Distributed Algorithms: Assignment 3 C

Christos Froussios (4322754) & Quinten Stokkink (4016270)

Monday 2nd June, 2014

1. Introduction

In this report we present the results of our implemenation of Afek & Gafni's algorithm for election in a completely connected network. We use a slightly modified version of the algorithm to avoid drowning single processes in so many messages they slow down the whole algorithm. This involves selecting from the untraversed link pool in a random order and a timeout between trying to capture the same node successively. Furthermore we demote candidate processes that have been killed and have no outstanding messages to just an ordinary process.

2. Methodology

To test our algorithm we simulated a fully distributed environment by giving each process its own JVM environment. This is quite costly for the simulating machine, but also far more accurate than testing between two or more physical machines as every process has access to the exact same capabilities (and delays). In fact through our tests we found that as the memory footprint nears 4 GB and the amount of threads gets close to 10,000 our simulating machine starts locking up.

As for our testing methodology we decided to always test the worst case, which occurs when all processes present themselves as candidate processes and start the algorithm. By testing the worst cases we should see a curve of the runtime that is in the order of $O(nlog(n))$ as advertised by the algorithm. We also simulate message delay with a normal distribution with mean 15 ms and standard deviation of 5 ms.

3. Results

We have run tests for the amounts of processes (which are both candidates and ordinary processes at the same time) of: 1, 10, 20, 30, 50, 80, 100, 130 and 150. For each of these processes we then reported the level it reached and whether it was elected (to verify correctness). For each of these experiments we also recorded the total time.

The results of the running time experiments are visualized in Figure 1 and the measurements of the maximum levels are visualized in Figure 2. We see that the running time does seem to follow an $O(nlog(n))$ curve and that the maximum levels reached seem to follow a linear curve. Furthermore, Figure 3 visualizes the total amount of captures performed in the algorithm versus the amount of processes. While this series seems to follow a linear curve it suffers from a lot more noise than the other two metrics.

Note that due to our optimization, candidates are only killed once. Therefore it does not make much sense to report on this.

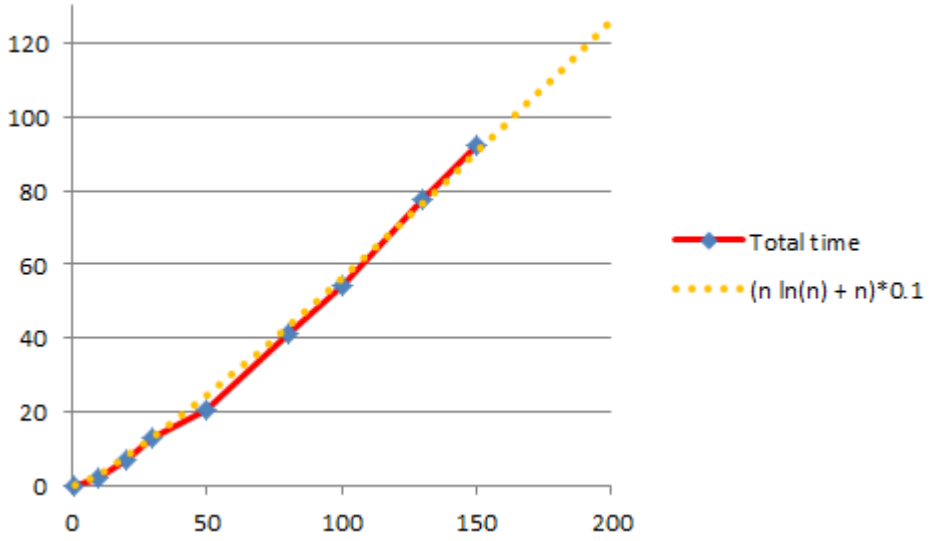

Figure 1: The total time taken for the algorithm to terminate versus the amount of processes

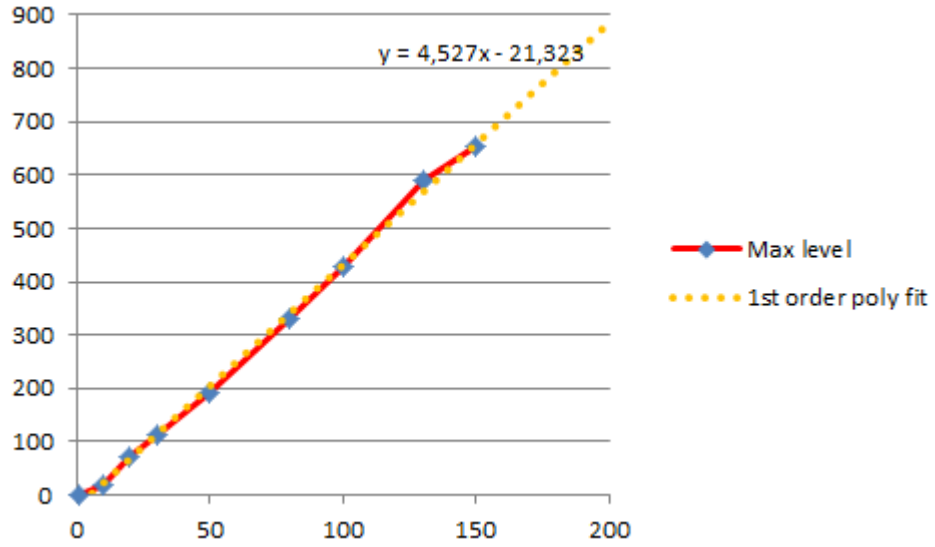

Figure 2: The maximum level achieved versus the amount of processes

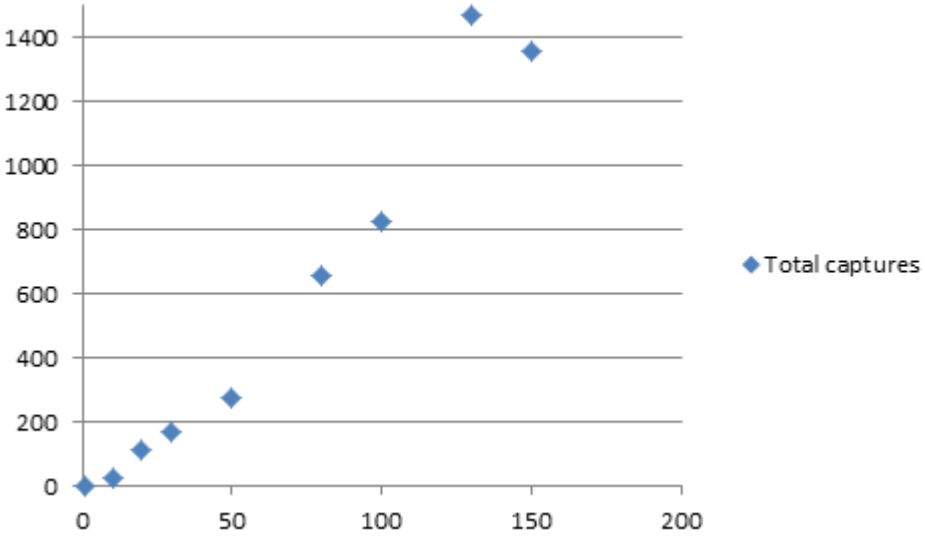

Figure 3: The total amount of captures in the algorithm versus the amount of processes