Carlos Valera
Proposal

I propose research into the development of a concurrent lock-free multi-queue data-structure which operates on the producer-consumer model, and which allows multiple of each. I will attempt two approaches to the problem.

In the first, I will attempt to use multiple queues (of order equal to the maximum of number of producers and consumers) in tandem to achieve an effect similar to having one producer and consumer for each queue. A queue manager will delegate which thread gets which queue at which time, and will manage access so that FIFO ordering is preserved. Two possibilities for such a manager object are a tree which routes requests, or a circular incrementing pointer.

Problems with this approach include the possibility of differences in the speed of threads leading to race conditions which (if properly guarded against) will slow the queue down to rates slower than available by current implementations. Another possibility is that the implementation of the manager will also slow the queue down significantly.

The idea for this structure is similar to that of the quiescent stack show in "Data Structures in the Multicore Age" by Nir Shavit

The second idea approach uses two separate queues for each producers and consumers. In a manner similar to a lock-free stack, the producers can add to the producer queue without the need for locks. When a consumer attempts to remove from a queue, the head of the queue is removed from the producer space, and appended to the consumer space from which producers can remove exclusively. The queue is saved as a doubly-linked list to allow traversal in two directions.

The concurrent queue is a well-studied and practical structure in concurrent programming, and many implementations have been attempted. Notable are the implementations by Michael and Scott in the paper "Simple, Fast, and Practical Non-Blocking and Blocking Concurrent Queue Algorithms" which provides both a locking and non-locking queue, and the implementation by Scherer et al. in the paper "Scalable Synchronous Queues" which is the basis for the SyncronousQueue in Java6.