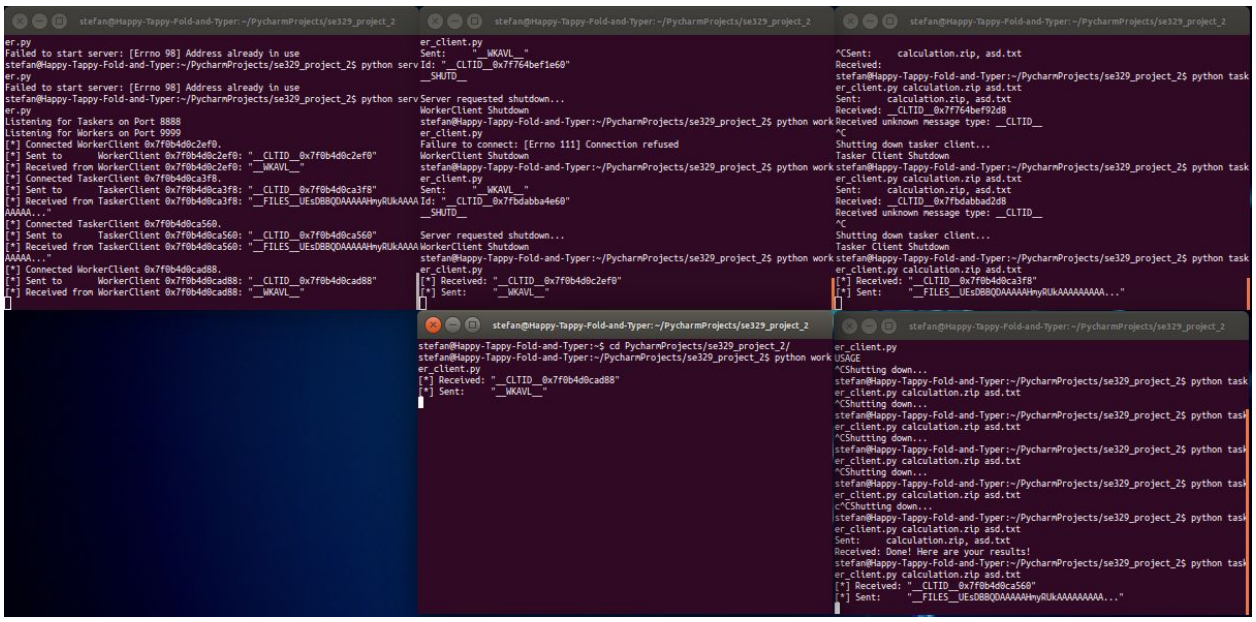
Project 2 Retrospective Report

International Justice League of Super Acquaintances

Brody Concannon
Nathan Karasch
Stefan Kraus
Gregory Steenhagen



A image showing Project Marx in action. On the far left, a running server connects to two Worker Clients (center) and two Tasker Clients (right). The Tasker Clients upload a file to the server which is passed to one of the running Worker Clients.

What we promised vs what we delivered

Minimum Viable Product (MVP)
In our proposal, we outlined the following requirements for our MVP:

1. Contains the three main parts of the project
    a. Tasker Client
        i. Provides a way to upload Python 2.7 code for execution
        ii. Provides a way to receive results from remote runs of the uploaded code
    b. Server
        i. Provides a way for Tasker Clients to connect and upload Python 2.7 code
        ii. Divides the code into chunks for distribution. **(Not Integrated)**
        iii. Connects to Worker Clients to send them Python 2.7 code chunks for execution
        iv. Collects results from the Worker Clients and sends the results back to the correct Tasker Clients **(Not Implemented)**
    c. Worker Client
        i. Provides a way to receive tasks (Python 2.7 code chunks) from the Server for execution
        ii. Executes downloaded tasks and records logs **(Not Implemented)**
        iii. Sends logs and results back to the server **(Not integrated)**
2. Has a Command Line Interface for each discrete portion

Post-Mortem
Although most parts of the system have been completed, they were not all integrated into the system. This includes the “Chunking” portion of the MVP: Code is not successfully split into chunks and sent for execution, and consequently the logic for re-assembling the results was unused in the final product. We also failed to have a successful run of downloaded python code on the Worker Clients.

Extensions
None of the proposed extensions have been completed for this project.

