Jared Barneck

CS 6210

Dr. Stephen W. Clyde

# Homework 4 Report

This report covers the architecture, development, and testing, and observations of running an election algorithm.

## Approach and Design Choices

The approach is outlined in the Election Architecture.vsdx document included alongside this report.

### Folder Structure

A key component of the design is the folder structure, and using the folders as buckets.

* Master
* Sleep
* WorkCompleted
* WorkInProgress
* WorkQueue
* WorkTerminated

This design uses the folders as buckets and uses serialized files inside the folders to manage the work.

## Election via Folder and Files

The election process is use the bully approach. The highest IP and Port combination is the master. The files are named after their IP and port.

If the current master doesn’t respond, another worker will take over. It will move the master’s file to the sleep directory.

### Resource Management

All the workers write a file to the Master directory named with their Ip and Port. So there should be no conflicts.

All the other folders and files are only managed by the Resource Manager, which is the master.

## Observations

There were many learning and observation experience while designing, developing, and running this solutions.

## Keeping process running

When you open a port with ZeroMQ, the process basically never stops running. So you have to call in in a thread or background task. I used async and await for this.

Threading adds significant complexity to both development and debugging time.

Also, at times, I would like to await a call to the message service, but other times I didn’t want to await it. I needed the program to stay alive and not end while various tasks are running.

### Complex Scenarios Observations

There are many scenarios for failure that have to be handled to make this project work. The master could go down, sound simple, but there is much more to it than that.

#### Simultaneously Starting

What if two devices start at exactly the same time? No Master exists. Both think they are the master. What if *n* devices start and all *n* think they are the master. There needs to be a delay and a recheck.

What if a Master file exists for a failed Master and two devices start at the same time. They both think they are the master and they both try to move the failed master out of the queue. This was solved by writing their file first, before they start doing an election.

#### Communication – Too Silent or Too Chatty

How often should the master check on the workers? How often should the workers check on the master? If we check too often, we risk a network storm. If we don’t check often enough, election make take too long.

#### Completed Task then chosen as Master

What if a worker is finished, tries to tell the master, but the master is down, now the election occurs, and that worker, with a finished task is the master.

#### Middle of the task

The longest running task is queuing up the work. It takes a while to do 1000 files from S3. However, if this crashes, not only do I need a new election, but the new Master needs to start queueing tasks that are not yet queued.

### Log File Observation

When testing locally, putting a Log file in the Log directory was easy to do. However, once we moved to S3, you can just append a line to an S3 file easily. So at this point, having the log directory be part of the folder structure became useless.

## Test Scenarios

Setup. The server setup was fine. It went quickly.

The S3 setup, pushing 1000 files, took some time.

### Issues

I couldn’t get eight programs to run. ZeroMQ crashed and I lost RDP connection every time a few of the programs were launched.

