Network Working Group
INTERNET DRAFT

M. Tuexen
Siemens AG
Q. Xie
Motorola
R. Stewart
E. Lear
Cisco
L. Ong
Nortel Networks
M. Stillman
Nokia

expires May 24, 2001

November 24, 2000

Requirements for Reliable Server Pooling

<draft-tuexen-rserpool-reqts-01.txt>

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of [RFC2026].

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.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet Drafts as reference material or to cite them other than as "work in progress."

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

The goal is to develop an architecture and protocols for the management and operation of server pools supporting highly reliable applications, and for client access mechanisms to a server pool.

This document defines requirements and architecture for management and access to server pools, including requirements from a variety of applications, building blocks and interfaces, different styles of

pooling, security requirements and performance requirements such as failover times and coping with heterogeneous latencies.

Important requirements of this architecture are

- network fault tolerance,
- highly available services,
- resistance against malicious attacks,
- and scalability.

1. Introduction

1.1. Overview

This document defines the reliable server pooling architecture and the requirements for the protocols used. Reliable server pools can be used for providing high available services by using a set of servers in a pool.

Real time applications must also be supported which leads to requirements on the processing time needed. Scalability is another important requirement.

Given that the server pool can be attacked by hackers, if one or more of the servers are hijacked then the server pool is compromised. Therefore, the security requirement is to catalog the threats to the reliable server pool and identify appropriate responses to those threats.

1.2. Terminology

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

Pool:
: A collection of clients or servers providing the same service.

Pool Element:
: A client or server which belongs to a pool.

Pool User:
: A client which gets served by a pool element.

1.3. Abbreviations

ASAP: Aggregate Server Access Protocol

ENRP: Endpoint Name Resolution Protocol

PE: Pool element

PU: Pool user

SCTP: Stream Control Transmission Protocol

TCP: Transmission Control Protocol

2. Reliable Server Pooling Architecture

In this section, we discuss what a typical reliable server pool architecture may look like.

2.1. Common Rserpool Functional Areas

The following functional areas or components may likely be present in a typical Rserpool system architecture:

- A number of logical "Server Pools" to provide distinct application services.

  Each of those server pools will likely be composed of some number of "Pool Elements (PEs)" - which are application programs running on distributed host machines, collectively providing the desired application services via, for example, data sharing and/or load sharing.

  Each server pool will be identifiable in the operation scope of the system by a unique "name".

- Some "Pool Users (PUs)" which are the users of the application services provided by the various server pools.

- PUs may or may not be part of a server pool themselves, depending on whether or not they wish to be accessed by pool name by others in the operation scope of the system.

- A "Name Space" which contains all the defined names within the operation scope of the system.

- One or more "Name Servers" which carry out various maintenance functions (e.g., registration and de-registration, integrity

checking) for the "Name Space".

2.2. Rserpool protocol overview

ENRP is designed to provide a fully distributed fault-tolerant real-time translation service that maps a name to a set of transport addresses pointing to a specific group of networked communication endpoints registered under that name. ENRP employs a client-server model with which an ENRP server will respond to the name translation service requests from endpoint clients on both the local host and remote hosts.

Aggregate Server Access Protocol (ASAP) in conjunction with ENRP provides a fault tolerant 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.

The fault tolerant 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 fault tolerant name translation service.

2.3. Typical Interactions between Rserpool Components

The following drawing shows the typical Rserpool components and their possible interactions with each other:

