ECE1747: Assignment 1

Yida Wang 996528538 yida.wang@mail.utoronto.ca

1. Introduction

This report presents the implementation and evaluation of Lightest, one of the dynamic load balancing algorithms introduced in *Locality Aware Dynamic Load Management for Massively Multiplayer Games*.

2. Setup

- **How to turn on / off active quest from code?**
  A switch is added in the Stage 2 of the main loop in WorldUpdateModule. Use this switch to turn on / off active quest to achieve 2 scenarios: a) there is no active quest and the players are moving more or less randomly, b) there is an active quest and as a result all the players are concentrated in one area in the map. This is a compile-time switch. Recompilation is needed.
- **Which server configuration file to use?**
  Use config_demo.ini for "Static" scheme and use config_lightest.ini for "Lightest" scheme. 4 threads are used for this report.
- **How to run multiple clients at the same time?**
  Use run_clients.sh in the top-level directory and provide server name, server port and number of clients as parameters. In this report, 1000 clients are used for evaluation purpose, which provides necessary workload for the server. It is recommended to run this script on a different machine from the server machine to avoid any performance interference and emulate a more realistic network environment.
- **How to collect logs?**
  All lines of logs related to performance analysis such as number of requests, request processing time, number of updates and update sending time start with "=+=+ LOGGING". Grep for it. All the logs for this report are under report directory. More than 10200 iterations of logs are collected. Our analysis focuses on iterations from 200 to 10199 since clients are joining in the first 200 iterations and they need to be ignored.
- **How to analyze the logs?**
  Use report/log_analyzer.py to parse the logs. This Python script generates 16 plots in which each colour represents 1 single thread and calculates averages of metrics for the performance analysis. Log paths in the beginning of the Python script need to be adjusted to accommodate to the location of actual files. Python library matplotlib is required. All the plots for this report are under report directory.

3. "Lightest" Algorithm Implementation

The algorithm is implemented in WorldMap::balance_lightest(). Lightest is a dynamic algorithm that attempts to optimize the cost of remapping by prioritizing shedding load to a single thread instead of several threads. An overloaded thread tries to shed load directly to the lightest loaded thread known. The precondition is that this thread's load has to be below light_level (i.e. 1.0) and the overloaded thread's load has to be above overloaded_level (i.e. 1.2):

light_load < number of players / number of threads * 1.0

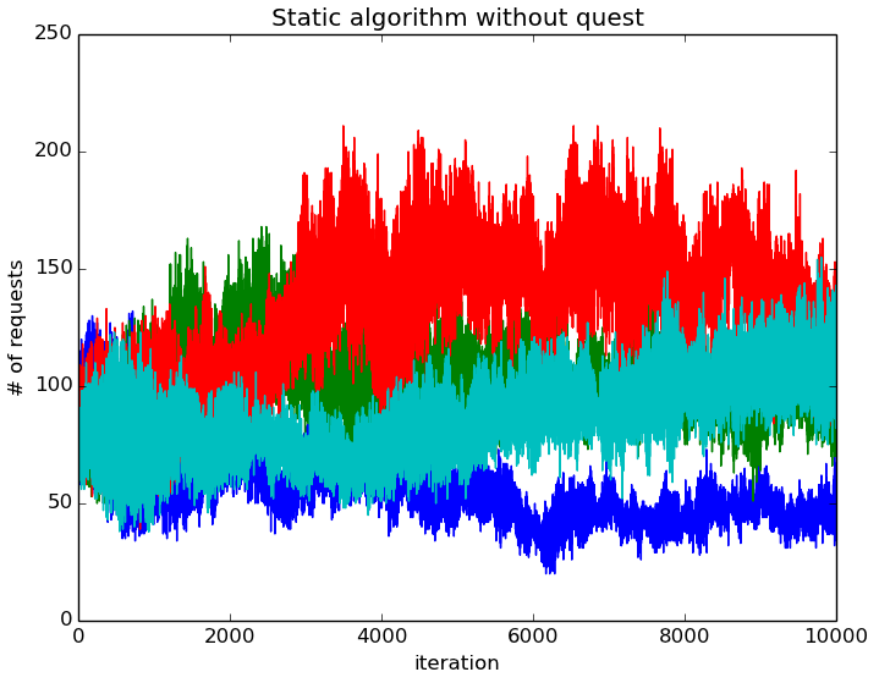
heavy_load > number of players / number of threads * 1.2

