**JunOS Socket Replication Parallelization**

**Software Functional Specification**

**Ananda Kumar M R <anandmr>**

|  |  |  |  |
| --- | --- | --- | --- |
| Rev # | Date | Revised by | Comments |
| 1.0 | 24-July-2017 | Ananda Kumar M R | Initial version |

TEMPLATE REVISION HISTORY AND GUIDELINES:

The revision history for this template can be found at the end of this document.

Table of Contents

[1. Introduction 5](#_Toc488656438)

[1.1 Example Use Cases 5](#_Toc488656439)

[1.2 Reference 5](#_Toc488656440)

[1.3 RLI List 5](#_Toc488656441)

[1.4 Feature Parity Traceability 6](#_Toc488656442)

[2. Functionality 7](#_Toc488656443)

[2.1 Goals 7](#_Toc488656444)

[2.2 Junos Socket Replication (JSR) 7](#_Toc488656445)

[2.3 Statistics 8](#_Toc488656446)

[2.4 Other Considerations 8](#_Toc488656447)

[2.5 Testing 8](#_Toc488656448)

[2.6 Exceptions 9](#_Toc488656449)

[2.7 Assumptions 9](#_Toc488656450)

[2.8 Functional competitive data 9](#_Toc488656451)

[2.9 APIs/Messages 9](#_Toc488656452)

[2.10 Manageability 9](#_Toc488656453)

[2.11 Examples or Interaction Descriptions 10](#_Toc488656454)

[2.12 Supportability (Serviceability, Diagnose-ability and Fault Handling) 10](#_Toc488656455)

[2.13 Dependencies and Interactions with other Components in the System 10](#_Toc488656456)

[2.14 Legal Considerations 10](#_Toc488656457)

[3. Other Requirements 12](#_Toc488656458)

[4. System Resource Estimation 14](#_Toc488656459)

[4.1 Performance Related Resources 14](#_Toc488656460)

[5. Scaling and Performance 15](#_Toc488656461)

[5.1 Target Scaling 15](#_Toc488656462)

[5.2 Target Performance 15](#_Toc488656463)

[6. Compatibility Issues 16](#_Toc488656464)

[7. Security Considerations 17](#_Toc488656465)

[8. Platforms Supported 18](#_Toc488656466)

[9. Common Infrastructure 19](#_Toc488656467)

[9.1 High Availability (HA) 19](#_Toc488656468)

[9.2 Aggregated Ethernet/ SONET/ IRB Support 19](#_Toc488656469)

[9.3 Services/JSF (JUNOS Services Framework) Impact 19](#_Toc488656470)

[9.4 Multi-Chassis Support 19](#_Toc488656471)

[9.5 64-Bit Support 19](#_Toc488656472)

[9.6 IPv6 Support 19](#_Toc488656473)

[9.7 Logical System Support 19](#_Toc488656474)

[10. SDK Impact 20](#_Toc488656475)

[10.1 SDK Customer Usage 20](#_Toc488656476)

[11. JUNOS Ready Software considerations 21](#_Toc488656477)

[12. Notes 22](#_Toc488656478)

[13. Glossary 23](#_Toc488656479)

[14. Design Specification exception 24](#_Toc488656480)

[15. Functional Specification Approver Checklist 25](#_Toc488656481)

[16. Template Revision History and Guidelines 1](#_Toc488656482)

[16.1 Revision History 1](#_Toc488656483)

[16.2 Obsolete Document List As A Result of this Release 1](#_Toc488656484)

[16.3 Approver List for this Revision 1](#_Toc488656485)