What went well

Planning and Setup

We put in a great effort to get set up early and put more thought into planning as compared to the first project. We were all able to get a development environment working and everyone was able to contribute to the project early, which was our most detrimental issue in the first project. In just the first few days, we already had the basics of the main server and clients working and sending rudimentary messages between themselves. The code also benefited from being slightly more organized than the code in the first project, which was another one of our main goals for the development portion of this project.

Areas we improved

Environment Setup
We were able to set up the environment very quickly and in the first few days had contributions from everyone on the team. This is a great improvement over the first project, and can be partially attributed to the fact that the issues with external tool setup were simply not a factor.

Reaching Out for Help
Communication was significantly improved from the first project, and we were able to work together in order to tackle issues together. This comes with the understanding that for small or short term projects such as this one, there isn't really a reason to have "owners" of particular parts of the codebase, and it helps development greatly if everyone has a good understanding of all of the parts as well as the overarching system. This ties into our team maturing and being able to work together more effectively.

Clarification of Requirements Early
On the first day of work on the project, we made sure to clarify and fully describe the requirements for this project and what we wanted the overall structure to look like. This helped keep the development moving forward, and although the aforementioned unforeseen roadblocks slowed this progress, we had less reworking or restructuring to do.

Productiveness of Meetings
Our formal meetings were much more focused and organized than those in the first project, and it definitely helped keep everyone on track for the project. We made sure to define meeting goals before the meeting and had our own standups. This semi-structured approach to meetings is working very well and promotes teamwork and collaboration between members.

Areas we can improve

Avoiding Procrastination
Much of the work in this project has been backloaded, which puts a lot of pressure on the group and causes frustration. This is not conducive to group cohesion and avoidance should be prioritized over most other aspects. When working in a group and having to depend on others, having a part delivered after when it is expected causes significant backups for the whole group. In the next project, we will be putting more effort towards creating and managing small tasks, and keep in mind consistent integration of these parts in order to quickly construct the minimum product.

Time Boxing
There were unforeseen issues with individual parts of the project that extended the amount of time needed to implement what we thought would be simple parts of the project. For example, the Threaded Socket Server needed a decent amount of shutdown logic in order to free sockets and properly exit. Furthermore, the zip file utility had some early issues as well. This led to group members spending a great deal more time on these parts of the project that we did not even consider as possible issues.

Readability of Code and Coding Standards
Although it is always preferred to have the cleanest code possible, it is very hard to employ or enforce this on short-term projects, and is not the priority for the Minimum Viable Product. For these reasons, we have decided to try to keep good coding in mind and adhere to it when we can, but prioritize getting something that works rather than something that looks good. On this project our code was not too bad, so this aspect was not considered to have been done poorly.

Foreseeing External Time Commitments and Mitigation
During the past two weeks, our group members have had to dedicate significant time that they originally planned to dedicate to the project. Some of these were completely unable to be predicted. In this case we should try to keep work delegation fluid, and not be dissuaded from asking for help in the case of being unable to complete a portion, even if it was promised or agreed upon in a meeting. In other cases the time commitments were known, and eventually turned into conflicts of time. For future projects, we should make an effort to anticipate how much time each group member will have to dedicate to the project. This may also help with our ability to determine how much of the project is deliverable in the time we have. Furthermore, we could also use this information to try to normalize the amount of time that each group member dedicates in order to minimize inequality in how much work each member puts in.

Things we couldn’t control

External Time Commitments
Although there are ways we can mitigate lost time to other commitments (as outlined in the “Areas we can improve” section), there is no way for us to get back the time lost without putting large strain on our group members, which would lead to many more problems.

Lessons learned

- Underpromise! The MVP should be the absolute bare minimum, the simplest use case, and the most basic example that shows the concept of the project. The remaining functionality should be fleshed out in the stretch goals.
- Communicate how much time members have to dedicate to the project beforehand and agree on the expected time commitments. This includes setting an upper limit on how much group members should be expected to work in order to prevent members from making Herculean efforts to try to deliver a project that is much too large for one or two people to tackle.
- Do not underestimate the amount of time that using or learning unfamiliar tools will take. It may be a good approach to take the amount of time for implementation of a feature with an unfamiliar tool “If it were easy” and double it in order to account for it “not being easy at all” or for special caveats.