Reliable Server Pooling Working group
Internet-Draft
Expires: January 16, 2005

L. Coene
Siemens
P. Conrad
University of Delaware
P. Lei
Cisco
July 18, 2004

Reliable Server pool applicability Statement <draft-ietf-rserpool-applic-02.txt>

Status of this Memo

Copyright Notice

Abstract

This document describes the applicability of the reliable server pool architecture and protocols to applications which want to have High availability services. This is accomplished by using redundant servers and failover between servers of the same pool in case of

server failure. Processing load in a pool may de distributed/shared between the members of the pool according to a certain policy. Also some guidance is given on the choice of underlying transport protocol (and corresponding transport protocol mapping) for transporting application data and Rserpool specific control data.

Table of Contents

1. INTRODUCTION

Reliable server pooling provides protocols for providing higly available services. The services are located in pool of redundant servers and if a server fails, another server will take over. The only requirement put on these servers belonging to the pool is that if state is maintained by the server, this state must be transfered to the other server taking over. The mechanism for transfering this state information is NOT part of the Reliable server pooling architecture and/or protocols and must be provided by other protocols.

The goal is to provide server based redundancy. Transport and network level redundancy are handled by the transport and network layer protocols.

The application may choose to distribute its traffic over the servers of the pool conforming to a certain policy.

The application wishing to make use of Rserpool protocols may use different transport layers(such as TCP and SCTP). However some transport layers may have restrictions build in in the way they might be operating in the Rserpool architecture and its protocols.

1.1 Scope

The scope of this document is to explore the different ways that Reliable server pool protocols can be used in order to provide a highly available service towards applications with different requirements.

1.2 Terminology

The terms are commonly identified in related work and can be found in the Aggregate Server Access Protocol and Endpoint Name Resolution Protocol Common Parameters documentRFC ARCH [2].

2. Reliable serverpool

2.1 Architecture

A overview of the reliable server pool architecture is given in the Rserpool architecture document RFC ARCH [2].

The Rserpool architecture is made up of clients(Pool Users - PU) and servers(Pool Elements - PE). Both PU and PE's can be grouped into a pool in which a PE provides a service(File transfer, storage, bank transaction) to a PU. The PU's may try to find out via name resolution servers using the Aggregate Server Access Protcol (ASAP) which PE's are active. The PU can set up a communication channel with a particular PE(chosen out of the server pool) by using ASAP or by using directly any of the transport protocols(UDP/TCP/SCTP/ RTP...). ASAP may be running on top of TCP or SCTP.

The name resolution servers may communicate among themselves via the use of the Endpoint Name Resolution protocol(ENRP). This will allow for name resolution and synchronisation of the namespace used.

The minimum mode of using Rserpool is to use only ASAP for Endpoint name resolution. The PU may setup the client - server communication WITHOUT ASAP, but using present transport protocols(such as UDP, TCP..)

The normal use of Rserpool is to use ASAP for Enpoint name resolution AND for client - server communication. ASAP may be using as underlying transport protocol TCP or SCTP.

2.2 ASAP/ENRP applicability

2.2.1 Minimal Rserpool service

The minimum service provided by Rserpool is the use of ASAP for Endpoint name resolution. The ASAP procol may be running over TCP or SCTP.

- Endpoint name resolution
- no automatic failover from one PE to another, has to be done by the application itself
- bussinesscard or cookie mechanism not possible
- May be used by already existing applications which do not want to change the interface between PU and PE.
- Only PU-NS and PE-NS communication will use Rserpool protocols

2.2.2 Full Rserpool service

The fullservice provided by Rserpool is the use of ASAP for both

Endpoint name resolution and for PU - PE communication . ASAP may be running over TCP or SCTP.

- Endpoint name resolution
- automatic failover from one PE to another is transparent for the application itself
- bussinesscard exhange for determining if a PU is a pool or not. It allows the PE to treat the PU's as pool and use Rserpool protocols for it
- cookie mechanism can be used for state transfer between PE's
- May be used by allready existing applications which do not want to change the interface between PU and PE.
- All entities will use Rserpool protocols for communication with their respective peers

2.2.3 Endpoint name resolution

Resolving a pool name towards the pool handle is the function of the ENRP servers.

- Endpoint name resolution for PU's and PE's
- Home ENRP server for the PE's: de/registration
- Discovery, maintenance and synchronisation of the ENRP namespace
-
-

3. Application and Control data Transport

3.1 Rserpool use between 2 pools

Bussinesscards will allow to detect if their peer is part of a pool itself. Both the PU and the PE can be part of their own pools. If the PU or PE would fails, then the businesscard will have informed the respective peer to contact a alternative fellow PE/PU belonging to the pool.

3.2 state sharing via the cookie

Every time a response is send back, a cookie could be send along the response. The cookie is "encrypted" and is stored by the PU, no modification at all it done to the cookie . If a PE fails then the cookie is send to a alternate PE, the PE check if the cookie is valid. The contents of the cookie is only provided and validated by the PE. It can be used for state sharing between the PE.