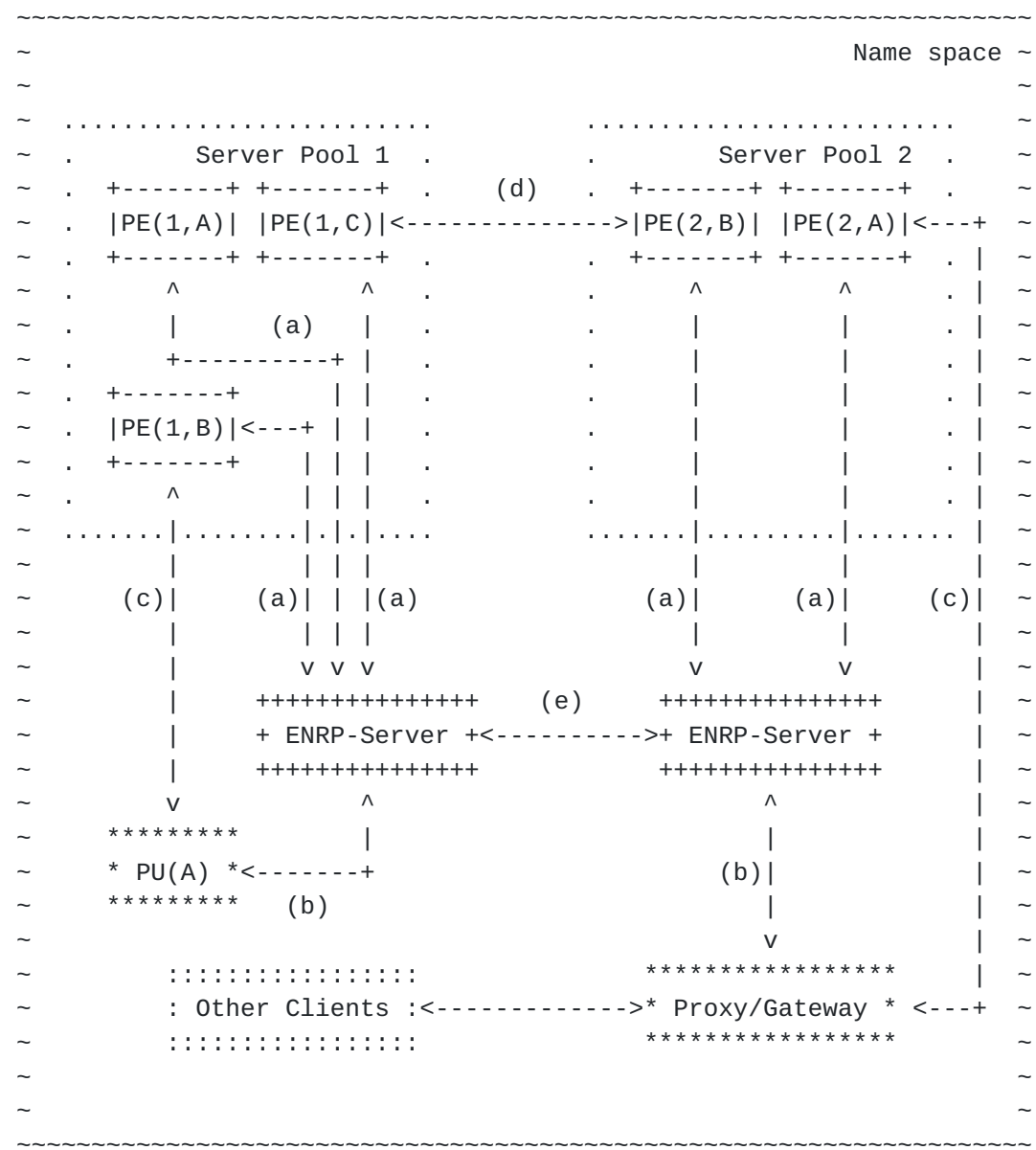


Figure 2.1 Rserpool components and their possible interactions.

Note, the "Proxy/Gateway" in Figure 2.1 is just a special kind of PU which is designed to allow non-Rserpool (e.g., legacy) clients to access services provided by Rserpool applications.

In Figure 2.1 we can identify the following possible interactions:

(a) Server Pool Elements <-> ENRP Server: (ASAP)

Each PE in a pool uses ASAP to register or de-register itself as well as to exchange other auxiliary information with the ENRP Server. The ENRP Server also uses ASAP to monitor the operational status of each PE in a pool.

(b) PU <-> ENRP Server: (ASAP)

A PU normally uses ASAP to request the ENRP Server for a name-to-address translation service before the PU can send user messages addressed to a server pool by the pool's name.

(c) PU <-> PE: (ASAP)

ASAP can be used to exchange some auxiliary information of the two parties before they engage in user data transfer.

