Abstract

This draft describes the application of Reliable Server Pooling (RSerPool) for Virtual Network Function Resource Pooling (VNFPOOL).

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on February 22, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect
to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction ............................................. 2
   1.1. Abbreviations ........................................... 2
2. Virtual Network Function Resource Pooling ................... 3
3. Reliable Server Pooling ................................... 3
   3.1. Introduction ............................................ 3
   3.2. Registrar Operations .................................... 4
   3.3. Pool Element Operations ................................. 5
   3.4. Takeover Procedure ...................................... 5
   3.5. Pool User Operations .................................... 6
      3.5.1. Handle Resolution and Response ....................... 6
      3.5.2. Pool Member Selection Policies....................... 6
      3.5.3. Handle Resolution and Response ....................... 6
   3.6. Automatic Configuration ................................ 7
   3.7. State Synchronisation .................................. 7
      3.7.1. Cookies ................................................ 7
      3.7.2. Business Cards ........................................ 8
   3.8. Protocol Stack .......................................... 8
   3.9. Extensions ............................................... 9
   3.10. Reference Implementation and Deployment............... 9
4. Usage of Reliable Server Pooling ........................... 9
5. Security Considerations ................................... 10
6. IANA Considerations ....................................... 10
7. TestBed Platform ........................................... 10
8. Acknowledgments ........................................... 10
9. References ............................................... 10
   9.1. Normative References ................................... 10
   9.2. Informative References ................................. 11
Authors’ Addresses ........................................... 14

1. Introduction

1.1. Abbreviations

- PE: Pool Element
- PR: Pool Registrar
- PU: Pool User
- RSerPool: Reliable Server Pooling
2. Virtual Network Function Resource Pooling

Virtualised Network Function (VNF) (e.g. vFW, vLB) -- as introduced in more detail in [12] -- provides the same function as the equivalent network function (e.g. FW, LB), but is deployed as software instances running on general purpose servers via virtualisation platform. The main features of VNF include the following aspects:

1. A service consists of a sequence of topologically distributed VNF instances where the data connections are preferably directly established between the instances.

2. There are potentially more factors that cause VNF instance transition or even failure; VNF pool refers to a group of VNF instances providing same network function.

Virtualisation technology allows network function virtualisation operators to build a reliable VNF by pooling the underlying resources, such as CPU, storage, networking, etc. to form a cluster of VNF instances. VNF pool refers to a cluster or group of VNF instances providing same network function. Each VNF pool has a Pool Manager (PM) to manage the VNF instance such as instance selection, monitoring, etc. There will be a redundancy mechanism for a reliable PM to achieve reliable VNF. More details on VNF pool can be found in [12].

3. Reliable Server Pooling

3.1. Introduction
An overview of the RSerPool framework -- which is defined as RFC in [2] -- is provided in Figure 1. There are three types of components:

- Pool Element (PE) denotes a server in a pool. PEs in the same pool provide the same service.
- Pool User (PU) denotes a client using the service of a pool.
- Pool Registrar (PR) is the management component for the pools.

The set of all pools within an operation scope (for example: an organisation, a company or a department) is denoted as handlespace. Clearly, a single PR would be a single point of failure. Therefore, PRs also have to be redundant. Within the handlespace, each pool is identified by a unique pool handle (PH).

### 3.2. Registrar Operations

The PRs of an operation scope synchronise their view of the handlespace by using the Endpoint haNdlespace Redundancy Protocol (ENRP, defined as RFCs in [4], [5]). In contrast to for instance the Domain Name System (DNS), an operation scope is restricted to a single administrative domain. That is, all of its components are under the control of the same authority (for example: a company). This property leads to small management overhead, which also allows for RSerPool usage on devices having only limited memory.
and CPU resources (for example: telecommunications equipment). Nevertheless, PEs may be distributed globally to continue their service even in case of localised disasters (like for example an earthquake). Each PR in the operation scope is identified by a PR ID, which is a randomly chosen 32-bit number.