safe_level = (light_level + overloaded_level) / 2 = (1.0 + 1.2) / 2 = 1.1

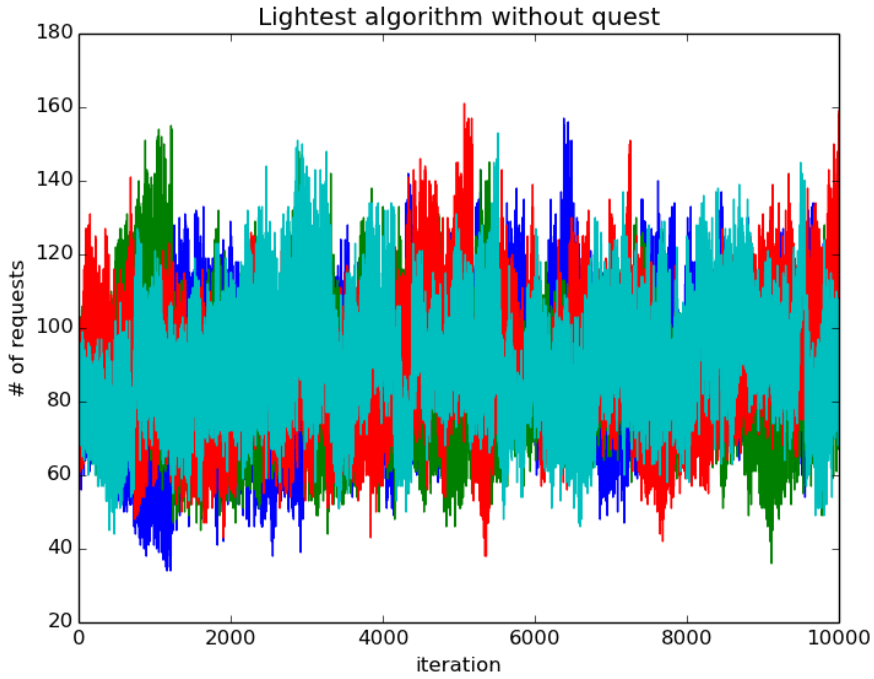
"Lightest" attempts to keep region clusters together by scanning adjacent regions in sequence. If a region cannot be placed in the bin without exceeding the safe load threshold, a subsequent region is selected, hence sacrificing on region locality. Regions of the overloaded thread are scanned in-order and assigned to the lightly loaded thread if it doesn't exceed the safe load threshold.