(d) Server Pool <-> Server Pool: (ASAP)

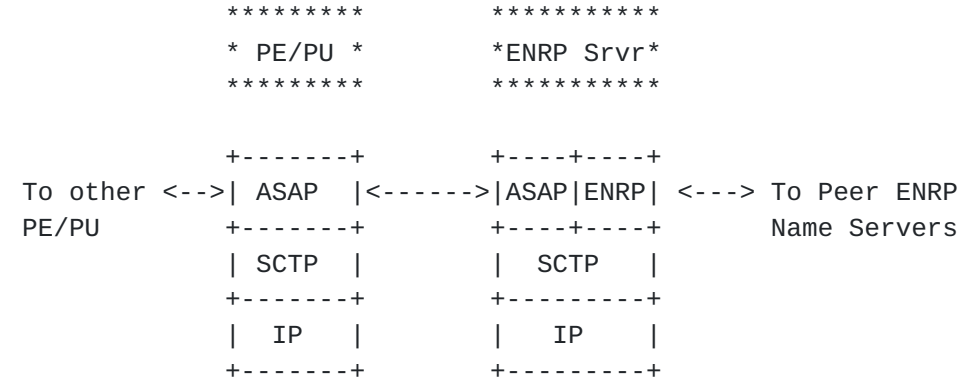
A PE in a server pool can become a PU to another pool when the PE tries to initiate communication with the other pool. In such a case, the interactions described in B) and C) above will apply.

(e) ENRP Server <-> ENRP Server: (ENRP)

ENRP can be used to fulfill various Name Space operation, administration, and maintenance (OAM) functions.

2.4. Typical Protocol Architecture for Rserpool

Some text is needed.