3.3 PE Registration Services

Pool Elements ("server") must use the following services to add or remove themselves from server pools: REGISTER, to add the pool element into a server pool using {pool handle, mapping mode, protocol or mapping id, port, policy info} where mapping mode is defined in Section 5. A response result code is returned. DEREGISTER, to remove the pool element from a server pool using {pool handle, mapping mode, protocol or mapping id, port, policy info} where mapping mode is defined in Section 5. A response result code is returned.

3.4 Failover Callback Function

Session/transaction failover is not performed by ASAP themselves. "If a server 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." Tthe RSerPool implementation may provide a "hook" for applications to provide their own application- specific failover mechanism(s).

Specifically, an application can specify a callback function that is invoked whenever a failover has taken place. This callback function is invoked immediately after the new transport layer connection/ association is established with a new server, and gives the application the opportunity to send one or more messages that may help the server to resume any transaction or session that was in progress when the first server failed.

As a simple example of how such a callback is useful, consider a file transfer service built using RSerPool. Assume that some FTP mirroring software is used to maintain mirrored sites, and that the actual mirroring is out of scope. RSerPool would be used to select a server from among the available mirror sites, and to failover in the middle of a file transfer if a primary server fails.

In this example, assume that a simple request/response protocol is used, where one request message results in one or more response messages. Each request message contains the filename, and the offset desired within the file, (default zero.) Each response message contains some portion of the file, along with the offset, length of the portion in this message, and the length of the entire file.

A single request results is sufficient to result in a sequence of response messages from the requested offset to the end of the file. For simplicity, assume that the response messages are delivered by the underlying transport strictly in order (although this requirement could be relaxed if a small amount of extra complexity were introduced.)

In this protocol, all that is needed for failover is for the application to keep track of the number of bytes that it has read from the server, and to provide a callback function that reissues the request to the new server, replacing the offset with this number. When there is no failover, only one request message is sent and the minimum number of response messages are returned; in the event of failover(s), single new request message is sent for each failover that occurs.

While this is a simple example, for more complex application requirements, the failover callback could be used in a variety of ways:

- The client might send security credentials for authentication by the server, and/or to provide a "key" by which the server could locate and setup state by accessing some application-specific (and out-of-scope) state sharing mechanism used by the servers.
- The client might keep track of various synchronization points in the transaction, and use the failover callback to replay message from a recent synchronization point.

3.5 PE Selection Services

When automatic failover is enabled, selection of a new pool element according to the pool policy in place is automatically performed by the RSerPool framework in case of a detected failure (e.g. provides automatic failover). No application intervention is required. Automatic failover may be enabled by setting the appropriate send

flag when used in conjuction with data channel services (described in Section 4.6) or explicitly during initialization when data channel services are not used. FAILOVER_INDICATION, delivered by callback, indicates that a failover has occurred and that any required application level state recovery should be performed. The newly selected pool element handle is provided. Business Card services: when automatic failover is used, the exchange of business cards for rendezvous services is automatically performed by the RSerPool framework (e.g. no application intervention is required. When automatic failover is not enabled, failover detection and selection of an alternate PE must be done by the upper layer/ application. The following primitives are provided: GET_PRIMARY_SERVER, takes as input a pool handle and returns the {IP address, transport protocol, transport protocol port} of the primary server. GET_NEXT_SERVER has a dual meaning. First, it indicates to the RSerPool layer the failure of the server returned by a previous GET_PRIMARY_SERVER or GET_NEXT_SERVER call. Second, it provides the {IP address, transport protocol, transport protocol port} of the next server that should be contacted, according to the best information available to the RSerPool layer at the present time. The appropriate pool policy for server selection for the pool should be used for selecting the next server.

3.6 Upper Layer/Application Level Acknowledgements

The RSerPool framework provides an upper layer/application level ack service. The upper layer protocol may request that the peer acknowledge receipt and successful processing of its sent data, providing an additional degree of confidence over transport level message retrieval. When used in conjuction with the data channel services (described in Section 4.6), any unacknowledged data will be automatically sent to a new pool element in case of failover, if desired (e.g. automatic failover is enabled). The following service primitive is used to acknowledge an upper layer acknowledgement request. ULP_ACK, responds to a received upper layer acknowledgement request.

3.7 RSerPool Managed Data Channel

The RSerPool framework provides these services to send and receive application layer data, which are used in place of the direct call of transport level system functions (e.g. send/sendto, recv/recvfrom) and provides additional functionality to those calls.

DATA_SEND, to send data to a pool element by using a pool handle, specific pool element handle, or by transport address. An upper layer acknowledgement may be requested with this service. Appropriate error code(s) are returned. When sending to a pool

handle, the specific pool element handle is returned.

DATA_INDICATION, delivered by callback, to indicate that data has been received from a pool element and to pass that data to the application layer protocol. An application layer acknowledgement request can be indicated along with the data.

The application MAY direct that the RSerPool framework multiplex both the control and data channels onto the same SCTP association/TCP connection/ etc., if desired.