4. Performance Analysis

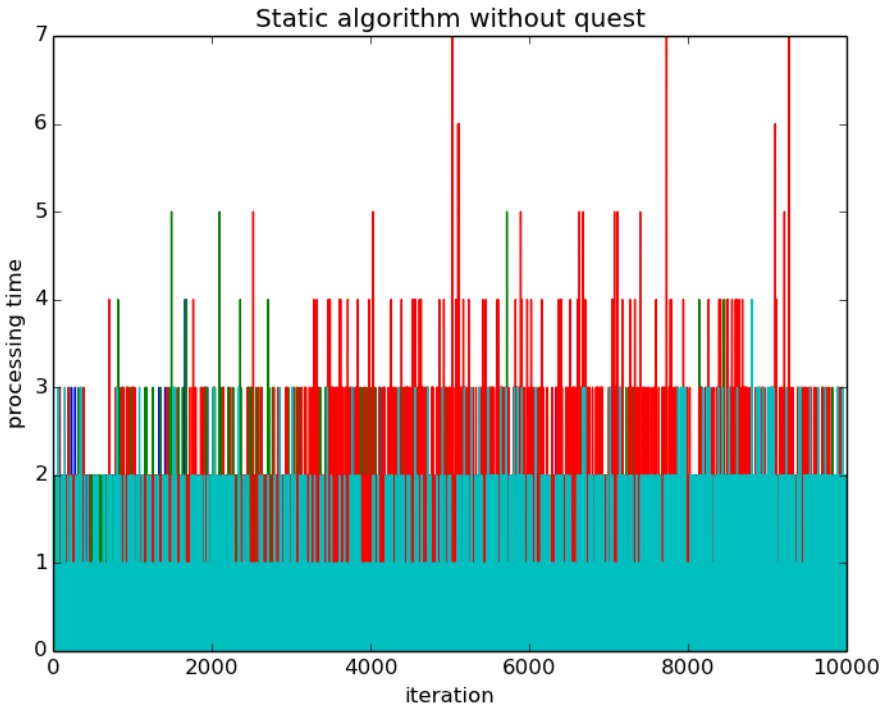
4.1 Scenario A: no active quest



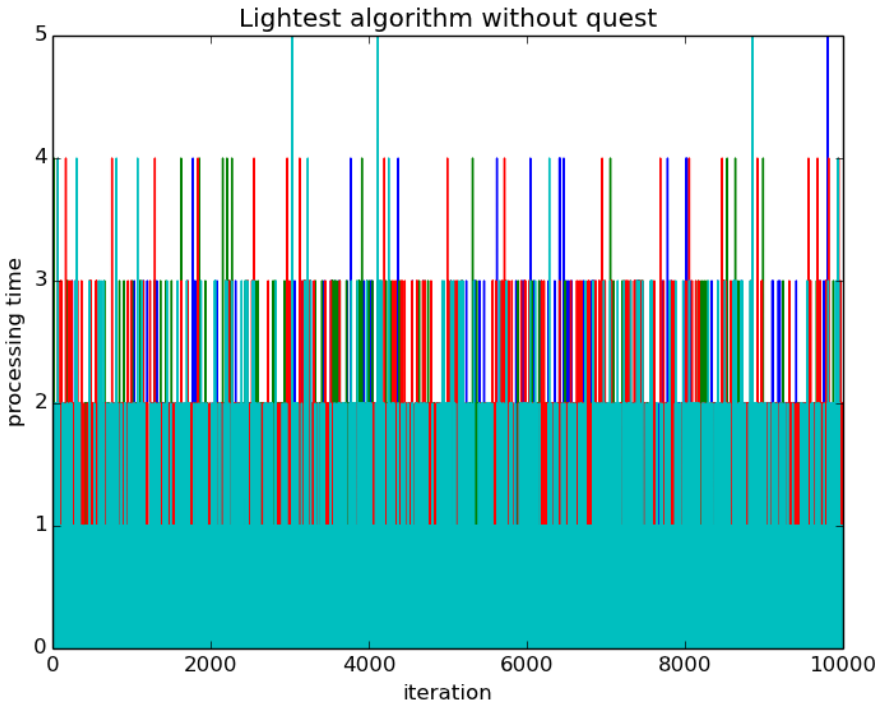
Avg: t0: **57** t1: **105** t2: **132** t4: **87**



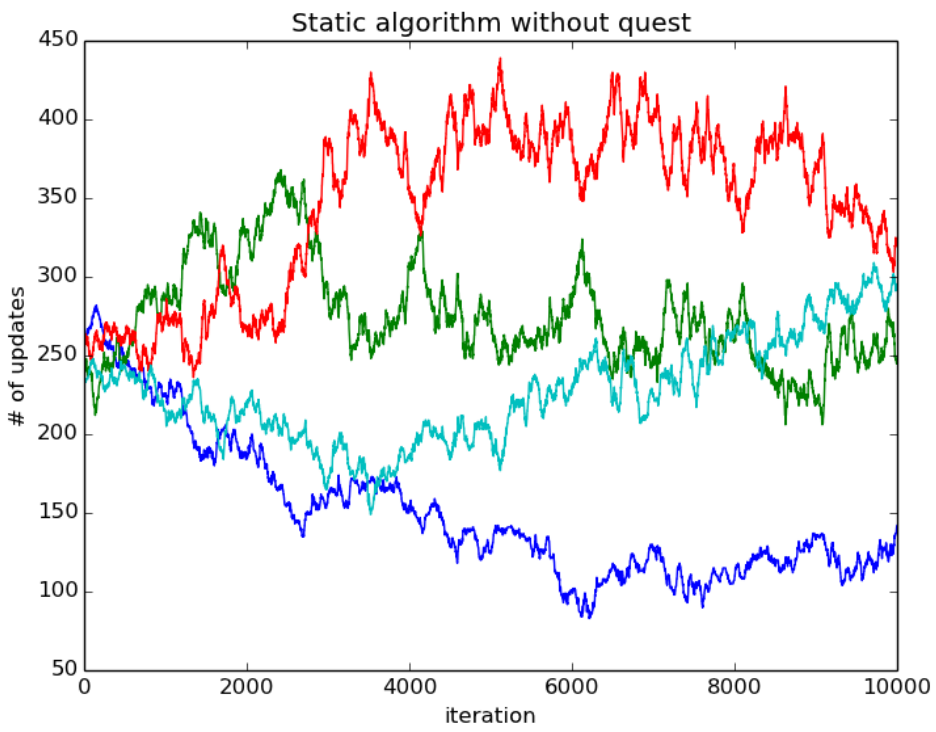
Avg: t0: **88** t1: **86** t2: **90** t4: **91**



