**Project Proposal**

**Concurrent Non-Blocking Priority Queue Implementation**

**CS-550 Advanced Operating System**

**September 5th, 2017**

**Bhagyashree Bagwe (A20399761)**

**Pradyot ()**

**Nilesh ()**

1. **Introduction**

A priority queue is a data structure for maintaining a collection of items each with an associated key (or priority)[5].Some places where a priority queue is often needed arethe process managers of operating systems, network bandwidth management and sorting algorithms. One common necessity of queues is to be able to achieve non-blocking synchronization in case multiple threads try to insert data into queue at the same time, so that no thread will be blocked for “waiting period”.

1. **BackgroundInformation**
2. **Problem Statement**

With the fast pacing world and emerging technologies it becomes human and technology necessity to have access and availability to those with the higher priority. Though the currently available applications are capable enough to provide the services to those which are based on First Come First Serve (FCFS) policy but are subsequently failing to mitigate the exigencies needed for higher preference demanded. Widely available parallel applications based on the thought of Concurrent FIFO queues failing to serve clients/applications that need an immediate attention. This need leads to increase in indefinite idle time for waiting clients and makes the system slower due to flooding their requests.

1. **Related Work**
2. **Proposed Solution**

Currently in Java8 the available implementations of priority queues are PriorityQueueclass and a thread-safe PriorityBlockingQueue class. However, even though it is thread safe, it is not efficient as it blocks the concurrent threads for synchronization which may lead to starvation of some threads indefinitely. On the other hand, Java8 has ConcurrentLinkedQueue class which implements thread-safe non-blocking FIFO queue. Their implementation is based on algorithm as described in ‘Simple, Fast, and Practical Non-Blocking and Blocking Concurrent Queue Algorithms’ by Maged M. Michael and Michael L. Scott. In this project we are trying to implement same algorithm in Java8 to achieve concurrent and non-blocking implementation of Priority Queue data structure.

1. **Evaluation**
2. **Conclusion**

The Priority queues are widely used in the distributed and parallel applications including bandwidth management to discrete event simulation. We came with the idea of using priorities for non blocking data algorithms. And this priority based non blocking data structures application is going to focus more onto the scheduling, sharing of the resources over the distributed network. We are going to check how high priority applications, resources get executed with minimal or no idle time. It is also going to check the performance improvement as compared to available non blocking FIFO algorithms.

1. **Additional Resources**

**8.1 Timeline**

|  |  |  |
| --- | --- | --- |
| Sr. No | Task | Week |
| 1 | Requirement Gathering   * 1. Analysis of Existing system   2. Literature Survey | 1 |
| 2 | System Design  2.1 Architecture Diagram  2.2 Flow Chart | 2 |
| 3 | Algorithm Designs | 3 |
| 4 | Mid Term Report Preparation | 4 |
| 5 | Implementation of proposed solution | 5 |
| 6 | Testing and evaluation of the proposed solution | 6 |
| 7 | Validation and Modification to the proposed solution | 7 |
| 8 | IEEE Paper and final report preparation | 8 |
| 9 | Project Presentation | 9 |

**8.2 Deliverables**

* Progress/Mid term report
* IEEE Paper
* PPT Presentation
* Source Code

**9. References**

[1] http://tutorials.jenkov.com/java-concurrency/non-blocking-algorithms.html#non-blocking-concurrent-data-structures

[2] https://www.cs.rochester.edu/research/synchronization/pseudocode/queues.html

[3] http://blog.shealevy.com/2015/04/23/use-after-free-bug-in-maged-m-michael-and-michael-l-scotts-non-blocking-concurrent-queue-algorithm/

[4]https://secweb.cs.odu.edu/~zeil/cs361/web/website/Lectures/priorityQueues/pages/ar01s02.html

[5] http://pages.cs.wisc.edu/~siff/CS367/Notes/pqueues.html