

# CS 138: Projects Overview

January 24, 2008

## Overview

Each of the five projects in CS138 depends on the projects before it. A problem in the first project can lead to even bigger problems down the road if your overall design is flawed. Because of this we want to provide you with an overview of all the projects before you start in on the first project. We're hoping that by knowing how each project will fit into the larger system you can make better design choices.

## 1 Comm

Comm is the foundation of the entire system. The Comm project defines a protocol for message passing between servers and from clients to servers. It will be multi-threaded so that you can easily simultaneously send and receive messages.

## 2 SDDB

SDDB uses Comm to build the first piece of the distributed database your system will form. Each server in this system will have a local database in which it stores key-data pairs, and these databases will implement replication to handle consistency in case of updates. You will use a simple test client to test the features of your database.

## 3 Quorum

In Quorum, you will add an important feature to your database – system level serialization. This means that all the servers will agree on the order in which updates arrived, even if two

competing updates are entered into different servers simultaneously. Servers will vote on which update to operate on first before committing any changes.

## 4 Fault

Fault is the implementation of fault tolerance. That is, if a server in the distributed system goes down, the others will notice and efficiently remove that server from the communications list so as not to waste resources on it. This will require each server to know to whom it is sending messages so that if it does not get a response, it can act accordingly. Similarly, if the down server rejoins from its last state rather than from an initial startup sequence, the rest of the distributed system must be aware and help reintegrating it.

## 5 Final Project: Awesome!

What you implement in the final project is up to you. At this point, the distributed database you've built is reasonably robust, but not extremely scalable. For example, since all the keys must be stored on every node, the number of keys in the system is limited by the memory of the least-powerful node. Similarly, performing an update requires finding a quorum of half the nodes, so growing the number of nodes by several orders of magnitude results in an unacceptable slow down. For the final project, you will propose a specific optimization to the system and implement it. Students generally work individually on this project, but for particularly ambitious proposals we allow pairs.