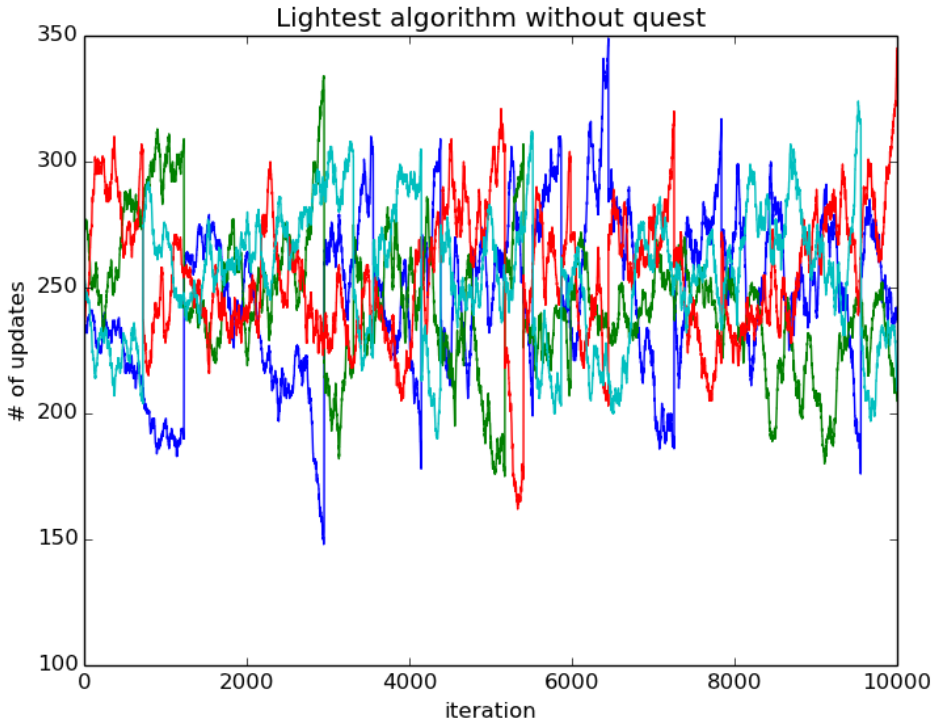
Avg: t0: **0.20** t1: **0.49** t2: **0.74** t4: **0.38**



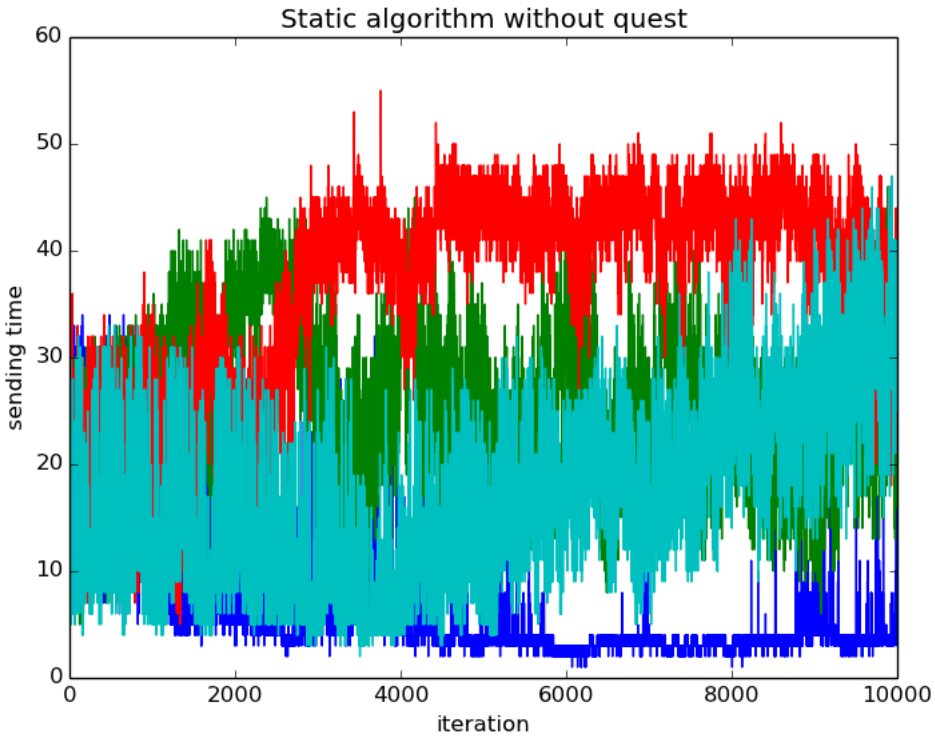
Avg: t0: **0.37** t1: **0.42** t2: **0.49** t4: **0.41**



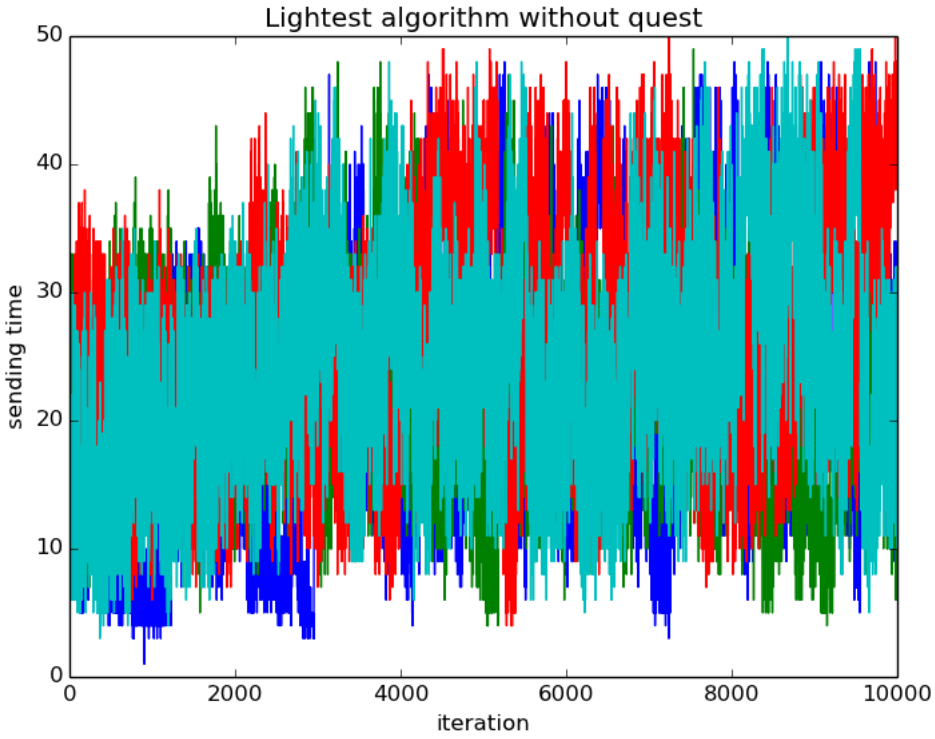
Avg: t0: **150** t1: **275** t2: **348** t4: **227**