Functional Specification Document Checklist

|  |  |  |  |
| --- | --- | --- | --- |
| Section | Section Title | Recommendation | Compliance |
| 1 | Introduction | MANDATORY |  |
| 1.1 | Reference | MANDATORY |  |
| 1.2 | RLI List | MANDATORY |  |
| 1.3 | Feature Parity Traceability | OPTIONAL |  |
| 2 | Functionality | MANDATORY |  |
| 2.1 | Goals | MANDATORY |  |
| 2.2 | Exceptions | MANDATORY |  |
| 2.2.1 | Caveats | MANDATORY |  |
| 2.2.2 | Limitations | MANDATORY |  |
| 2.2.3 | Non Goals | MANDATORY |  |
| 2.3 | Assumptions | MANDATORY |  |
| 2.4 | Functional Competitive Data | OPTIONAL |  |
| 2.5 | API/Messages | MANDATORY |  |
| 2.6 | Manageability | MANDATORY |  |
| 2.6.1 | CLI Config | MANDATORY |  |
| 2.6.2 | CLI Commands | MANDATORY |  |
| 2.6.3 | SNMP | MANDATORY |  |
| 2.6.4 | Syslog- ERRMSG | MANDATORY |  |
| 2.6.5 | Software Feature Licensing | MANDATORY |  |
| 2.6.6 | Software Packaging | MANDATORY |  |
| 2.6.7 | J-Web Quick Configuration and Monitor Screen | MANDATORY |  |
| 2.6.8 | Intergration with Management Systems | MANDATORY |  |
| 2.7 | Examples or Interaction Descriptions | OPTIONAL |  |
| 2.8 | Serviceability and diagnose-ability | MANDATORY |  |
| 2.9 | Dependencies and Interactions with other Components in the System | MANDATORY |  |
| 2.10 | Legal Considerations | MANDATORY |  |
| 3 | Other Requirements | OPTIONAL |  |
| 4 | System Resource Estimation | MANDATORY |  |
| 4.1 | Performance Related Resources | MANDATORY |  |
| 5 | Scaling and Performance | MANDATORY |  |
| 5.1 | Target Scaling | MANDATORY |  |
| 5.2 | Target Performance | MANDATORY |  |
| 6 | Compatibility Issues | MANDATORY |  |
| 7 | Security Considerations | MANDATORY |  |
| 8 | Platforms Supported | MANDATORY |  |
| 9 | Common Infrastructure |  |  |
| 9.1 | High Availability (HA) | MANDATORY |  |
| 9.1.1 | Graceful RE Switchover (GRES), ISSU and NSSU Impact | MANDATORY |  |
| 9.1.2 | NSR Impact | MANDATORY |  |
| 9.2 | Aggregated Ethernet/ SONET/ IRB Support | MANDATORY |  |
| 9.3 | Services/JSF (JUNOS Services Framework) Impact | MANDATORY |  |
| 9.4 | Multi-Chassis Support | MANDATORY |  |
| 9.5 | 64-Bit Support | MANDATORY |  |
| 9.6 | IPv6 Support | MANDATORY |  |
| 9.7 | Logical System Support | MANDATORY |  |
| 10 | SDK Impact | MANDATORY |  |
| 10.1 | SDK Customer Usage | MANDATORY |  |
| 11 | JUNOS Ready Software considerations | MANDATORY |  |
| 12 | Notes | OPTIONAL |  |
| 13 | Glossary | OPTIONAL |  |
| 14 | Design Specification exception | OPTIONAL |  |

# Introduction

The JUNOS SOCKET REPLICATION (JSR) is a key infra/module in kernel used for building NSR feature in JUNOS. JSR module is tightly coupled with JUNOS network TCP/IP stack and the socket layer.

JSR module replicates the data from a socket on the primary RE to the secondary RE. The idea is that during a switchover, the secondary RE's socket will contain all the information needed to ensure that the connection stays up. This information includes both the data in the socket buffer as well as any kernel state (TCP state for instance).

As a part of JUNOS network stack parallelization effort (RLI 23722, 27176, 27177), the JLOCK usage in JUNOS network IO path (TCP/IP) is changed from exclusive to shared mode to get the benefits of parallelism. But NSR feature was not included in that effort. Meaning, JUNOS network stack parallelization feature and NSR were mutually exclusive.