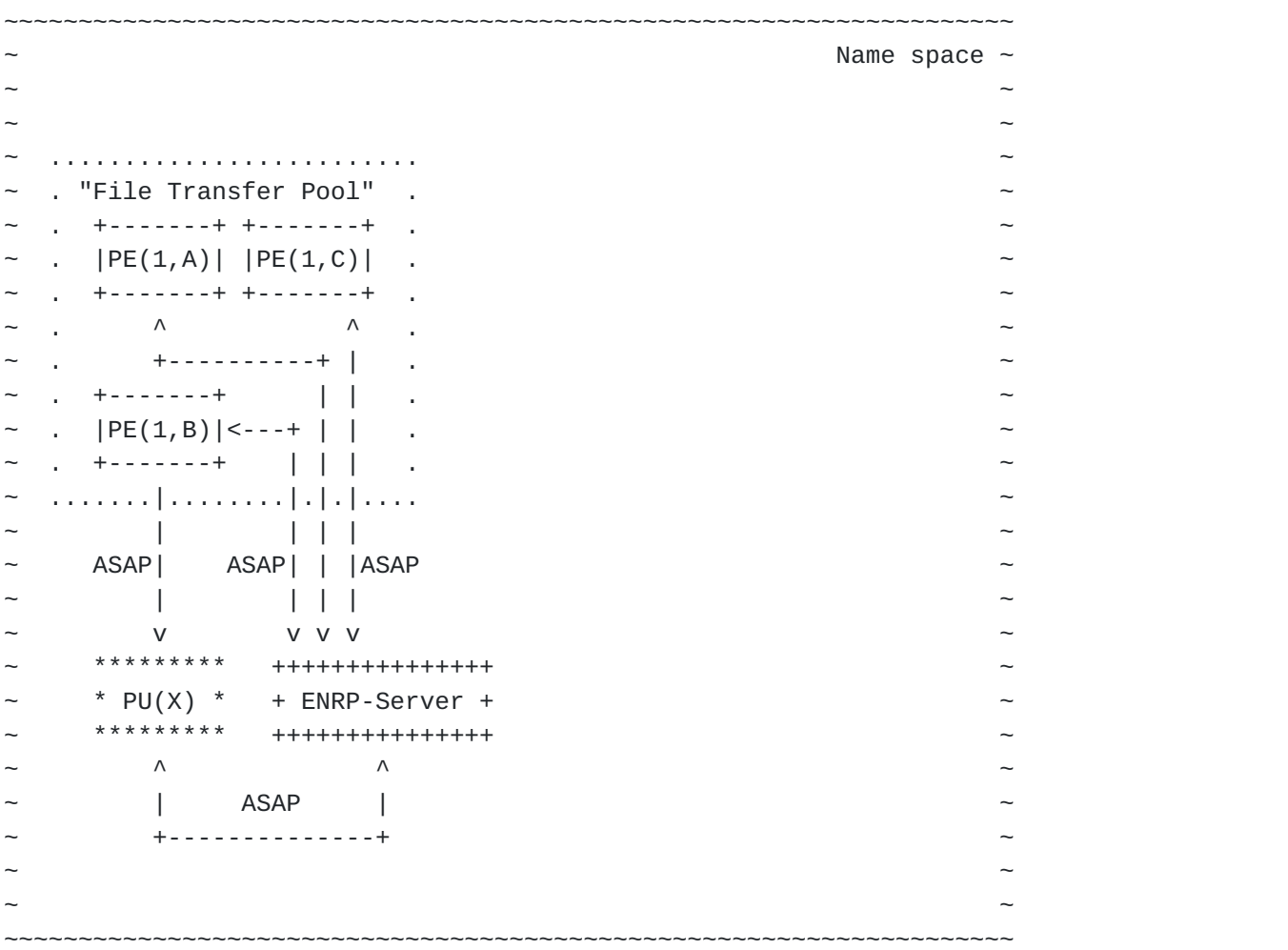
2.5. Two file Transfer examples

In this section we present two separate file transfer examples using ENRP and ASAP. We present two separate examples demonstrating an ENRP/ASAP aware client and a client that is using a Proxy or Gateway to perform the file transfer. In this example we will use a FTP [RFC959] model with some modifications. The first example (the rserpool aware one) will modify FTP concepts slightly so that the file transfer takes place over SCTP streams. In the second example we will use TCP between the unaware client and the Proxy. The Proxy itself will use the

modified FTP with SCTP as illustrated in the first example.

Please note that in the example when using SCTP it does NOT follow FTP [RFC959] precisely but we use FTP like concepts and attempt to adhere to the basic FTP model. These examples use FTP for illustrative purposes, FTP was chosen since many of the basic concept are well known and should be familiar to readers.

2.5.1. The ENRP/ASAP aware file transfer over SCTP.



To effect a file transfer the following steps would take place.

1) The application in PU(X) would send a login request to the Pool name "File Transfer Pool".

2) PU(X)'s ASAP layer would not find the name in its local cache, I.e. it encounters a "cache miss".

3) PU(X)'s ASAP layer would send an ASAP request to its "local" ENRP server to request the list of Pool Elements (PE's). The ASAP layer queues the data to be sent in local buffers until the ENRP server responds.

4) The ENRP server would return a list of the three PE's A, B and C to the ASAP layer in PU(X).

5) PU(X)'s ASAP layer would now populate its local cache with the name mapping "File Transfer Pool" to the returned list and, using the server selection algorithm in the reply message from the ENRP server, would select a member of the pool and transmit its request. Note that the selection algorithm could involve queries to other entities, but for illustrative purposes we will assume that the "File Transfer Pool" is using a simple round robin selection scheme that would not involve any reference to other balancing or plug in agents.

6) The sending of the "login" request to the selected PE would implicitly bring up an SCTP association. In our example the requesting PU would allow a large number of streams inbound to it.

7) The selected server (for illustrative purposes 'PE(1,A)') would respond on stream 0 of the SCTP association with a challenge for a password.

8) The user would type the password in, and send it to the handle returned to it in the successful send call (the send of the login) over stream 0 to complete the login.

9) PE(1,A) would accept the password and send back a prompt on stream 0.

10) The user would now type a "get filename" to retrieve the file he wanted. This would generate a send of a request to stream 0 but it would specify a particular SCTP stream in place of an IP address and port for the server to connect to.

11) The server would read the command on stream 0 and begin sending the file according to its file transfer protocol on the specified SCTP stream number.

