# A poster for a robust key-value distributed system

**Haocheng Zhang**
**Department of Computer Science, College of Engineering, University of Illinois at Urbana-Champaign**

## Introduction

When we try to do some assignments, we also consider to make the backups to avoid the data loss. And the concept is also used in the Computer Science: a company will usually to have a distributed system, which have several machines all over the world. And when any machine update its data, other machine will also update; when one of the machines loss the data, it can retrieve the data from other system. So my project for this course is to build a simple key-value system to simulate the update on different machines.

## Goal

There are two big parts of this project:

1. I will build a simple system that supports communication among different servers/clients. They can send messages among each other and make response, just like the social software like Facebook or Imessager.

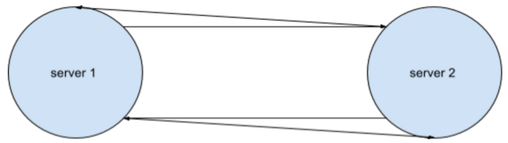
2. Based on the previous system, the different servers can also send some commands among each other. And the commands are mainly from key-value system, such as add, delete and update. In addition, when one of the servers change its local key-value system, other servers will change too.

## Method

1. For the communication part:

First I have 4 servers that can communicate between each other. And I will read the configurations of the servers from a configuration file, which includes the server number, server ip address port number and etc.

The algorithm I have used is for every server, it will have an input stream and an output stream. The input stream will read the data from the command line and other servers, while the output stream will send the data to other servers. The two stream will also both have a FIFO queue to have the data packet come on the time sequence. Here is a sample graph between two servers:




**2**. For the key-value system part:

First every server will have its own local key-value system, the system supports the basic command such as add, delete and add. And it will change the local system based on the command receive from other servers.

I will have a central server node which will broadcast all the requests from any server to all of the servers, including itself. The algorithm is: when any server receives the command, it will first send the cammand to the central server. Then the central server will broadcast the command to all of the servers. When any server receives the command, it will first change its own key-value system. Then it will respond to the central server. When the central server receives response from all of the servers, it will respond to the original server that the command has been implemented successfully.

## Results

Now the servers can communicate among each other and send the response back and forwards. In addition, it now support several key-value commands, here are they:

insert: insert a new key-value pair

delete: delete the key-value pair

update: update the key's value

find: find whether key is on the system

For more details, you can run my code and have a try to see what will happen

## What I have learnt

First, I have solidified my knowledge from CS241, including the networking, multiple threads, Linux commands and etc. In addition, I have self-learnt some interest algorithms and concepts from distributed system, such as central coordinates and linearizability. And the project seems to be useful to simulate in the real life environment.

ILLINOIS