Now, the purpose of this RLI is to enable parallelization in JSR code so that JUNOS network stack parallelization feature can be enabled even along with NSR feature on JUNOS.

## Example Use Cases

As mentioned above, in the latest JUNOS release NSR feature cannot be enabled along with network stack parallelization feature. Once the JSR parallelization effort is complete both these features can co-exist. So, after the implementation of this RLI, TCP/IP input/output related to any of the application or the interrupt path and the JSR data path can execute in parallel on different CPUs.

## Reference

**Functional specification of RLI 23722, 27176, 27177**

[JunOS Network Stack Parallelization Functional Specification](https://junipernetworks.sharepoint.com/sites/engdoccenter/_layouts/15/DocIdRedir.aspx?ID=FQHC3P4XRDAF-1315789153-315626)

**Design specification of RLI 23722, 27176, 27177**

[JunOS Network Stack Parallelization Design Specification](http://cvs.juniper.net/cgi-bin/viewcvs.cgi/*checkout*/sw-projects/os/core-ipv4v6/RLI-23722-27176-27177-DS.docx?rev=1.5)

## RLI List

|  |  |
| --- | --- |
| **RLI No** | **Description** |
| 34388 | JunOS Socket Replication Parallelization |

## Feature Parity Traceability

N/A

# Functionality

As part of this RLIs, the data path in JSR code will be changed to use JLOCK in shared mode. The control path in JSR code will still continue with JLOCK in exclusive mode.

The control path in JSR code involves:

JSR handle Allocation/Cleanup, JSR Split, JSR Merge, Auto Merge/Keepalive, JSR Unreplicate (these are one time operations invoked during initialization/cleanup/error path) & JSR module specific timer routines.

The data path in JSR code involves:

Replication of outgoing data (JSR\_SDRL\_REPLICATE) on master/backup

Replication of incoming data (JSR\_PRL\_REPLICATE) on master/backup

Receiving/Sending of ACK for both incoming & outgoing replication messages on master/backup respectively.

No new sysctl or cli knob will be introduced as a part of this RLI to enable/disable this feature.

Enabling Network stack parallelization will enable this feature & disabling Network stack parallelization feature will disable this feature.

Also, RLI code changes are confined to kernel JSR module only. And the code changes neither change the existing JSR/NSR functionality nor introduce any new functionality. So, no code changes are required in the NSR applications that uses JSR infra (ex: RPD/BGP).

## Goals

Examine the use of JLOCK in JSR code to achieve the following –

* Change JLOCK in exclusive mode to JLOCK in shared mode wherever possible
* Eliminate the use of JLOCK altogether wherever possible

Overall goal is to enable parallelism in JSR code so that NSR feature can be enabled along with network stack parallelism feature in JUNOS.

## Junos Socket Replication (JSR)

JSR infra can be basically classified into two parts:

1. JSR APIs: These are API’s exported to applications so that they can make use JSR infra (control path)
2. JSR kernel infra: This is a complete kernel infrastructure that helps in replicating the actual socket data to backup RE (data path).

The JSR API’s (control path) constitutes, RTSOCK call from application to kernel requesting for JSR service, this path is currently protected by JLOCK in exclusive mode. Also, these are one-time operations (during replicated socket creation/closure). So, no changes/improvements will be made in this path as a part of this RLI.

In the JSR data path, both input and output path, packet is first sent to the backup RE (from socket layer in output path and IP layer in the input path) and only after receiving an acknowledgement from the backup RE, packet is injected to TCP layer.

In the output path, data written by the application is –

1. Written to the KKCM output queue in the context of the application & notifies (wakes up) the KKCM thread.
2. KKCM thread sends the packet from its output queue to backup RE over a TCP connection.
3. Once the acknowledgement is received from the back RE in the context of KKCM thread original packet is re-injected to the TCP layer.

