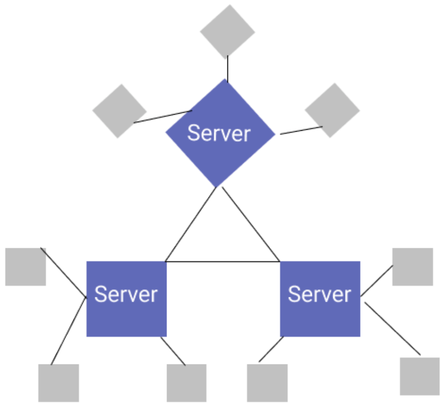
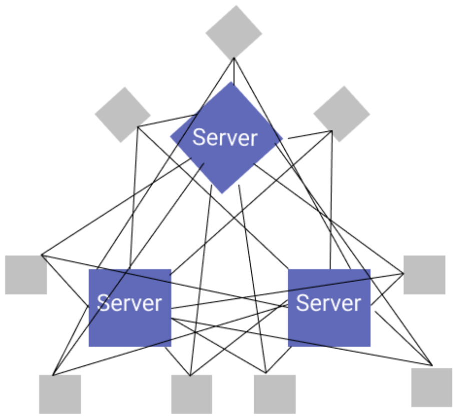
Programming Assignment 1: Distributed Hash Table
Thomas Salemy

Design / Execution Flow:
I. Network Topology

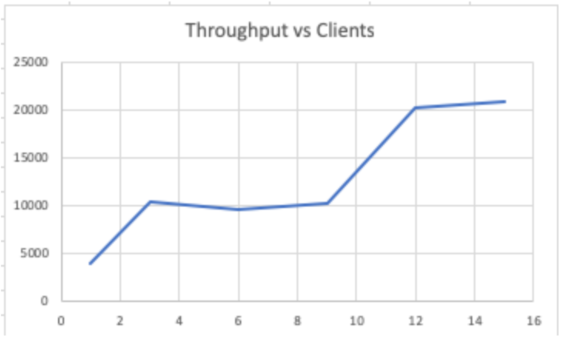


VS

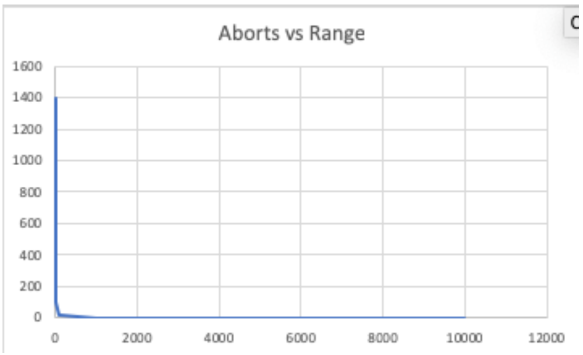


II. Develop the system module-by-module
1) Started with implementing the concurrent hash table, in which I utilized Heller's lazy list to optimistically synchronize threads in conflict
2) Thereafter, I implemented sending messages across the network and the distributed barrier through a naïve, master-oriented approach
3) Finally, I integrated each of the parts together and extended the features to handle necessary requirements, such as recording aborts

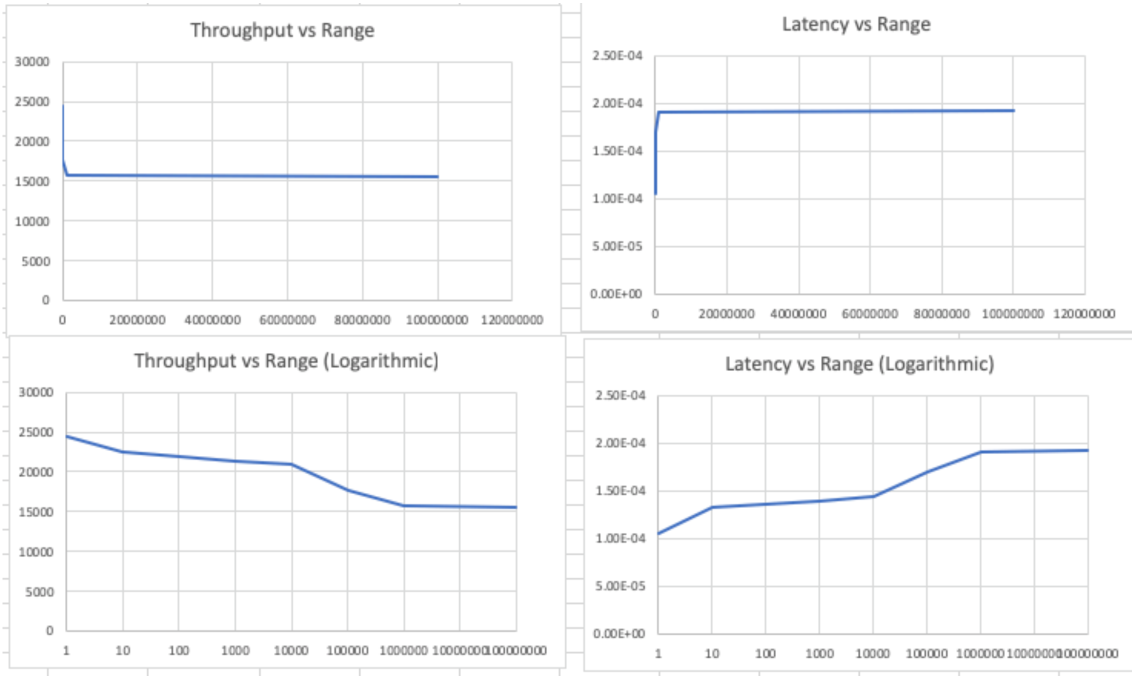
III. Benchmarks
The network acts as the most significant bottleneck in the system; as a result, anything to reduce the number of bytes necessary for a message and/or coordination will have a high impact on performance



As the number of clients increase, the throughput increases on average given the multithreaded nature of the server in which utilizes a new thread for every client



As the range increases and therefore probability of collision decreases exponentially, the number of aborts rapidly descends towards 0.



These graphs reflect two behaviors in the system: (1) the concurrent hash map loses throughput as the range increases because it does not resize and therefore must linearly probe for more elements (2) the largest bottleneck in the system remains in network communication. **It would be awesome to see how RDMA adjusted these graphs**