Avg: t0: **247** t1: **243** t2: **255** t4: **255**



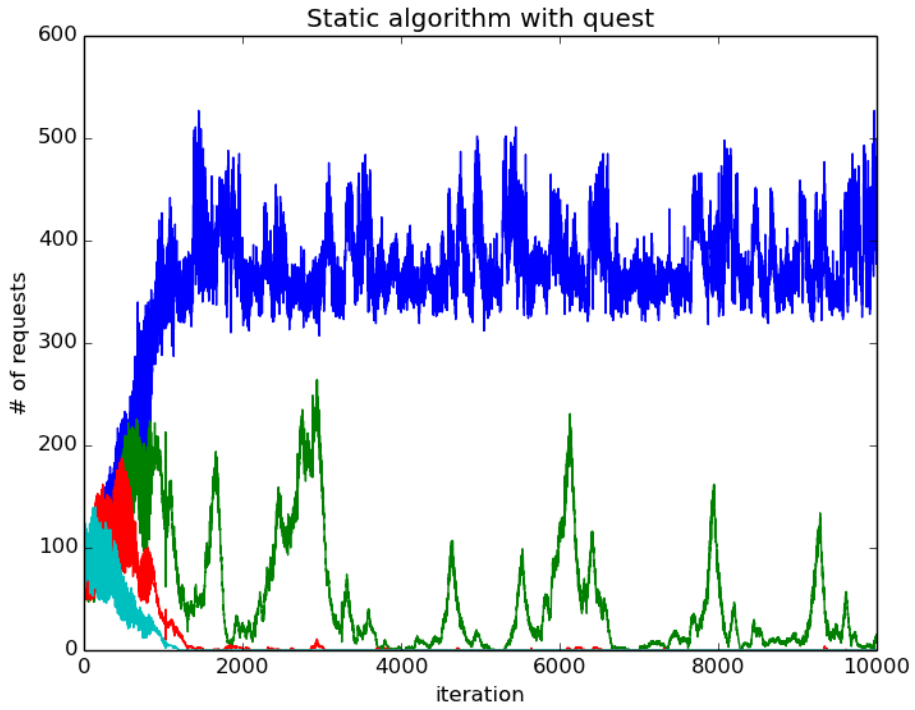
Avg: t0: **6.28** t1: **26.23** t2: **36.39** t4: **16.30**



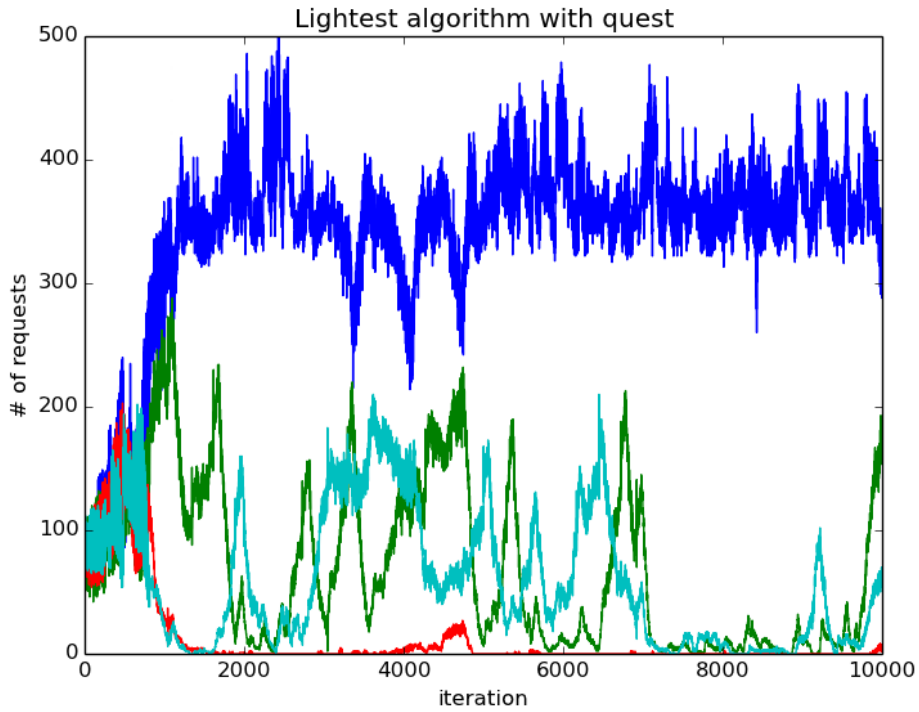
Avg: t0: **20.77** t1: **21.81** t2: **25.64** t4: **23.09**

