

# 6.005 Project 2: Collaborative Editor

## Design Documentation

Yonglin Wu, Yunzhi Gao, Shu Zheng

Last Revised: 12/04/2012

### Contents

I. General Design

II. Design Choices

## I. General Design

### 1. Overview

A collaborative editor allows multiple users to work on a single document simultaneously, across a network.

### 2. Graphical Summary of Design

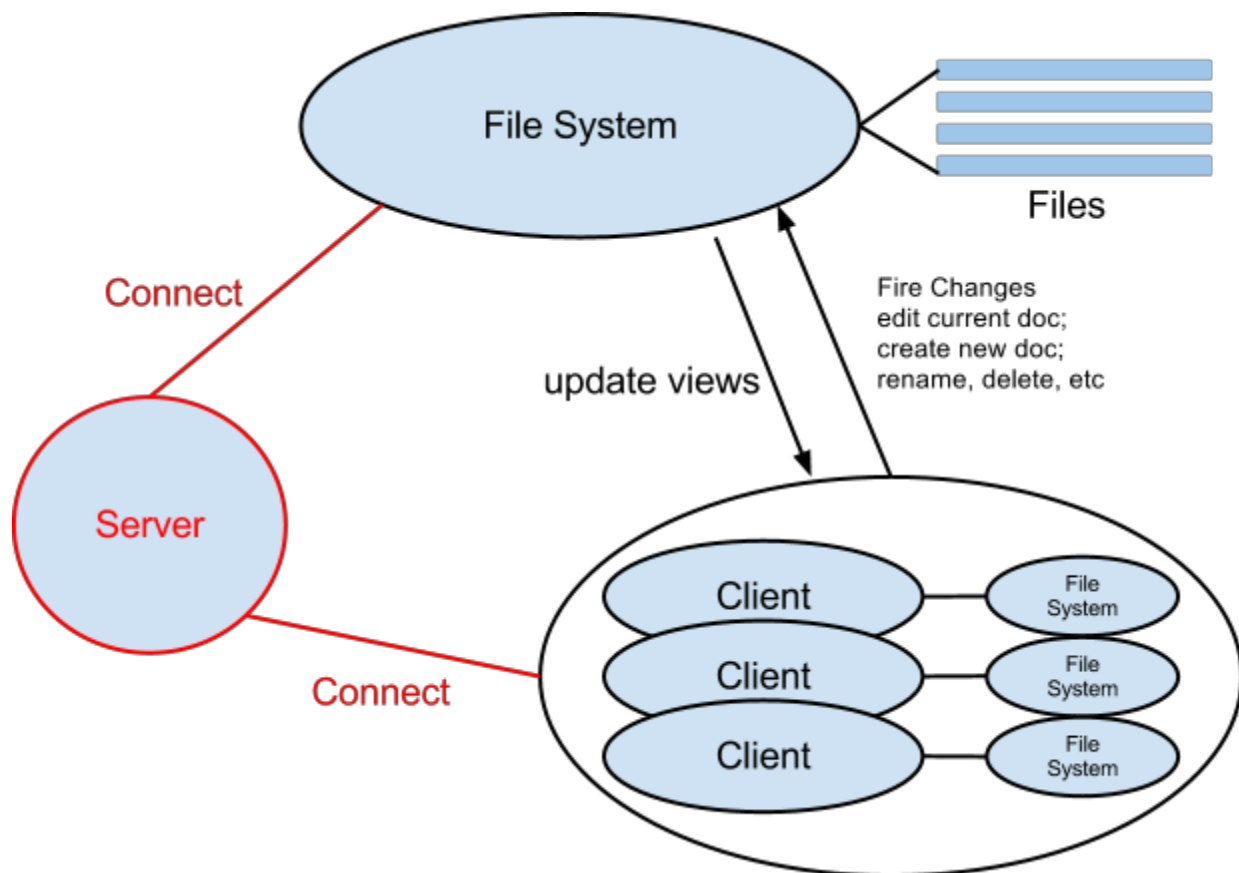


Figure 1 A

\* To support multiple document, we design a File System to store all the documents created so far.

*A File System is bound to each client and to the server. When a client makes an edit, create a new file, or insert/delete file, the local GUI will send a message (tentatively, via Java Event Package) to the File System in the server and update File System accordingly. The server will then send the same event package back to all the connected clients, and update the state of their respective local File Systems. Finally, the change of local File System will be reflected on each of the local GUIs.*

*In this way, each client/editor will be able to see the instantaneous change made on the working document (as well as the File System) whenever any of the co-editors has modified a single document or the multi-document set, hence achieving the collaborative functionality of the program.*

\* A new GUI View is created for each new editor/client connected;

### **3. Datatype**

The primary data that the users will access is the text in the textfield of the editor. We represent the text as an object of Java's default AbstractDocument. A document listener can be added to AbstractDocument to listen for edits on the document.

### **4. Multithreading & Concurrency**

AbstractDocument supports multiple reader and one writer. When multiple users try to write on the document at the same time, relevant listener on the view will send an update message to the File System. The state of the File System (e.g., individual file name and content, ordering of files, insertion and deletion of files) will be instantly updated according to the message received from the client side. We shall use synchronized methods to preserve the atomicity of File System states and prevent interleaving.

## **II. Design Choices**

*\* what set of editing actions are provided?*

For an individual document: change caret position, cut, copy, paste, select text, select all;

For multiple documents: insert/delete/rename files, save file to local disk.

*\* how documents are named and accessed by users?*

When the document is first created, it will have a default name (e.g., "Default Document"). Any user can change that name in GUI at any time. Same document name is shared among users.

*\* where documents are stored (e.g. at a central server or on clients)?*

Documents are stored in the File System of the central Server.

*\* what guarantees are made about the effects of concurrent edits?*

When a user edits the document, the attached document listener will fire the change to the File System on the server. The changes will be ordered (e.g., via a queue) in the server, which then handles the changes one by one and return the latest state of the File System to each client. In this way, valid concurrent multiple edits would be guaranteed.