In the input path, packet received by the IP layer is –

1. Written to the KKCM input queue in the context of the NetISR thread.
2. KKCM thread sends the packet from its input queue to backup RE over a TCP connection.
3. Once the acknowledgement is received from the back RE, in the context of KKCM thread original packet is re-injected to the TCP layer.

JSR framework uses socket extension data structure (jsrsock) to handle the above scenarios.

Currently, KKCM thread is running by holding the JLOCK in exclusive mode. Following options will be explored to improve the parallelism/performance.

1. Check if KKCM thread can be run in JLOCK shared mode so that multiple readers can run parallel when KKCM thread is running.
2. Check if JSR data structures can be protected with fine-grained locks. This would help in protecting these data structures in data path with JLOCK in shared mode.

## Statistics

Writing to statistics is another operation that requires exclusive access. Statistics increments are present in all the paths discussed. Existing code writes to statistics inline. There are no APIs to update statistics.

## Other Considerations

Explore the possibility of completely eliminating the use of JLOCK by replacing it with fine-grained locks. Since the JunOS network stack code itself is relaying on JLOCK and JSR code is tightly coupled with JunOS network stack, achieving this is going to be difficult.

## Testing

Changes in the JSR code will be tested using the jsrmon/jsrutil application. Multiple instances of the BGP sessions will be created between two nodes and the health of the BGP sessions will be monitored using netstat, jsrmon across multiple switchovers.

Since this RLI does not change existing functionality, all the existing regression suites related to NSR will have to be run.

All development will be done with WITNESS/INVARIANTS enabled.

## Exceptions

### Caveats

Note that this RLI deals only with JUNOS socket replication part of a packet’s in and out paths. Overall improvement and performance depends on other components, such as user space apps, other kernel infra like ifstate and peer proxy, and PFE, as well.

Accesses to JLOCK-protected JFE s/w data in the existing code are not organized. There are no well-defined APIs to read and write these data structures. A lot of data is protected by one coarse-grained lock. Accesses to data are spread all over the execution paths. This makes the job of moving these data structures to fine-grained locks difficult. That is not a goal of this RLI.

### Limitations

Changes made, as part of this RLI, will be limited to unicast execution paths for native IPv4/IPv6 over Ethernet interfaces.

### Non Goals

Though having parallelism in JSR code would yield some performance improvement in network IO (when NSR is enabled), This RLI is not primarily targeted towards improving the overall performance. The main goal of this RLI is to enable parallelism in JSR code so that NSR feature can be enabled along with network stack parallelism feature in JUNOS.

## Assumptions

## Functional competitive data

NA

## APIs/Messages

## Manageability

### CLI Config

No new CLI commands are needed.

#### CLI Config Details

NA

### CLI Commands

No new CLI commands are needed.

#### CLI Command Details

NA

#### JUNOScript

### SNMP

### Syslog – ERRMSG

No new Syslog messages.

### Software Feature Licensing

### Software Packaging

### J-Web Quick Configuration and Monitor Screen

### Integration with Management Systems

## Examples or Interaction Descriptions

## Supportability (Serviceability, Diagnose-ability and Fault Handling)

### Serviceability and Diagnose-ability

### Fault Handling

NA

## Dependencies and Interactions with other Components in the System

≥

## Legal Considerations

### Third-Party Materials:

Are there any **Third-Party (non-Juniper) Materials** incorporated into the SW? “**Third-Party Materials**” include (1) Open Source Software, technology, and materials (“OSS”), and (2) Third-Party commercial technology, materials, and code (including code embedded in an ASIC).