Request processing time and update sending time are both expressed in millisecond (ms). We can clearly see from the plots as well as the average numbers that our Lightest algorithm does a pretty good job in balancing the load under this scenario where there is no active quest and the players are moving more and less randomly. As a result, processing time and update sending time are saved tremendously: speedup in processing on average is 0.74 / 0.49 = 1.61 and speedup in sending on average is 36.39 / 25.64 = 1.42.

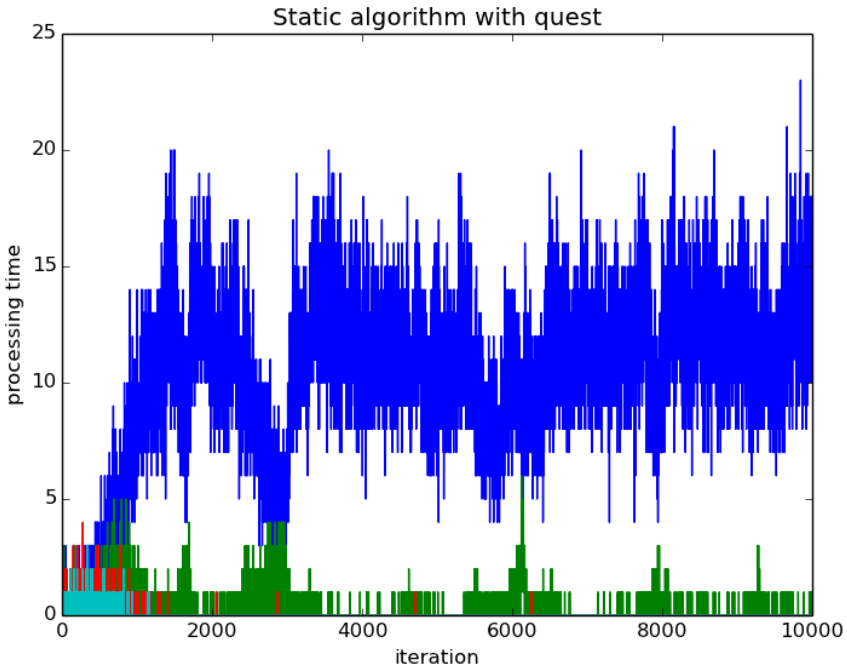
4.2 Scenario B: active quest



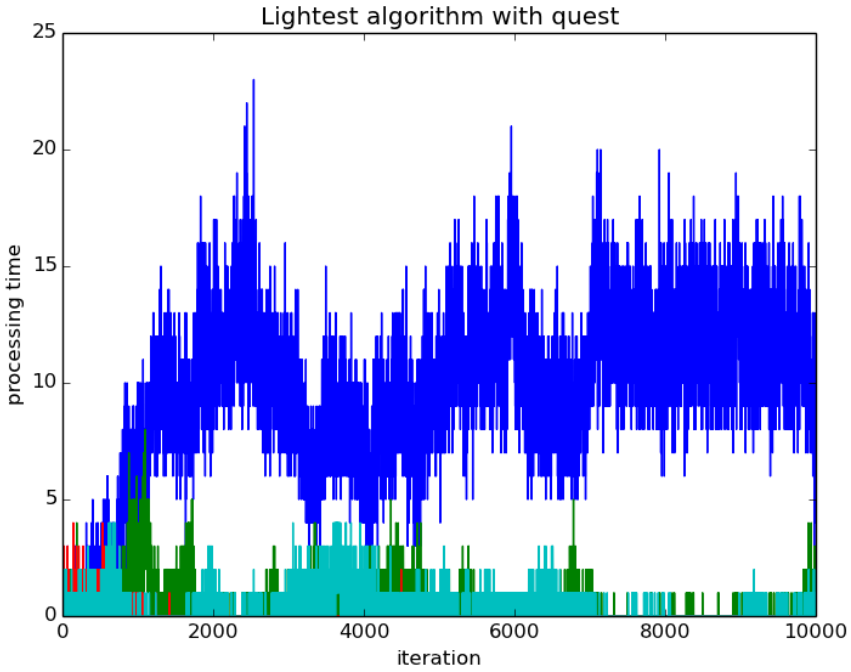
Avg: t0: **359** t1: **52** t2: **10** t4: **5**



