Network Working Group
Request for Comments: 3237
Category: Informational

M. Tuexen
Siemens AG
Q. Xie
Motorola
R. Stewart
M. Shore
Cisco
L. Ong
Ciena
J. Loughney
M. Stillman
Nokia

January 2002

Requirements for Reliable Server Pooling

Status of this Memo

Copyright Notice

Abstract

This document defines a basic set of requirements for reliable server pooling.

The goal of Reliable Server Pooling (RSerPool) 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.

1. Introduction

1.1. Overview

The Internet is always on. Many users expect services to be always available; many businesses depend upon connectivity 24 hours a day, 7 days a week, 365 days a year. In order to fulfill this level of performance, many proprietary solutions and operating system dependent solutions have been developed to provide highly reliable and highly available servers.

This document defines requirements for an architecture and protocols enabling pooling of servers to support high reliability and availability for applications.

The range of applications that can benefit from reliable server pooling includes both mobile and real-time applications. Reliable server pooling mechanisms will be designed to support functionality for flexible pooling such as registration and deregistration, and load balancing of traffic across the server pool. Mechanisms will need to balance the needs of scalability, overhead traffic and response time to changes in pool status, as discussed below.

1.2. Terminology

This document uses the following terms:

Operation scope:
The part of the network visible to pool users by a specific instance of the reliable server pooling protocols.

Pool (or server pool):
A collection of servers providing the same application functionality.

Pool handle (or pool name):
A logical pointer to a pool. Each server pool will be identifiable in the operation scope of the system by a unique pool handle or "name".

Pool element:
A server entity having registered to a pool.

Pool user:
A server pool user.

Pool element handle (or endpoint handle):
A logical pointer to a particular pool element in a pool, consisting of the name of the pool and one or more destination transport addresses for the pool element.

Name space:
A cohesive structure of pool names and relations that may be queried by an internal or external agent.

Name server:
Entity which is responsible for managing and maintaining the name space within the RSerPool operation scope.

RSerPool:
The architecture and protocols for reliable server pooling.

1.3. Abbreviations

PE: Pool element

PU: Pool user

SCTP: Stream Control Transmission Protocol

TCP: Transmission Control Protocol

2. Requirements

2.1. Robustness

The solution must allow itself to be implemented and deployed in such a way that there is no single point of failure in the system.

2.2. Failover Support

The RSerPool architecture must be able to detect failure of pool elements and name servers supporting the pool, and support failover to available alternate resources.

2.3. Communication Model

The general architecture should support flexibility of the communication model between pool users and pool elements, especially allowing for a peer-to-peer relationship to support some applications.

2.4. Processing Power

It should be possible to use the protocol stack in small devices, like handheld wireless devices. The solution must scale to devices with a differing range of processing power.

2.5. Transport Protocol

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] and must be compliant with [RFC2914].

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

Network failures have to be handled and concealed from the application layer as much as possible by the transport protocol. This means that the underlying transport protocol must provide a strong network failure handling capability on top of an acknowledged error-free non-duplicated data delivery service. The failure of a network element must be handled by the transport protocol in such a way that the timing requirements are still fulfilled.

2.6. Support of RSerPool Unaware Clients

The architecture should allow for ease of interaction between pools and non-RSerPool-aware clients. However, it is assumed that only RSerPool-aware participants will receive maximum timing and notification benefits the architecture offers.

2.7. Registering and Deregistering

Another important requirement is that servers should be able to register to (become PEs) and deregister from a server pool transparently without an interruption in service. This means that after a PE has deregistered, it will continue to serve PUs which started their connection before the deregistration of the PE. New connections will be directed towards an alternative PE.

Servers should be able to register in multiple server pools which may belong to different namespaces.

2.8. Naming

Server pools are identified by pool handles. These pool handles are only valid inside the operation scope. Interoperability between different namespaces has to be provided by other mechanisms.

2.9. Name Resolution

The name resolution should not result in a pool element which is not operational. This might be important for fulfilling the timing requirements described below.

2.10. Server Selection

The RSerPool mechanisms must be able to support different server selection mechanisms. These are called server pool policies.

Examples of server pool policies are:

- Round Robin
- Least used
- Most used

The set of supported policies must be extensible in the sense that new policies can be added as required. Non-stochastic and stochastic policies can be supported.