**[ ] Yes** **[ ] No**

**If Yes,** submit a [**Third-Party Tracking Chart**](http://www-int.jnpr.net/legal/ip/docs/third_party_tracking_chart.xls) created by your [**Codemaster**](http://www-int.jnpr.net/legal/ip/docs/list_of_codemasters.pdf)listing all such Third-Party Materials to your [BU IP Technology Transaction Attorney](http://www-int.jnpr.net/legal/contacts/byFunction.html?IP). Your “**Codemaster”** is the team member assigned to track all Third-Party Materials for a Functional Specification.

**All Third-Party Materials must be approved prior to Functional Specification approval**, as follows:

* All OSS must be approved in accordance with [**Juniper’s Inbound OSS Policy**](http://www-int.jnpr.net/legal/ip/docs/inbound_oss_policy.pdf); and
* All Third-Party commercial technology must be approved by Legal in accordance with Juniper’s [**Third-Party Commercial Technology Policy**](http://www-int.jnpr.net/legal/ip/docs/third_party_commercial_technology_policy.pdf).

### Export Control Matters

### Encryption

All encryption features/content must be reviewed and all US/International legal requirements met prior to HW/SW release. Such review/approval may take **up to 50 days**. “HW/SW” includes a system, card, IC device, software OS or application or module embedded therein, or revision of any of the foregoing.

|  |  |  |
| --- | --- | --- |
| **#** | **Encryption Questions** | **Yes/No** |
| a | Is there cryptography incorporated in, controlled by, or linked (dynamically or otherwise) to HW/SW? |  |
| b | Is cryptography used for key exchange, authentication, or message integrity? |  |
| c | Does cryptography operate in the HW/SW to decrypt already-encrypted content? |  |
| d | Is cryptography used to encrypt or decrypt network management, monitoring or administrative traffic? |  |
| e | Is “publicly available” or OSS-derived cryptography used? |  |
| f | Is the HW/SW designed to use encryption embedded in another product? |  |
| g | Is there cryptography in or used by the HW/SW that was furnished, or developed, by a Third-Party? |  |
| h | Does any hardware component incorporated in the HW/SW implement (or accelerate computations for) any cryptographic algorithm or security protocol? |  |

If you answered “Yes” to any of the above questions, then there is encryption and you must complete the [Encryption Checklist](https://www.t-baccess.com/juniper/engineering/default.aspx) at <https://www.t-baccess.com/juniper/engineering/default.aspx>.

### High-Speed IO Technology

Is the product, subsystem, component or feature designed to, or does it operate to enable the system in which it is incorporated to support >120Gbps per channel I/O? **[ ] Yes [ ] No**

If Yes, then email [kniven@juniper.net](mailto:kniven@juniper.net) and cc [compliance\_helpdesk@juniper.net](mailto:compliance_helpdesk@juniper.net) with “URGENT–High Speed Interconnect Issue” as the Subject Line. In the email, identify the project code name/feature description and a brief description of the product or feature.

For questions about encryption or the [Encryption Checklist](https://www.t-baccess.com/juniper/engineering/default.aspx), email [new\_product\_export\_classification@juniper.net](mailto:new_product_export_classification@juniper.net). For all other questions, email [legal-ip checklist@juniper.net](mailto:legal-ip%20checklist@juniper.net).

# Other Requirements

**NA**

# System Resource Estimation

NA

## Performance Related Resources

NA

# Scaling and Performance

The changes are expected to help improve performance. However, there are no specific targets. Also, other components like RPD, ifstate, peer-infra and PFE needs to be improved to gain the full benefit.

## Target Scaling

## Target Performance

# Compatibility Issues

NA

# Security Considerations

This RLI does not introduce any new functionality. It enables parallel execution of paths inside JunOS socket replication modules. As such, behavior of the system remains the same.

This RLI does not also change the way RE communicates with PFEs and the other RE. Hence it does not affect its current behavior with respect to Fresnel’s reflection.

# Platforms Supported

All platforms running JunOS socket replication is supported.

# Common Infrastructure

## High Availability (HA)

### Graceful RE Switchover (GRES), ISSU and NSSU Impact

Changes done for this RLI does not impact GRES/ISSU.