12) If during the file transfer conversation, PE(1,A) fails, assuming the PE's were sharing state of file transfer, a fail-over to PE(1,B) could be initiated. PE(1,B) would continue the transfer until complete. In parallel a request would be made to ENRP to request a cache update for the server pool "File

Transfer Pool" and a report would also be made that PE(1,A) is non-responsive (this would cause appropriate audits that may remove PE(1,A) from the pool if the ENRP servers had not already detected the failure).

13) At file transfer completion the "end of transfer" record would be passed over our stream.

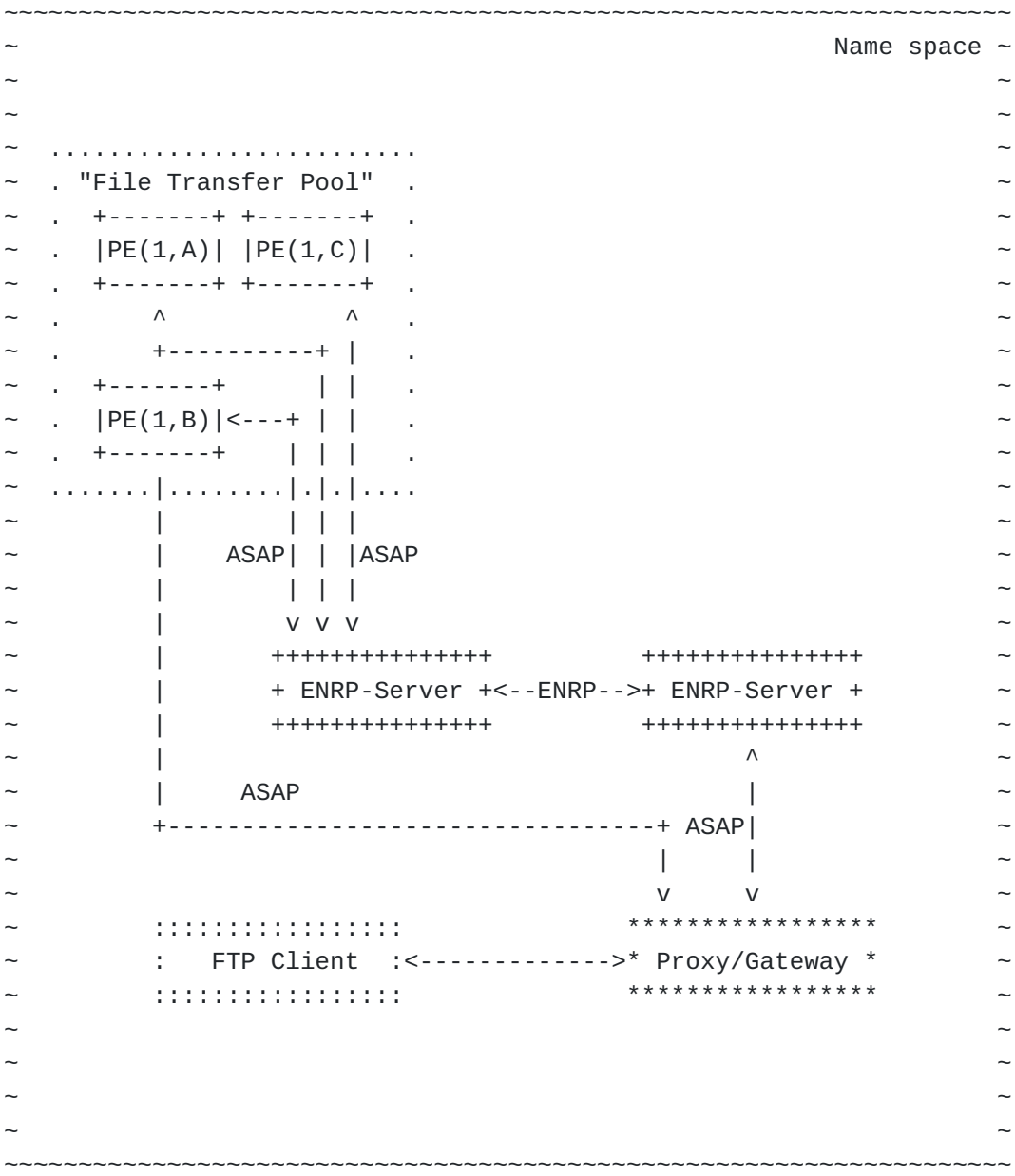
14) The user now types quit, ending our session and shutting down the SCTP association.