3.3. Pool Element Operations

Within their operation scope, the PEs may choose an arbitrary PR to register into a pool by using the Aggregate Server Access Protocol (ASAP, defined as RFCs in [3], [5]). The registration is performed by using an ASAP_REGISTRATION message. Within its pool, a PE is characterised by its PE ID, which is a randomly chosen 32-bit number. Upon registration at a PR, the chosen PR becomes the Home-PR (PR-H) of the newly registered PE. A PR-H is responsible for monitoring the availability of its PEs by ASAP_ENDPOINT_KEEP_ALIVE messages (to be acknowledged by a PE via an ASAP_ENDPOINT_KEEP_ALIVE_ACK message within a configured timeout). The PR-H propagates the information about its PEs to the other PRs of the operation scope via ENRP_UPDATE messages.

PEs re-register regularly in an interval denoted as registration lifetime and for information updates. Similar to the registration, a re-registration is performed by using another ASAP_REGISTRATION message. PEs may intentionally deregister from the pool by using an ASAP_DEREGISTRATION message. Also like for the registration, the PR-H makes the deregistration known to the other PRs within the operation scope by using an ENRP_UPDATE message.

3.4. Takeover Procedure

As soon as a PE detects the failure of its PR-H (that is: its request is not answered within a given timeout), it simply tries another PR of the operation scope for its registration and deregistration requests. However, as a double safeguard, the remaining PRs also negotiate a takeover of the PEs managed by a dead PR. This ensures that each PE again gets a working PR-H as soon as possible. The PRs of an operation scope monitor the availability of each other PR by using ENRP_PRESENCE messages, which are transmitted regularly. If there is no ENRP_PRESENCE within a given timeout, the peer is assumed to be dead and a so-called takeover procedure (see also [21] for more details) is initiated for the PEs managed by the dead PR: from all PRs having started this takeover procedure, the PR with the highest PR ID takes over the ownership of these PEs. The PEs are informed about being taken over by their new PR-H via an ASAP_ENDPOINT_KEEP_ALIVE with Home-flag set. The PEs are requested to adopt the sender of this Home-flagged message as their new PR-H.
3.5. Pool User Operations

3.5.1. Handle Resolution and Response

In order to access the service of a pool given by its PH, a PU requests a PE selection from an arbitrary PR of the operation scope, again by using ASAP. This selection procedure is denoted as handle resolution. Upon reception of a so-called ASAP_HANDLE_RESOLUTION message the PR selects the requested list of PE identities and returns them in an ASAP_HANDLE_RESOLUTION_RESPONSE message.

3.5.2. Pool Member Selection Policies

The pool-specific selection rule is denoted as pool member selection policy or shortly as pool policy. Two classes of load distribution policies are supported: non-adaptive and adaptive strategies (a detailed overview is provided by [16], [18], [23], [19]). While adaptive strategies base their selections on the current PE state (which requires up-to-date information), non-adaptive algorithms do not need such data. A basic set of adaptive and non-adaptive pool policies is defined as RFC in [7].

Defined in [7] are the non-adaptive policies Round Robin (RR), Random (RAND) and Priority (PRIO) as well as the adaptive policies Least Used (LU) and Least Used with Degradation (LUD). While RR/RAND select PEs in turn/randomly, PRIO selects one of the PEs having the highest priority. PRIO can for example be used to realise a master/backup PE setup. Only if there are no master PEs left, a backup PE is selected. Round-robin selection is applied among PEs having the same priority. LU selects the least-used PE, according to up-to-date application-specific load information. Round robin selection is applied among multiple least-loaded PEs. LUD, which is evaluated by [20], furthermore introduces a load decrement constant that is added to the actual load each time a PE is selected. It is used to compensate inaccurate load states due to delayed updates. An update resets the load to the actual load value.