### NSR Impact

RLI code changes are confined to kernel JSR module only. And the code changes neither change the existing JSR/NSR functionality nor introduce any new functionality. So, no change/impact on NSR.

## Aggregated Ethernet/ SONET/ IRB Support

While execution paths over AE interfaces are in scope of this work, IRB is out of scope of this RLI.

## Services/JSF (JUNOS Services Framework) Impact

NA

## Multi-Chassis Support

Supported.

## 64-Bit Support

Supported.

## IPv6 Support

Supported.

## Logical System Support

Supported.

# SDK Impact

NA

## SDK Customer Usage

NA

# JUNOS Ready Software considerations

NA

# Notes

NA

# Glossary

NA

# Design Specification exception

NA

# Functional Specification Approver Checklist

|  |  |  |
| --- | --- | --- |
| **Org.** | **Role** | **Approver’s checklist**  (If you agree with the statements in the cell next to your role, please sign and date in the Approved by column) |
| PLM | Approver | The FS has detailed requirements for the feature and is in-line with the customer/product requirements. |
| SW Eng. | Approver | The FS has detailed functional and system scale & performance requirements and correctly captures the overall system impact |
| Sys Test | Approver | The FS has detailed functional and system scale & performance requirements for use to create the feature test plan |
| Information Experience | Approver | The FS has detailed functional and system scale & performance requirements for use as input to create the documentation plan |
| Architecture Reviewer | Approver  (optional) | The FS has detailed functional and system scale & performance requirements and is consistent with overall system architecture |
| Hardware | Approver  (Optional) | The FS has detailed functional and system scale & performance requirements and is consistent with the H/w design |
| Regression Test | Approver | The FS has detailed requirements that provide inputs to identify the regression test plan |
| Program Mgmt | Approver  (Optional) | The FS has detailed requirements and correctly captures the overall requirements for the feature |
| Customer Service | Approver  (Optional) | The FS captures the overall requirements for the feature and provide inputs required to create any Customer support plan document |
| Advanced Customer Engineering | Approver  (Optional) | The FS has detailed supportability requirements in Section 2.8 and captures supportability, diagnosability, debug requirements needs for the feature |
| Manufacturing | Approver  (Optional) | The FS correctly identifies any impact to Manufacturing process |

# Template Revision History and Guidelines

**NOTE**: This section may be deleted from the actual deliverable because it is dedicated to revisions of the template itself instead of the contents. For revisions of the contents of the Software Functional Specification for a specific program, please refer to the Document Revision History section at the beginning of this document.

## Revision History

|  |  |  |
| --- | --- | --- |
| ***Revision*** | ***Effective Date*** | ***Change Description*** |
| *3.0* | *1-Apr-2013* | *Update to version 3.0 as part of NPI Back to Basics to align versions in template, on website and in Documentum.* |
| *3.1* | *16-Apr-2013* | *Updated Function Specification Document Checklist table. Section 7 in the table now states "Security Consideration" instead of "Security Concerns" which matches the actual section name.*  *Updated Section 15 – Functional Specification Approver Checklist. Deleted last column "Approved by & date (or NA)" since this information is duplicate of DRT records - FS approval is managed via the DRT system, and hence this column is not needed.* |
| *3.2* | *29-Jul-2013* | *Updated Functional Specs Approvers Checklist to include ACE* |

## Obsolete Document List As A Result of this Release

|  |  |
| --- | --- |
| ***Document Number*** | ***Document Name*** |
| *N/A* | *N/A* |

## Approver List for this Revision

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Name*** | ***Function*** | ***Location*** | ***Date*** | ***Approval Method*** |
| *Sumina Verma* | *SW Eng*  *Ops* | *Sunnyvale* | *7/29/13* | *Email* |
| *Ramesh R.N.* | *SW Eng* | *Bangalore* | *7/29/13* | *Email* |