15-440, Fall 2012
Project Assignment P3: Design Your Own Distributed System

Assigned: Nov. 9
Project Plan due: Tues., Nov. 20
Preliminary demonstration and code review: Nov. 28–29
Final demonstration and code review: Dec. 4-5
In-class demonstrations: Dec. 6

This project should be done in teams of two.

For this project, you will design, implement, and thoroughly test a distributed system, implementing some application, such as a multi-player game, a collaboration tool, or a transaction system.

Requirements

Since this is a course in distributed systems, we want it to have "interesting" features from a systems perspective. Here are some important properties your system should have:

- The system must support multiple, autonomous agents (either human or automated) contending for shared resources and performing real-time updates to some form of shared state.
- The state of the system should be distributed across multiple client or server nodes.
  - The only centralized service should be one that supports users logging on, adding or removing clients or servers, and other housekeeping tasks.
- The system should be robust
  - The system should be able to continue operation even if one of the participant nodes crashes.
  - It should be possible to recover the state of a node following a crash, so that it can resume operation.

We will let you choose your own application, and we will give you wide latitude in the overall and the detailed design of your implementation.

- Since it is difficult to predict just how hard implementing a new system will be, you should formulate as a set of "tiers," where the basic tier is something you're sure you can complete, and the additional tiers add more features, at both the application and the system level. For example, your system enhancements could include increasing the level of fault tolerance, decreasing the time to recovery, or adding features to increase scalability or security.
- You may choose any programming language. Although Go is a logical choice, there are other possibilities, including Java.
- You may use any of the packages in the Go library (or comparable packages in whatever language you choose.)
- You may *not* use any externally created distributed system packages, such as Zookeeper or Cassandra.
- You *are* allowed to use external packages to support the application or user interface. Examples include things such as GUI builders or frameworks, image/video/audio processing packages, online mapping APIs, etc.

Evaluation

Your system will be judged mainly on how it operates as a distributed system. The primary evaluation will be according to whether your system has the following attributes:

- It should be an interesting distributed system, making use of some of the algorithms we have covered in class for distributed synchronization, replication, fault tolerance and recovery, security, etc.
- The software should be well designed and well implemented, in terms of the overall architecture and the detailed realization.
- You should devise and apply systematic testing procedures, at both the unit and systems levels.
- The system should operate reliably and with good performance, even in the face of failures.

Important, but secondary considerations include:

- How interesting is the application: is it a fun game, a useful tool, or a solution to a real-world problem?
- How nice is the application's appearance: does it have a nice interface or a compelling visual display?

Some Project Ideas

The following list of projects is intended to spark your imagination. Feel free to come up with something totally different. Contact one of the teaching staff if you are uncertain of the suitability of your ideas.

- Shared document editing, in the style of Google docs. The system should support real-time editing and viewing by multiple participants. Multiple replicas would be maintained for fault tolerance. Caching and/or copy migration would be useful to minimize application response time.

- A simulated life game, in the style of The Sims or Farmville. The state of the system would be partitioned spatially, with replication for fault tolerance.
- A multi-player real-time game, based on shooting, hunting for treasure, etc. There are many possibilities here.
- An airline reservation system. Each airline would maintain its own collection of servers, with enough state replication to enable automatic fail-over. It would be possible to book travel that involves multiple airlines.
- Streaming p2p video. Warning: This is a little more challenging than it might seem. Contact binfan if you want to do this one. **THIS PROJECT IS NO LONGER SUGGESTED.**
- A low-latency notification system. E.g., watch a whole bunch of RSS feeds and send all subscribers an email when one is updated. Interface with both the raw RSS feeds and Google's update notification service. Replicate and partition the state of the monitoring system so that it can scale and survive node failures.
- Implement a distributed filesystem that does something interesting. Maybe you want one for storing your MP3s or movies. Or perhaps for something entirely different. What design constraints and optimization criteria would you have?
- A solution to a problem that really bugs you or that you would like to see solved—perhaps your new startup idea!

Logistics

At the beginning of class, Nov. 20, you should submit a project proposal. This should include:

- A description of the application.
- The overall structure of the implementation.
- The distributed system features and algorithms you intend to implement.
- Your plan for testing the system.
- How you intend to implement your system as a series of tiers.
- A schedule for how you plan to carry our your design and implementation.

This document should be around four pages long, and it will count for 25% of the total grade. We encourage you to meet with one of the course staff during the planning stages to get some feedback on your ideas prior to submitting your proposal.

On Nov. 28–29, we will schedule 30-minute time slots where you will meet with one of the course staff. You should be able to give a demonstration where some aspects of the system are working. You will go over your code, and you will discuss how you will complete the project. This review will count for 15% of the overall grade.

On Dec. 4–5, we will schedule 30-minute time slots where you will meet with one of the course staff. You will give a final demonstration and a code walk-through, including how you did your testing. This will count for 60% of the overall grade.

We will have a “demo day” in class on Dec. 8, where we will let projects (voluntarily) present about their work and let others try out the applications and learn about their implementations. The projects voted best by their classmates will receive prizes to be announced.

Good luck!