15) A subsequent request for login would result in a 'cache hit' using the last cache entry and possibly picking PE(1,C), repeating the sequence between (6) to (14).

2.5.2. The ENRP/ASAP unaware client file transfer using TCP and SCTP.

In this example we investigate the use of a Proxy server assuming the

same set of scenario as illustrated above.



In this example the steps will occur:

A) The Proxy/Gateway listens on the TCP port for FTP.

B) The Client connects to the Proxy and sends a login request.

C) The Proxy/Gateway follows steps (1) - (7) above to find and begin the login process.

D) When the challenge for password arrives at the Proxy (Step 8) it is forwarded to the FTP client.

E) The user responds on the TCP connection which is forwarded completing step 8) above.

F) When step 9 above happens, the login acceptance is forwarded through the TCP connection.

G) A new TCP connection is established between the listening FTP client and the proxy. Then the get command is forwarded through the proxy to the selected PE.

H) As the subsequent steps commence, the data is transmitted from the selected PE to the proxy server and then forwarded from the proxy over the new TCP data connection to the client. Commands from the server are forwarded to the command TCP connection by the proxy and user input (from the FTP client) is forwarded to stream 0 by the proxy.

Note that in this example high availability is maintained between the Proxy and the server pool but a single point of failure exists between the FTP client and the Proxy, i.e. the command TCP connection and its one IP address it is using for commands.

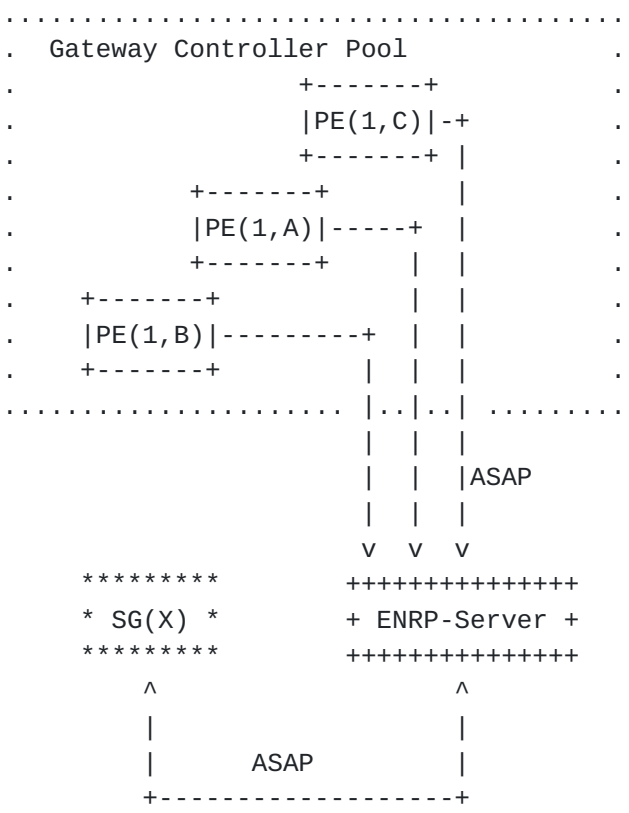
2.6. Telephony Signaling Example

This example shows the use of ASAP/Rserpool to support server pooling for high availability of a telephony application such as a Voice over IP Gateway Controller service.

2.6.1. Initial Server Selection using RSerpool

In this stage, the Signaling Gateway (SG) attempts to locate the Gateway

Controller server to handle an incoming signaling message.