4. Transport protocols used by ENRP & ASAP

4.1 ASAP on top of TCP

TCP provides full reliable delivery with congestion control of the message to its peer node. It provides for a single homed, single stream delivery of a byte stream from or to the server. Change over will retrieve the unsent messages and send them on another TCP connection to a different server of the server pool.

ASAP uses the TCP mapping layer RFC TCPM [7]

4.2 ASAP on top of SCTP

PR-SCTP is the only know protocol which allows the choice of full, partial or no reliable delivery with congestion control of the message to its peer node. If the no-reliable delivery option is selected of SCTP, then ASAP will function as described in ASAP over UDP and including congestion control.

if multihoming, streams, unsequenced and/or assured delivery are required for the application, then SCTP should be used for ASAP. See SCTP aplicability statement RFC 3257 [10].

4.3 Address hiding

If an application requires only a single address(due to memory constraints) to reach a pool element of a pool , then ASAP can provide one address at a time when quering the ENRP server. If that pool element fails, then the client must request a new address from the ENRP server, before it can fail-over(as it has no information about the other pool elements of the same pool except the pool handle). This is done by ASAP itself in the full Rserpool service, but must be done by the client software itself in minimal Rserpool service.

This may require some buffering in the client during the failover.

5. Proxies and Rserpool

Application which require absolutely no protocol changes to their clients, may be able to use Rserpool protocols by using a proxy between the client and the server pool. Neither ASAP nor ENRP is used by the client application, but the proxy employs ENRP and ASAP. The client will only know the IP address and portnumbers of the proxy to contact. This can be accomplished via normal DNS queries.

The main drawback is that the proxy becomes the single point of failure for the connection between the client and the server.

6. Issues for Reliable Server pooling

6.1 State transfer accoss the server pool

Rserpool protocols(ENRP and ASAP) do NOT provide any service for directly transfering state information of a application from one Processing Element(PE) to another PE.

However by using the ASAP cookie mechanims, the PU may be able to transfer some state provided by the PE to the PU, to the new PE in case of failover. This is the responsability of the PU to do this.

6.2 ENRP Name servers

7. Security considerations

The protocols used in the Reliable server pool architecture only tries to increase the availability of the servers in the network. Rserpool protocols does not contain any protocol mechanisms which are directly related to user message authentication, integrity and confidentiality functions. For such features, it depends on the IPSEC protocols or on Transport Layer Security(TLS) protocols for its own security and on the architecture and/or security features of its user protocols.

A overview of possible treats to Reliable Server pooll protcols is detailed in RFC TREAT [9].

Rserpool architecture allows the use of different Transport protocols for its application and control data exchange. Those transport protocols may have mechanisms for reducing the risk of blind denial-of-service attacks and/or masquerade attacks. If such measures are required by the applications, then it is advised to check the SCTP applicability statement[RFC3057] for guidance on this issue.

8. Acknowledgments

The authors wish to thank H. Hazewinkel, M. Urena and M. Stillman and many others for their invaluable comments.

9 References

[1] Tuexen, M., Stewart, R., Shore, M., Xie, Q., Ong, L., Loughney, J. and M. Stillman, "Requirements for Reliable Server Pooling", RFC 3237, January 2002.

[2] Tuexen, M., Stewart, R., Shore, M., Xie, Q., Ong, L., Loughney, J. and M. Stillman, "Architecture for Reliable Server Pooling", Draft in progress , October 2002.

[3] Stewart, R., Xie, Q., Stillman, M. and M. Tuexen, "Aggregate Server Access Protocol (ASAP)", Draft in progress , October 2002.

[4] Xie, Q., Stewart, R. and M. Stillman, "Endpoint Name Resolution Protocol (ENRP)", Draft in progress , October 2002.

[5] Stewart, R., Xie, Q., Stillman, M. and M. Tuexen, "Aggregate Server Access Protocol and Endpoint Name Resolution Protocol Common Parameters", Draft in progress , October 2002.

[6] Conrad, P. and P. Lei, "Services Provided By Reliable Server Pooling", Draft in progress , January 2003.

[7] Conrad, P. and P. Lei, "TCP Mapping for Reliable Server Pooling Failover Mode", Draft in progress , June 2003.

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

[9] Stillman, M., Gopal, R., Sengodan, S., Guttman, E. and M. Holdrege, ""Threats Introduced by Rserpool and Requirements for Security in response to Threats"", RFC zzzz, Nov 2002.

[10] Coene, L., ""Stream Control Transmission Protocol Applicability statement"", RFC 3257, April 2002.

Authors' Addresses

Lode Coene
Siemens
Atealaan 32
Herentals 2200
Belgium

Phone: +32-14-252081
EMail: lode.coene@siemens.com

Phil Conrad
University of Delaware
USA

Phone: +
EMail: pconrad@acm.org

Peter Lei
Cisco
8735 W Higgins Rd, Suite 300
Chicago, IL 60631
USA

Phone: +1 847 870 7201
EMail: peter.lei@ieee.org

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.