3.5.3. Handle Resolution and Response

PE may fail, for example due to hardware or network failures. Since there is a certain latency between the actual failure of a PE and the removal of its entry from the handlespace -- depending on the interval and timeout for the ASAP_ENDPOINT_KEEP_ALIVE monitoring -- the PUs may report unreachable PEs to a PR by using an ASAP_ENDPOINT_UNREACHABLE message. A PR locally counts these reports for each PE and when reaching the threshold MAX_BAD_PE-REPORT (default is 3, as defined in the RFC [3]), the PR may decide to remove the PE from the handlespace. The counter of a PE is reset...
upon its re-registration. More details on this threshold and guidelines for its configuration can be found in [22].

3.6. Automatic Configuration

RSerPool components need to know the PRs of their operation scope. While it is of course possible to configure a list of PRs into each component, RSerPool also provides an auto-configuration feature: PRs may send so-called announces, that is, ASAP_ANNOUNCE and ENRP_PRESENCE messages which are regularly sent over UDP via IP multicast. Unlike broadcasts, multicast messages can also be transported over routers (at least, this is easily possible within LANs). The announces of the PRs can be heard by the other components, which can maintain a list of currently available PRs. That is, RSerPool components are usually just turned on and everything works automatically.

3.7. State Synchronisation

RSerPool has been explicitly designed to be application-independent. Therefore, RSerPool has not intended to define special state synchronisation mechanisms for RSerPool-based applications. Such state synchronisation mechanisms are considered as tasks of the applications themselves. However, RSerPool defines two mechanisms to at least support the implementation of more sophisticated strategies: Cookies and Business Cards. Details on these mechanisms can also be found in Subsection 3.9.5 of [16].

3.7.1. Cookies

ASAP provides the mechanism of Client-Based State Sharing as introduced in [17]. Whenever useful, the PE may package its state in form of a state cookie and send it -- by an ASAP_COOKIE message -- to the PU. The PU stores the latest state cookie received from the PE. Upon PE failure, this stored cookie is sent in an ASAP_COOKIE_ECHO to the newly chosen PE. This PE may then restore the state. A shared secret known by all PEs of a pool may be used to protect the state from being manipulated or read by the PU.

While Client-Based State Sharing is very simple, it may be inefficient when the state changes too frequently, is too large (the size limit of an ASAP_COOKIE/ASAP_COOKIE_ECHO is 64 KiB) or if it must be prevented that a PU sends a state cookie to multiple PEs in order to duplicate its sessions.
3.7.2. Business Cards

Depending on the application, there may be constraints restricting the set of PEs usable for failover. The ASAP_BUSINESS_CARD message is used to inform peer components about such constraints.

The first case to use a Business Card is if only a restricted set of PEs in the pool may be used for failover. For example, in a large pool, each PE can share its complete set of session states with a few other PEs only. This keeps the system scalable. That is, a PE in a pool of n servers does not have to synchronise all session states with the other n-1 PEs. In this case, a PE has to tell its PU the set of PE identities being candidates for a failover using an ASAP_BUSINESS_CARD message. A PE may update the list of possible failover candidates at any time by sending another Business Card. The PU has to store the latest list of failover candidates. Of course, if a failover becomes necessary, the PU has to select from this list using the appropriate pool policy -- instead of performing the regular PE selection by handle resolution at a PR. Therefore, some literature also denotes the Business Card by the more expressive term "last will".

In symmetric scenarios, where a PU is also a PE of another pool, the PU has to tell this fact to its PE. This is realised by sending an ASAP_BUSINESS_CARD message to the PE, providing the PH of its pool. Optionally, also specific PE identities for failover may be provided. The format remains the same as explained in the previous paragraph. If the PE detects a failure of its PU, the PE may -- now in the role of a PU -- use the provided PH for a handle resolution to find a new PE or use the provided PE identities to select one. After that, it can perform a failover to that PE.