As shown in the figure, the following sequence takes place:

- the Pool Elements (PEs) in the pool have registered with the ENRP Name Server(s) under the appropriate name for the GWC function

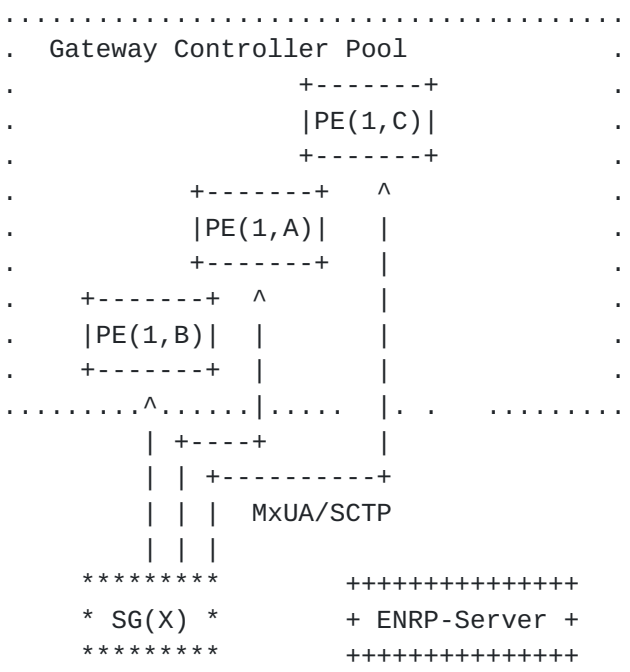
- the Signaling Gateway (SG) receives an incoming signaling message to be forwarded to the GWC. SG(X)'s ASAP layer would send an ASAP request to its "local" ENRP server to request the list of Pool Elements (PE's). The ASAP layer queues the data to be sent in local buffers until the ENRP server responds.

- the ENRP server would return a list of the three PE's A, B AND C to the ASAP layer in SG(X) together with information to be used for load-sharing traffic across the gateway controller pool.

2.6.2. Access to the Pool

ASAP then uses this information to decide which server to send the message to, using an adaptation layer MxUA, as defined in [RFC2719], and

an SCTP association to that server.



As shown in the figure, subsequent signaling messages may be sent to different servers in the pool according to the load sharing method agreed upon between the ENRP server and the pool elements.

2.6.3. Variations

Some possible variations in this architecture might include:

- combining the ENRP-Server function internally into the same physical system as the SG

3. General Requirements

The general architecture should be based on a peer to peer model. However, the binding should be based on a client server model.

It should be possible to use the protocol stack in small devices, like cellular phones. Therefore it must be possible to implement the protocols on clients with a large range of processing power.

Furthermore, it is expected that there will be a transition phase with some systems supporting the rserpool architecture and some are not. To make this transition as seamless as possible it should be possible for hosts not supporting this architecture to use also the new pooling services via some mechanism.

Another important requirement is that servers should be able to enter (become PEs) and leave server pools transparently without an interruption in service. It must also be possible that ENRP servers integrate themselves into the name space system.

The protocols used for the pool handling should not cause network congestion. This means that it should not generate heavy traffic, even in case of failures, and has to use flow control and congestion avoidance algorithms which are interoperable with currently deployed techniques, especially the flow control of TCP [RFC793] and SCTP [RFC2960]. Therefore, for large pools, only a subset of all possible IP-addresses are returned by the system.

There must be a way for the client to provide information to the ENRP server about the pool elements.

The architecture should not rely on multicast capabilities of the underlying layer. Nevertheless, it can make use of it if multicast capabilities are available.

4. Namespaces and Pools

Services are provided to the clients through a pool of servers. Clients will access these servers by name. The namespaces are flat. Selection of a server in the pool will be performed on behalf of the client. If more that one server registers under a name, this name becomes a name of a server pool. The name resolution results in access to one specific server out of a pool of servers. The selection of the server is transparent to the client and is governed by server pool policy.

Servers are registered by name in a namespace to join a server pool. There will be no globally unique namespace available, so multiple independent namespaces must be supported.

Since it is necessary to support multiple namespaces, it should also be possible for a host to refer to entities in namespaces the host does not belong to.

It must also be possible for a host to be registered in more than one namespace or using multiple names in one namespace.

