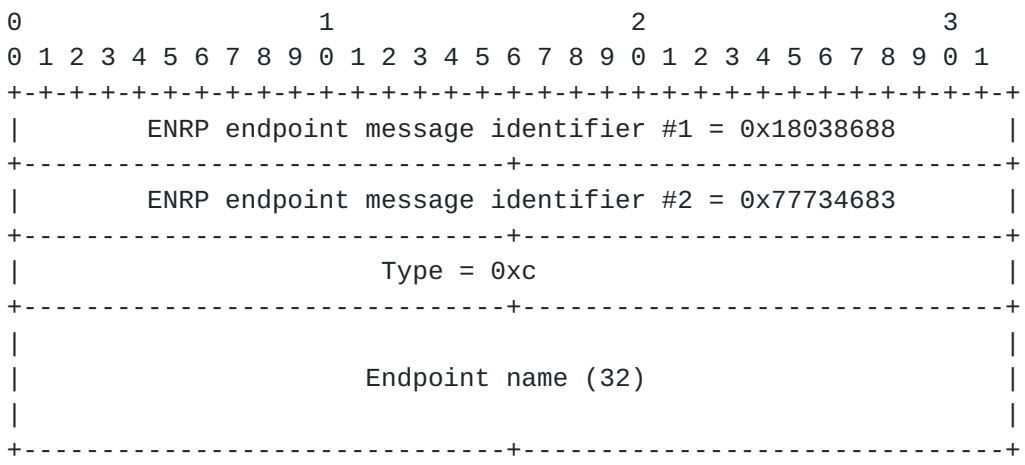
0x0 --- NORMAL_REPORT: warning to the server.
0x1 --- FINAL_REPORT: the specified endpoint must be removed by the server immediately.

3.11 SERVER_HUNT message



The 'criticality' field shall take one of the following values:
  0x1 --- LOW_CRITICALITY
  0x2 --- MED_CRITICALITY
  0x3 --- HIGH_CRITICALITY

3.12 SERVER_HUNT_RESPONSE message



4. Variables, Time Values, and Thresholds

The following is a summary of the variables, time values, and pre-set thresholds used in ASAP and ENRP protocol.

4.1 Variables

Endpoint-report-failures --- per endpoint; keeps the number of endpoint-unreachable reports concerning this endpoint.

Endpoint-sanity-failures --- per endpoint; keeps the number of failed sanity message send attempts concerning this endpoint.

Peer-server-last-heard --- per peer server; a time stamp on when the

last message was received from this peer server.

4.2 Time values

Endpoint-sanity-cycle --- the period for a home ENRP server to start a new round of endpoint sanity check.

Peer-heartbeat-cycle ---the period for an ENRP server to send out a heart heat message.

T1-ENRPrequest - A timer started when a request is sent by ASAP to the ENRP server (providing application information is queued). Normally set to 15 seconds.

T2-registration - A timer started when sending a registration request to the local ENRP server, normally set to 30 seconds.

T3-registration-reattempt - If the registration cycle does not complete this timer is begun to restart the registration process. Normal value for this timer is 10 minutes.

T4-reregistration - This timer is started after successful registration into the ASAP name space and is used to cause a re-registration at a periodic interval. This timer is normally set to 10 minutes.

4.3 Thresholds

Max-endpoint-sanity-failures --- pre-set threshold for Endpoint-sanity-failures.

Max-endpoint-report-failures --- pre-set threshold for Endpoint-report-failures.

Max-time-last-heard --- pre-set threshold for Peer-last-heard.

Max-time-no-response --- pre-set threshold for a peer server to answer a PEER_PRESENCE message with reply required.

Timeout-registration --- pre-set threshold; how long an endpoint will wait for the REGISTRATION_RESPONSE from its home ENRP server.

Timeout-server-hunt --- pre-set threshold; how long an endpoint will wait for the REGISTRATION_RESPONSE from its home ENRP server.

num-of-serverhunts - The current count of server hunt messages that have been transmitted.

registration-count - The current count of attempted registrations.

max-reg-attempt - The maximum number of registration attempts to be

made before a server hunt is issued.

max-request-retransmit - The maximum number of attempts to be made when requesting information from the local ENRP server before a server hunt is issued.

5. References

[SCTP] R. R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. J. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and, V. Paxson, "Stream Control Transmission Protocol," <RFC 2960>, October 2000.

[ENRP] Q. Xie, R. R. Stewart "Endpoint Name Resolution Protocol", draft-xie-rserpool-enrp-00.txt, work in progress.

6. Acknowledgements

The authors wish to thank John Loughney, Lyndon Ong, and Maureen Stillman and many others for their invaluable comments.

7. Authors' Addresses

Randall R. Stewart
24 Burning Bush Trail.
Crystal Lake, IL 60012
USA

Phone: +1-815-477-2127
EMail: rrs@cisco.com

Qiaobing Xie
Motorola, Inc.
1501 W. Shure Drive, #2309
Arlington Heights, IL 60004
USA

Phone: +1-847-632-3028
EMail: qxie1@email.mot.com