### Test 1

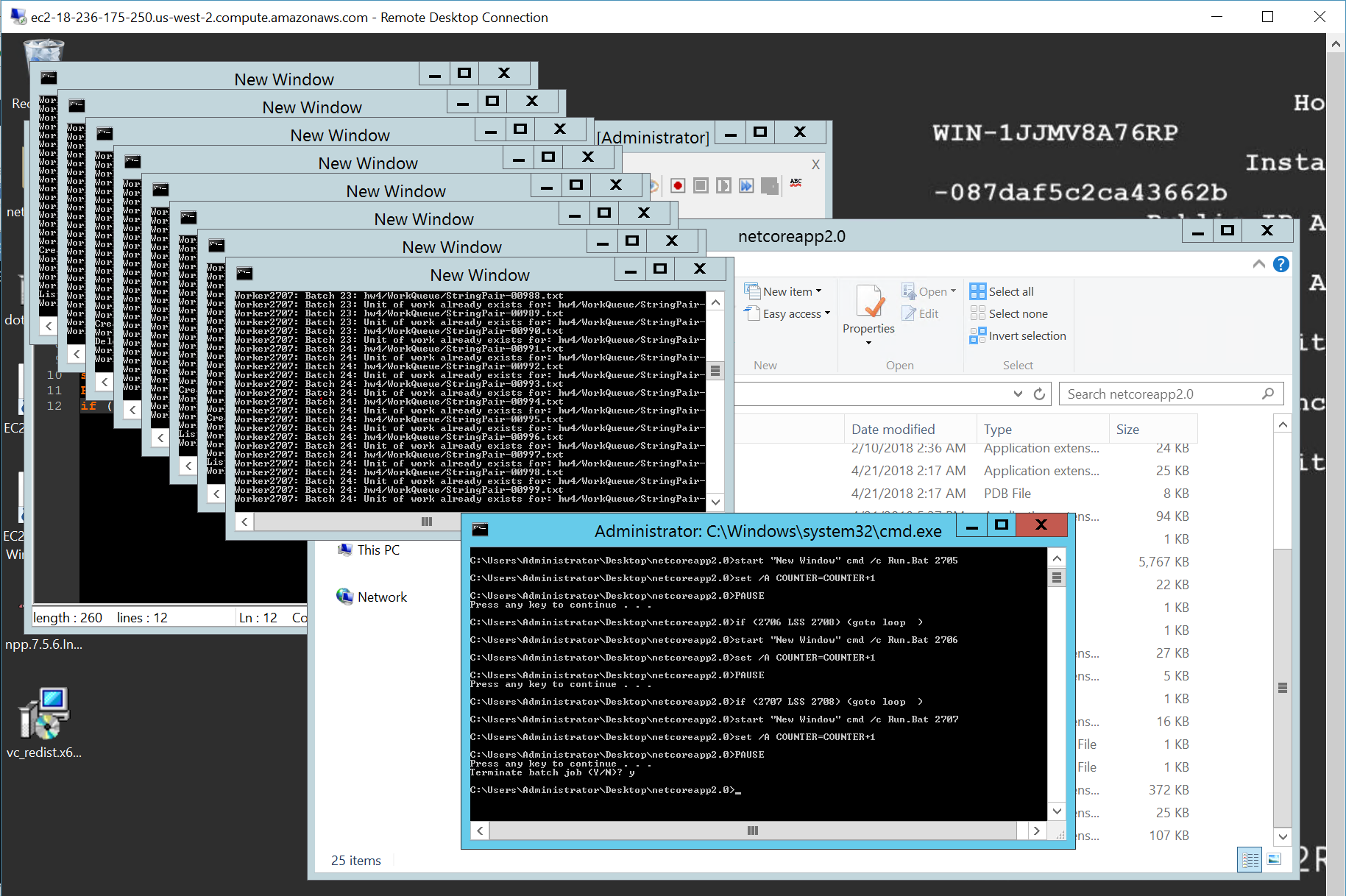
This processing was pretty slow with S3. It took the better part of an hour. I had to do some debugging and some performance debugging.

It seems that the master was bogged down, didn’t respond, and so workers kept shutting down.

Finally got it working:

Another bug is that I started them in ascending order, so each subsequent start would take over master, and then I ended with a bunch of processes not working. What happened is they were added to the workers list in the current master file. I needed to fix a bug to not only call elections to all workers included in the Master file as Workers, but also to notify all Nominated Workers that posted a file to the Master’s folder.

Here is a screenshot of 8 of them running on 1 machine. I realized it was only supposed to be four on one machine after I got it all working. I didn’t worry about redoing the observation with only 4 processes.



### Test 2

**Prep**

I had to reset all the files into place, which took significant time on S3. I could reupload, but that took a long time. It was faster to copy the files in blocks of 300 manually.

Perhaps it was a bad idea to move all the files. It sounded like a good idea, but that is what makes observations great. It would be better to leave the files in place and store the status of each task by id in a separate repository, such as serialized file.

I saw a master file moved to the sleep folder. It didn’t all work out though the first time. The master takes over, but the other processes did not start using the master. I fixed some bugs and tried again.