A namespace can consist of a large number (up to 500) of pools. This upper limit is important since the system will be used for real time applications. So handling of name resolution has to be fast. Another consequence of the real time requirement is the supervision on the pool entities. The name resolution should not result in a pool element which is not able to provide the required service.

The registration and deregistration process is a dynamic one. It must be possible for hosts to register in a pool without affecting the other elements of a pool. This will be used for example, if a pool is under high load and more servers are installed to provide the service of the pool. It must also be possible to remove host from a pool without affecting the rest.

5. Server selection

Services are provided by a pool of servers. If a client wants to connect to a server one of the servers of the pool has to be chosen. This functionality is called server selection.

Server selection is driven by server pool policy. Some examples of selection policies are load balancing and round robin. The set of supported policies should be extensible in the sense that new policies can be added as required.

The ENRP servers should be extensible using a plug-in architecture. Then clients can provide some hints for the ENRP servers. Combining this information with the plug-ins will result in a more refined server selection by ENRP servers.

The server selection should not be based on internal features of the underlying transport protocol. This means, for example, in the case of SCTP that only the state of associations will be taken into account and not the state of the paths of the associations.

For some applications it is important that a client repeatedly connect to the same server in a pool. This feature should be supported if it is possible, i. e. if that server is still alive.

6. Reliability aspects

Host failures are handled by the pool concept. If one pool element fails and there are other pool elements which are able to provide the service than the other pool elements will be used.

Transaction failover is not provided by reliable server pooling. 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.

Network failures have to be handled by the transport protocol. This means that the underlying layer protocol must provide at least a acknowledged error-free non-duplicated transfer data delivery service. Therefore SCTP is the preferred (required) transport protocol for Rserpool.

7. Security aspects

Security is a major point of this architecture. There are several aspects which have to be considered:

- The transport layer protocol used should support concepts for dealing with denial of service attacks.
- The security architecture must be scalable.
- The ENRP Client has to use authentication before registration, deregistration.
- It should be possible that the name resolution is encrypted.
- The communication between ENRP Servers must fulfill the same requirements as the communication between ENRP clients and servers.
- Different trust relationships should be supported.
- The server registration (becoming a PE) should be based on an authentication.
- The security architecture must support hosts having a wide range of processing power.
- It should be possible to encrypt the communication between the client and the host.

8. Acknowledgements

The authors would like to thank Melinda Shore, Werner Vogels and many others for their valuable comments and suggestions.

9. References

[RFC793] J. B. Postel, "Transmission Control Protocol", RFC 793, September 1981.

[RFC959] J. B. Postel, J. Reynolds, "File Transfer Protocol (FTP)", RFC 959, October 1985.

[RFC2026] S. Bradner, "The Internet Standards Process -- Revision 3", RFC 2026, October 1996.

[RFC2719] L. Ong et al., "Framework Architecture for Signaling Transport", RFC 2719, October 1999.

[RFC2960] R. R. Stewart et al., "Stream Control Transmission Protocol", RFC 2960, November 2000.

10. Authors' Addresses

Michael Tuexen
Siemens AG
ICN WN CS SE 51
D-81359 Munich
Germany
Tel.: +49 89 722 47210
e-mail: Michael.Tuexen@icn.siemens.de

Qiaobing Xie
Motorola, Inc.
1501 W. Shure Drive, #2309
Arlington Heights, Il 60004
USA
Tel.: +1 847 632 3028
e-mail: qxie1@email.mot.com

Randall Stewart
Cisco Systems, Inc.
24 Burning Bush Trail
Crystal Lake, Il 60012
USA
Tel.: +1 815 477 2127
e-mail: rrs@cisco.com

Eliot Lear
Cisco Systems, Inc.

USA
Tel.: +1
e-mail: lear@cisco.com

Lyndon Ong
Nortel Networks
4401 **Great America Parkway**
Santa Clara, CA 95054
USA

Tel.: +1 408 495 5072
e-mail: long@nortelnetworks.com

Maureen Stillman
Nokia
127 **W. State Street**
Ithaca, NY 14850
USA

Tel.: +1 607 273 0724 62
e-mail: maureen.stillman@nokia.com

This Internet Draft expires May 24, 2001.