3.8. Protocol Stack

The protocol stack of a PR provides ENRP and ASAP services to PRs and PEs/PUs respectively. But between PU and PE, ASAP provides a Session Layer protocol in the OSI model. From the perspective of the Application Layer, the PU side establishes a session with a pool. ASAP takes care of selecting a PE of the pool, initiating and maintaining the underlying transport connection and triggering a failover procedure when the PE becomes unavailable.

The Transport Layer protocol is by default SCTP (as defined in [1]) -- except for the UDP-based automatic configuration announces (see Section 3.6) -- over possibly multi-homed IPv4 and/or IPv6. SCTP has been chosen due to its support of multi-homing and its reliability features (see also [26]).
3.9. Extensions

A couple of extensions to RSerPool are existing: Handle Resolution Option defined in [9] improves the PE selection by letting the PU tell the PR its required number of PEs to be selected. ENRP Takeover Suggestion introduced in [11] ensures load balancing among PRs. [10] defines a delay-sensitive pool policy. [8] defines an SNMP MIB for RSerPool.

3.10. Reference Implementation and Deployment

RSPLIB is the Open Source reference implementation of RSerPool. It is currently -- as of February 2016 -- available for Linux, FreeBSD, MacOS and Solaris. It is actively maintained. Particularly, it is also included in Ubuntu Linux as well as in the FreeBSD ports collection. RSPLIB can be downloaded from [14]. Further details on the implementation are available in [16], [24].

RSerPool with RSPLIB is deployed in a couple of Open Source projects, including the SimProcTC Simulation Processing Tool-Chain for distributing simulation runs in a compute pool (see [25] as well as the simulation run distribution project explained in [26] for a practical example) as well as for service infrastructure management in the NorNet Core research testbed (see [27], [28]).

4. Usage of Reliable Server Pooling

**** TO BE DISCUSSED! ****

The following features of RSerPool can be used for VNFPOOL:

- Pool management.
- PE selection with pool policies.
- Session management with help of ASAP_BUSINESS_CARD.

The following features have to be added to RSerPool itself:

- Support of TCP including MPTCP as additional/alternative transport protocols.
- Possibly add some special pool policies?
- See also [13] for ideas on a next generation of RSerPool.

The following features have to be provided outside of RSerPool:
State synchronisation for VNFPOOL.

Pool Manager functionality as an RSerPool-based service.

5. Security Considerations

Security considerations for RSerPool can be found in [6]. Furthermore, [23] examines the robustness of RSerPool systems against attacks.

6. IANA Considerations

This document introduces no additional considerations for IANA.

7. Testbed Platform

A large-scale and realistic Internet testbed platform with support for Reliable Server Pooling and the underlying SCTP protocol is NorNet. A description of and introduction to NorNet is provided in [28], [29], [30]. Further information can be found on the project website [15] at https://www.nntb.no.

8. Acknowledgments

The authors would like to thank Xing Zhou for the friendly support.

9. References

9.1. Normative References


9.2. Informative References


Authors' Addresses

Thomas Dreibholz
Simula Research Laboratory, Network Systems Group
Martin Linges vei 17
1364 Fornebu, Akershus
Norway

Phone: +47-6782-8200
Fax: +47-6782-8201
Email: dreibh@simula.no
URI: https://www.simula.no/people/dreibh
Michael Tuexen
Muenster University of Applied Sciences
Stegerwaldstrasse 39
48565 Steinfurt, Nordrhein-Westfalen
Germany
Email: tuexen@fh-muenster.de
URI: https://www.fh-muenster.de/fb2/personen/professoren/tuexen/

Melinda Shore
No Mountain Software
PO Box 16271
Two Rivers, Alaska  99716
U.S.A.
Phone: +1-907-322-9522
Email: melinda.shore@nomountain.net
URI: https://www.linkedin.com/pub/melinda-shore/9/667/236

Ning Zong
Huawei Technologies
101 Software Avenue
Nanjing, Jiangsu  210012
China
Email: zongning@huawei.com
URI: https://cn.linkedin.com/pub/ning-zong/15/737/490