There must be a way for the client to provide operational status feedback to the name server about the pool elements.

The name server protocols must be extensible to allow more refined server selection mechanisms to be implemented as they are developed in the future.

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

2.11. Timing Requirements and Scaling

Handling of name resolution must be fast to support real-time applications. Moreover, the name space should reflect pool membership changes to the client application as rapidly as possible, i.e., not waiting until the client application next reconnects.

The architecture should support control of timing parameters based on specific needs, e.g., of an application or implementation.

In order to support more rapid and accurate response, the requirements on scalability of the mechanism are limited to server pools consisting of a suitably large but not Internet-wide number of elements, as necessary to support bounded delay in handling real-time name resolution.

Also, there is no requirement to support hierarchical organization of name servers for scalability. Instead, it is envisioned that the set of name servers supporting a particular pool is organized as a flat space of equivalent servers. Accordingly, the impact of relatively frequent updates to ensure accurate reflection of the status of pool elements is limited to the set of name servers supporting a specific pool.

2.12. Scalability

The RSerPool architecture should not require a limitation on the number of server pools or on the number of pool users, although the size of an individual pool may be limited by timing requirements as defined above.

2.13. Security Requirements

2.13.1. General

- The scaling characteristics of the security architecture should be compatible with those given previously.

- The security architecture should support hosts having a wide range of processing powers.

2.13.2. Name Space Services

- It must not be possible for an attacker to falsely register as a pool element with the name server either by masquerading as another pool element or by registering in violation of local authorization policy.

- It must not be possible for an attacker to deregister a server which has successfully registered with the name server.

- It must not be possible for an attacker to spoof the response to a query to the name server

- It must be possible to protect the privacy of queries to the name server and responses to those queries from the name server.

- Communication among name servers must be afforded the same protections as communication between clients and name servers.

2.13.3. Security State

The security context of an application is a subset of the overall context, and context or state sharing is explicitly out-of-scope for RSerPool. Because RSerPool does introduce new security vulnerabilities to existing applications application designers employing RSerPool should be aware of problems inherent in failing over secured connections. Security services necessarily retain some state and this state may have to be moved or re-established. Examples of this state include authentication or retained ciphertext

for ciphers operating in cipher block chaining (CBC) or cipher feedback (CFB) mode. These problems must be addressed by the application or by future work on RSerPool.

3. Security Considerations

Security issues are discussed in section 2.13.

4. Acknowledgements

The authors would like to thank Bernard Aboba, Matt Holdrege, Eliot Lear, Christopher Ross, Werner Vogels and many others for their invaluable comments and suggestions.

5. References

[RFC793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC959] Postel, J. and J. Reynolds, "File Transfer Protocol (FTP)", STD 9, RFC 959, October 1985.

[RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.

[RFC2608] Guttman, E., Perkins, C., Veizades, J. and M. Day, "Service Location Protocol, Version 2", RFC 2608, June 1999.

[RFC2719] Ong, L., Rytina, I., Garcia, M., Schwarzbauer, H., Coene, L., Lin, H., Juhasz, I., Holdrege, M. and C. Sharp, "Framework Architecture for Signaling Transport", RFC 2719, October 1999.

[RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, September 2000.

[RFC2960] 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, November 2000.

6. Authors' Addresses

Michael Tuexen
Siemens AG
ICN WN CS SE 51
D-81359 Munich
Germany

Phone: +49 89 722 47210
EMail: Michael.Tuexen@icn.siemens.de

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

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

Randall Stewart
Cisco Systems, Inc.
24 Burning Bush Trail
Crystal Lake, Il 60012
USA

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

Melinda Shore
Cisco Systems, Inc.
809 Hayts Rd
Ithaca, NY 14850
USA

Phone: +1 607 272 7512
EMail: mshore@cisco.com

Lyndon Ong
Ciena
10480 Ridgeview Court
Cupertino, CA 95014
USA

Phone: +1 408 366 3358
EMail: lyong@ciena.com

John Loughney
Nokia Research Center
PO Box 407
FIN-00045 Nokia Group
Finland

Phone: +358 50 483 6242
EMail: john.loughney@nokia.com

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

Phone: +1 607 273 0724 62
EMail: maureen.stillman@nokia.com

7. Full Copyright Statement

Acknowledgement

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