**TESTING DOCUMENTATION FOR LINUX NETWORK PACKET STATISTICS DISPLAY PROJECT**

**Contents**

[Overview 3](#_Toc167976550)

Test Plan....................................................................................................................3

[Testing Approach 3](#_Toc167976556)

Test Cases..................................................................................................................4

Test Execution...........................................................................................................5

[Conclusion 6](#_Toc167976557)

# **Overview**

This documentation provides an overview of a multi-threaded application designed to capture, analyze, and display packet statistics using shared memory. The application includes two main threads: one for capturing and analyzing packet data and another for displaying the statistics. The synchronization between these threads is managed through shared memory.

**Test Plan**

**Objectives:**

· Verify the application handles multiple operations concurrently.

· Ensure the shared memory is correctly updated and accessed.

· Validate the application's performance and reliability with typical and edge case scenarios.

**Testing Approach:**

* **Define Test Cases**: Detailed test cases for the server and client based on the requirements.
* **Prepare Test Data**: Create sample data that covers all possible scenarios including edge cases.
* **Execute Tests**: Run the tests and record the results.
* **Bug Reporting**: Document any defects found during testing.
* **Regression Testing**: Re-test after bug fixes to ensure issues are resolved without introducing new problems.

**Test Case**

**Server-Client Connection**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case # | Description | Input | Expected Result | Status (P/F) |
| TC1 | Start the server and verify shared memory creation | Server start | Shared memory created successfully | P |
| TC2 | Connect display client to serve | Client start | Client reads initial statistics | P |
| TC3 | Server runs and client connects, then disconnects | Client start and stop | Client reads and stops without affecting server | P |

**Display Formats**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case # | Description | Input | Expected Result | Status (P/F) |
| DF1 | Display in tabular format | ./program tabular | Correct tabular display of packet statistics |  |
| DF2 | Display in graphical format | ./program graph | Correct graphical display of packet statistics |  |

**System Testing**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case # | Description | Input | Expected Result | Status (P/F) |
| ST1 | End-to-end test with typical data | Standard packet sizes | Correct data generation, storage, and display | P |
| ST2 | End-to-end test with edge cases | Maximum packet sizes | System handles edge cases without errors | P |
| ST3 | Performance test with large data set | High volume of packets | System performs efficiently without errors | P |

**Test Execution**

· **Setup Test Environment**: Ensure shared memory and semaphore mechanisms are operational.

· **Run Test Cases**:Follow the outlined test cases, recording results and noting any discrepancies.

· **Document Defects**:Capture any issues, providing detailed reproduction steps and severity.

· **Review and Fix**: Collaborate with development for resolution, re-testing fixed issues.

· **Final Validation**: Ensure no new issues post-fix, confirming system stability and readiness.

### **Conclusion**

The testing approach, consisting of defining detailed test cases, preparing comprehensive test data, executing tests meticulously, and documenting any defects, provides a robust framework for quality assurance. The inclusion of regression testing ensures that any issues found are resolved without introducing new problems, thus maintaining system stability.