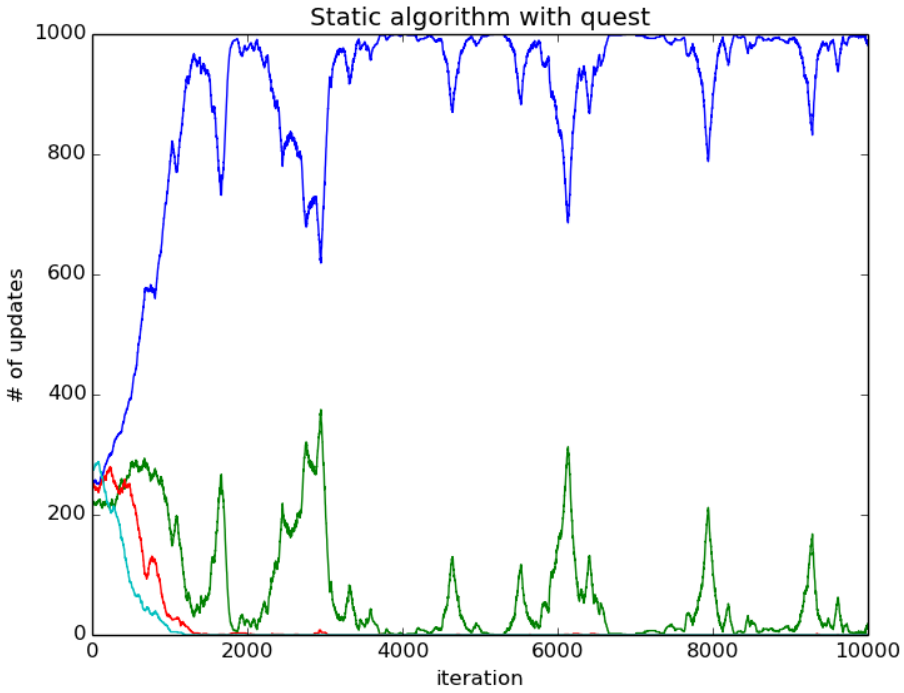
Avg: t0: **340** t1: **69** t2: **12** t4: **59**



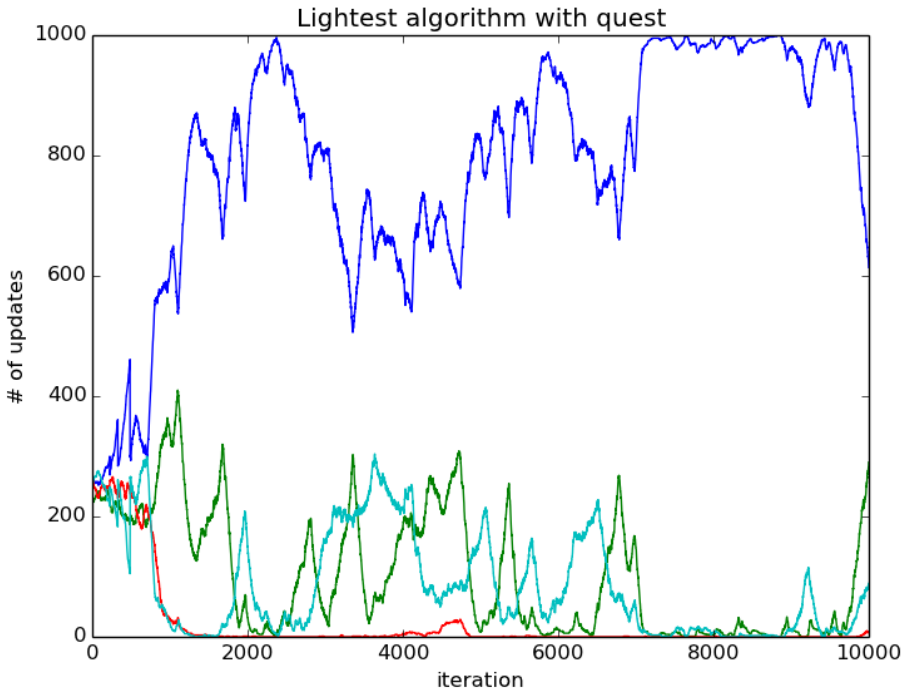
Avg: t0: **10.23** t1: **0.32** t2: **0.04** t4: **0.02**



