# **CS425 – MP3 Report**

## **Group 31** **members**:

1. Xiaoming Chen (**xchen135**) 2. Piyush Shrivastava (**pshrvst2**)

## **Design**

This MP extends from MP2. At the end of MP2, we have a cluster implemented with Gossip Protocol for failure detection. In this MP we have designed a simple distributed file systems which uses Gossip protocol for failure detection and the Bully algorithm for leader election.

We have implemented the leader election protocol – Bully algorithm – because in our design, only the leader has the privilege of making updates on the file list. Below is the description of the algorithm running at each node -

### Algorithm used to join group:

1. Each machine has a unique id (ip: timestamp) and initially asks the user whether he/she wishes to join the membership. The user then selects the option to join. As soon as the node joins, it sends a UDP packet to the introducer.
2. The introducer listens to the incoming messages and as soon as it receives a packet from a new member, it adds it onto its membership list. It now gossips the membership list. All nodes gossip randomly to two nodes periodically.
3. Updates to the membership list is based on heartbeat and not on local clock. Please refer our MP2 report for further details on membership list update. In addition to MP2, in this MP the leader assigns a new process id to each joining node. In this way the oldest member in the network has the lowest id and gets the preference for being the leader in case the existing one dies.

### Algorithm for Introducer Rejoin

We have a scheduler which spawns a thread every 5 seconds to check whether it’s has the introducer in its list or nor. In case it doesn’t find the introducer in the list, it sends him heartbeat hoping that it will be alive at some point in the future. This logic allows us to have one group all the time. If we don’t have this feature, there will be a formation of two groups: one having older members and the other having the introducer and the nodes joining later.

### Algorithm for Leader Election

As soon as we have two nodes in the system, the first leader election takes place. Every node has an assigned id. We implemented Bully algorithm for election. The only difference from the original algorithm is that in our case, the lowest id gets the priority and ultimately sends the coordination message.

### Integration with MP1

Like MP1 and MP2, we have used log4j in our MP3 solution to log the activities occurring at the respective machines. With MP1 we were easily able to retrieve entries from logs like deleting a member from the member list, the transfer of files using get and put, leader election, etc. MP1 was extremely helpful in debugging the issues which we had during the development phase.

## **Results and graphs**

Time between master failure and election results: Avg. time = 3589 ms | Std. Dev = 129 ms

Avg. bandwidth = 3.65 KB/ sec

Transferring the entire Wikipedia corpus was done in parts of 5 sharded files. Average transfer time was 1002 seconds.

Graph analysis: We notice that the range of time for transferring small files is somewhat big when compared to transfer of large files. One reason is the socket connection time which for the small files transfer time cannot be ignored.

**Summary**: In this MP we implemented a Simple Distributed File System. The highlights for our SDFS was the gossip based failure detection, gossip based file list transfer and the implementation of Bully algorithm for leader election. In this SDFS, for a put command, the leader randomly allows the put on any three active nodes. In future, we plan to have a load balancing mechanism so that every node have somewhat equal load.