

# Server Cluster Load Balancing

<https://github.com/nikhilsidhu/server-network-simulation>

Nikhil Sidhu

December 14, 2022

## 1 Intro

The motivation for this project comes from the desire to optimize resource utilization in server networks. Although it is possible to make more powerful servers, at the end of the day this becomes expensive and has diminishing returns. Load balancing refers to the distribution of traffic within a network, in an attempt to improve server availability and efficiency of processing.

The set up for this project involves using 5 slow and 3 fast servers handling requests (fig.1), along with two load balancing algorithms that determine where the network traffic flows. The first algorithm is simple, directing requests to the server node with the shortest queue. The more advanced algorithm prefers to send traffic to the faster servers as much as possible to maximize efficiency.

## 2 Libraries

**Ciw** - event simulation for open queuing networks, packet-switched networks are of this kind.

**numpy** - used for calculations

**matplotlib** - used for creating graphs

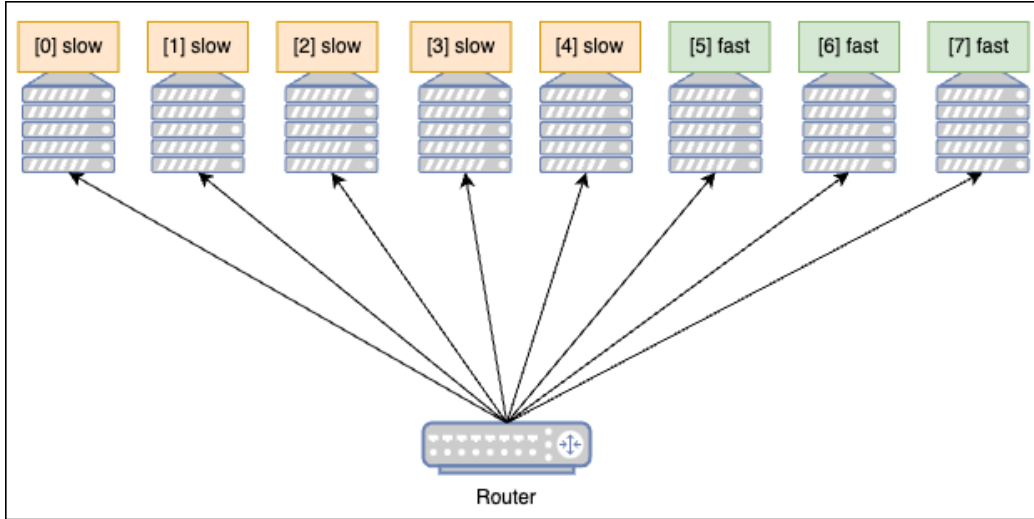


Figure 1: Layout of the server network

### 3 Instructions

Simply run the simulation with 'python3 Network.py'. Matplotlib will open the graphs for each run at the end of simulation, these can be closed to end the program.

### 4 Results

The results were as expected, the simple load balancer performed worse than the more advanced algorithm, that preferentially selects faster servers, after convergence (fig. 2).

The server utilization for the simple algorithm is high for all servers (fig.3). When the second algorithm is executed, we see that the slow servers aren't even used in the first block of time that is simulated (fig.4), as they aren't at full load. In the second iteration, the slow servers begin to be used at a slow rate (fig.5). By the last iteration, the algorithm has reached convergence and the numbers stabilize (fig.6).

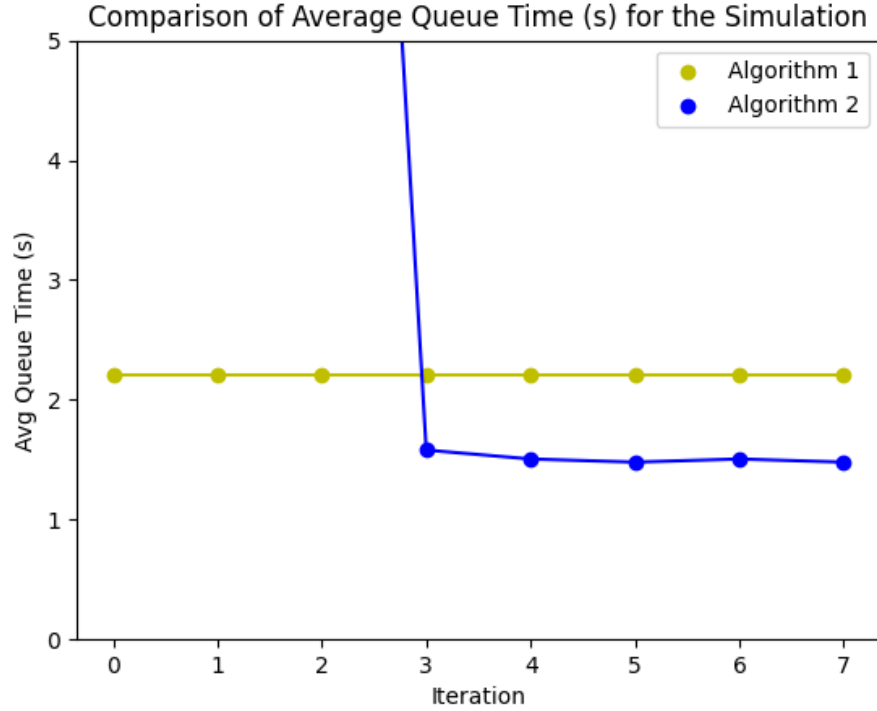


Figure 2: Average queue time in each iteration

## 5 Discussion

It would be interesting to experiment with larger server clusters and also having servers shut down randomly to simulate failure. There are much more advanced algorithms for load balancing and implementing those in such a setting could be a useful tool when paired with differently structured networks.

## 6 Papers Referenced

[https://sites.pitt.edu/~dtipper/2130/2130\\_Slides5.pdf](https://sites.pitt.edu/~dtipper/2130/2130_Slides5.pdf)

<https://ieeexplore-ieee-org.uml.idm.oclc.org/abstract/document/4403185>

<https://ieeexplore-ieee-org.uml.idm.oclc.org/abstract/document/8368317>

```
Algorithm 1:
  Queued Time: 2.00308348944659
  Completion Time: 3.3494865099745743
  Average Server Utilization: 0.8367502829098279
  Server Utilization:
    Server 2: 0.9409595585603965
    Server 3: 0.9242199648127544
    Server 4: 0.8995368088661216
    Server 5: 0.8681679131193334
    Server 6: 0.7692410397648455
    Server 7: 0.7075706610212582
    Server 8: 0.6245135966047302
```

Figure 3: Average server utilization for the simple algorithm

```
Algorithm 2: [Iteration 0]
  Queued Time: 148.5399544035108
  Completion Time: 1.866742835048046
  Average Server Utilization: 0.37436777844098734
  Server Utilization:
    Server 2: 0.0
    Server 3: 0.0
    Server 4: 0.0
    Server 5: 0.0
    Server 6: 0.9991843719279085
    Server 7: 0.9983023204491133
    Server 8: 0.9974555351508769
```

Figure 4: Server utilization in algorithm 2 iteration 0

```
Algorithm 2: [Iteration 7]
  Queued Time: 1.4795905182853348
  Completion Time: 3.1978104594903862
  Average Server Utilization: 0.7996113338423388
  Server Utilization:
    Server 2: 0.808665230251268
    Server 3: 0.7662361126810724
    Server 4: 0.7061209431260105
    Server 5: 0.6299979819703005
    Server 6: 0.9142832978303781
    Server 7: 0.878311863745042
    Server 8: 0.8351351172275467
```

Figure 5: Server utilization in algorithm 2 iteration 7