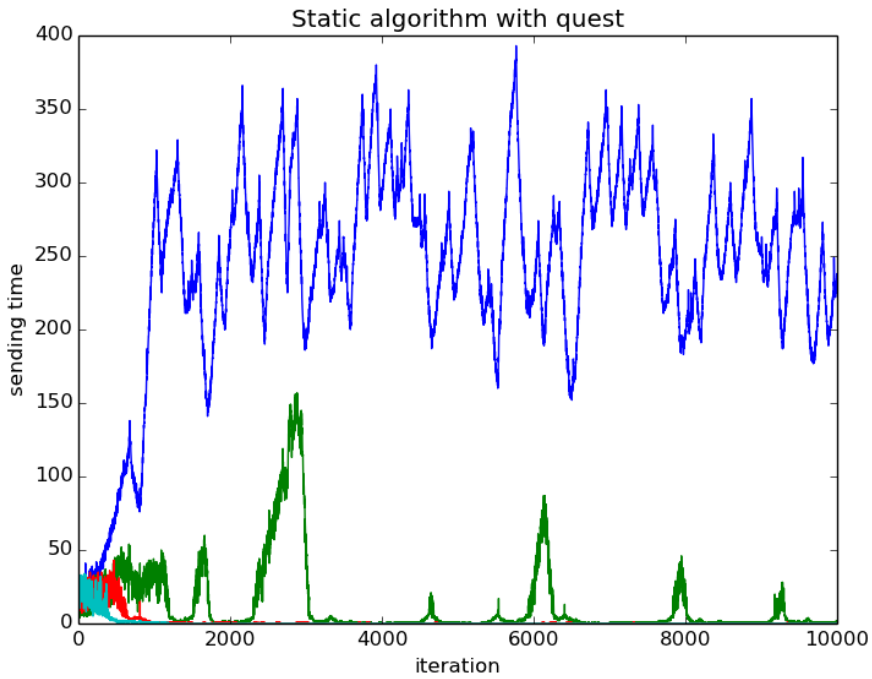
Avg: t0: **9.55** t1: **0.45** t2: **0.05** t4: **0.37**



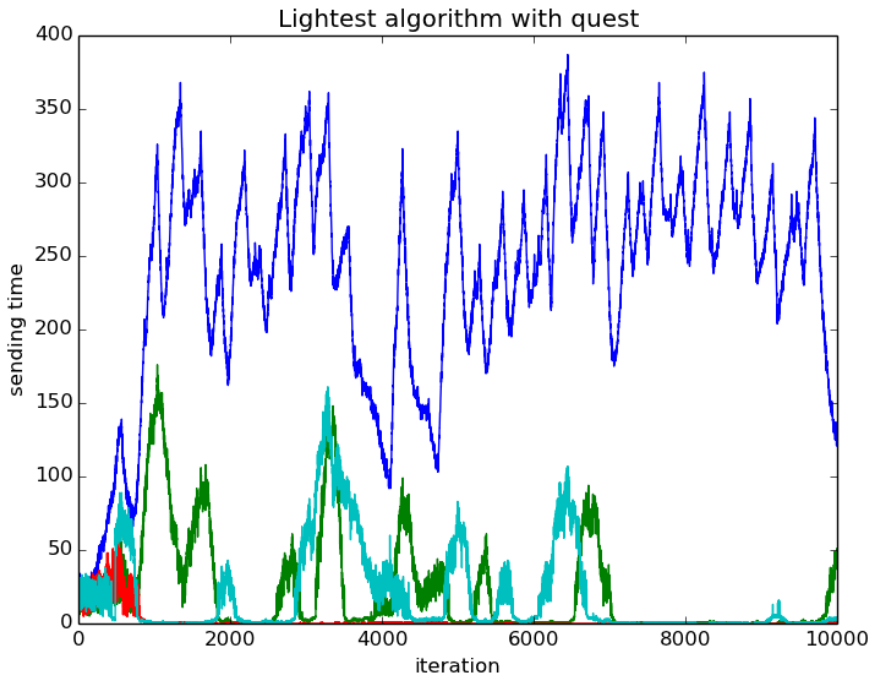
Avg: t0: **893** t1: **75** t2: **19** t4: **13**



Avg: t0: **793** t1: **100** t2: **21** t4: **86**



Avg: t0: **242.77** t1: **12.78** t2: **1.38** t4: **0.67**



Avg: t0: **236.77** t1: **21.68** t2: **1.87** t4: **18.76**

Request processing time and update sending time are both expressed in millisecond (ms). Neither algorithms handle well in this scenario where there is an active quest and as a result all the players are concentrated in one area on the map. It is observed that players tend to flock into one single region and no matter what algorithm we use, that region has to be assigned to an individual thread and thus the load cannot be balanced. To mitigate this issue, maps need to be further divided into more micro-regions and therefore we can have finer grain parallelization.

With that said, “Lightest” still does slightly better than “Static”. We can clearly see from the plots as well as the average numbers that there’s a dominant thread in both algorithms. Aside from the dominant thread, “Lightest” has 2 other threads being active while “Static” only has 1 other thread being active. As a result, processing time and update sending time are slightly saved: speedup in processing on average is 10.23 / 9.55 = 1.07 and speedup in sending on average is 242.77 / 236.77 = 1.04.