Eldin Turulja & Edin Hanic

White Paper

Our current project is implemented as a one-shot client to a single server, so we will be extending to a one-shot with multiple servers. Firstly, because this seems like the simplest to implement as it doesn't require many changes on our client side, and less changes will lead to more time developing the new features on the server. Secondly, this allows us to use a static list of servers that the clients can connect to, so when one server goes down, the others can hold their election and then the clients can be serviced by the new coordinator.

We will use the Bully Algorithm to elect which server is going to act as the coordinator. We will detect if a server is down by the client first attempting to connect to the downed server. The client will not be able to connect, so it will try the next server in the list. When the server that is not a coordinator receives the clients connection, it will ping the coordinator to check if it is truly down. If the ping comes back negative, then we know we must hold an election. Also, when a server comes back to life, it will also hold another election. We chose this route of implementation because it provides a simple, clear, and consistent methodology for making sure the client is always serviced, if servers are available.

We will use Client-Centric Consistency. We chose this model since we want to use Vector clocks to synchronize. To implement we will base our algorithm off of the slides 58-60 in the replication-and-consistency-handout.pdf. Our inconsistency window will be however long it takes for a server to fetch the writes from another server that had been servicing the current client.

We will be using Vector clocks to track changes in our servers as that is what was recommended in class. When we synchronize we will be copying the entire database. Though this may not be the most resource efficient way, we believe it solves the problem that was

proposed by this assignment. We do not expect to have to service an enormous amount of clients at once, so this solution should be plenty robust to allow our servers to function properly under the low load we are expecting.