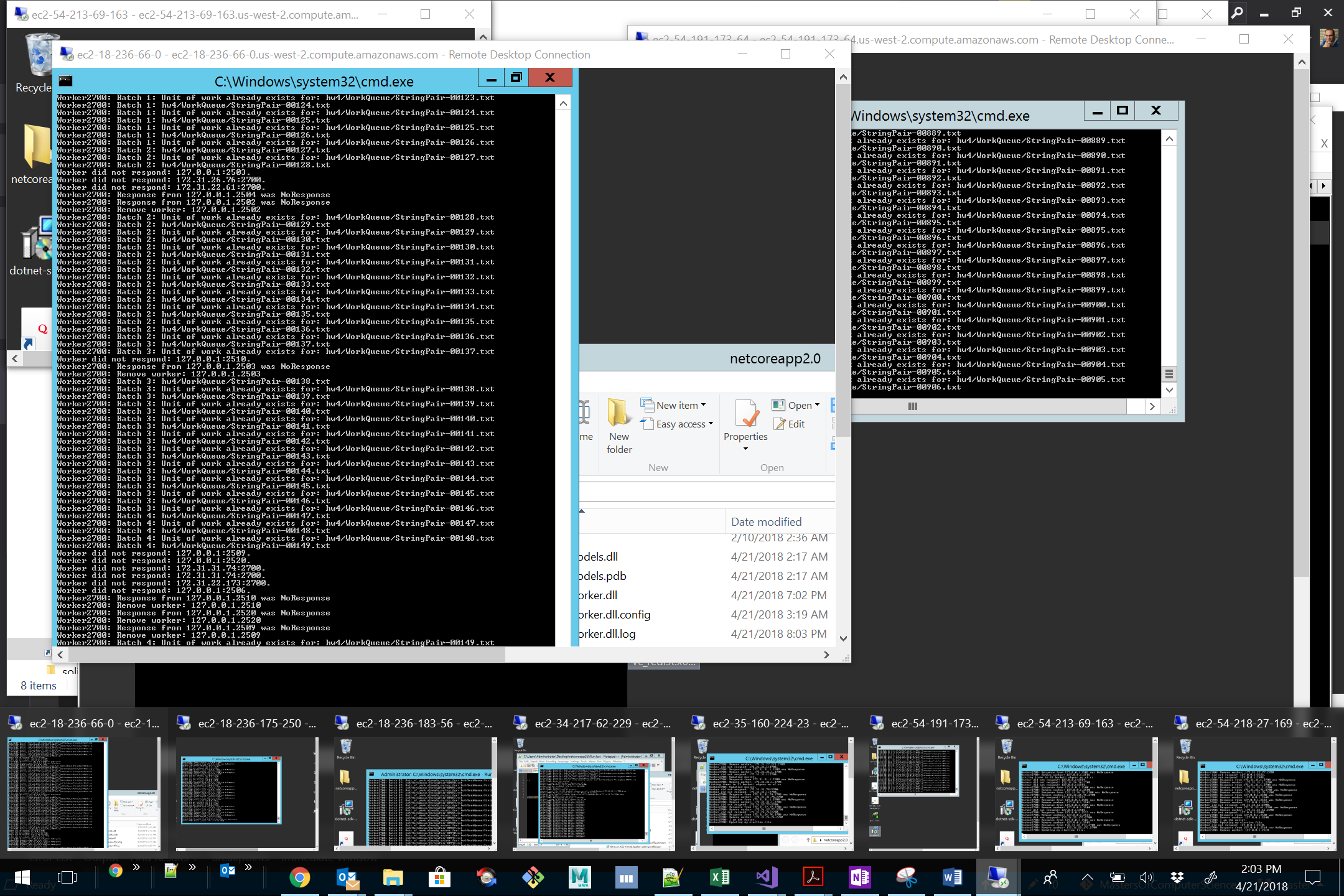
After I fixed some bugs, all the processes started working.

### Test 3

I got eight systems running. Windows is less friendly than linux, without SSH, so it isn’t as scriptable.

I setup the firewall on both the AWS security group and the Windows Advanced Firewall, yet the pings between the Workers all failed. I troubleshot this using telent <ip> <port> and their was not a connectivity problem.

Here is an image of all eight systems running.



### Test 4 – 5 – 6

Did not run. I need to troubleshoot why they are not communicating but there is no time left.

## Home Work Time

I spent approximate 48 hours, twice the estimate, on this homework assignment.

I was not able to complete the homework in 24 hours. I wasn’t able to code the homework in 8 hours. My estimated time is included in the attached Excel file.

I am not sure how much time other students put into the homework, but I was at almost 30 hours of coding before the class period where you said to not spend more than 8 hours.

I spend a lot of time costing work at my current job. My estimates before starting this assignment was 65 hours of development time. I skimped on unit tests to get that time down. I worked day and night when not at work or at school.

I am not sure if I am doing it wrong, or too complex. Perhaps my code is overly complex. Perhaps over the years, I have lost the ability to minimize my code.