- Controller stores:

  - List of active storage nodes (could be a map)
    nodeID → node{}

  - Mapping for file names & chunks
    → Storage nodes.

- For client PUT request:

  - Reply with a list of dest. storage nodes.
  - Replica locations
  - Controller sees no files (only metadata).

FIMP

- Replication

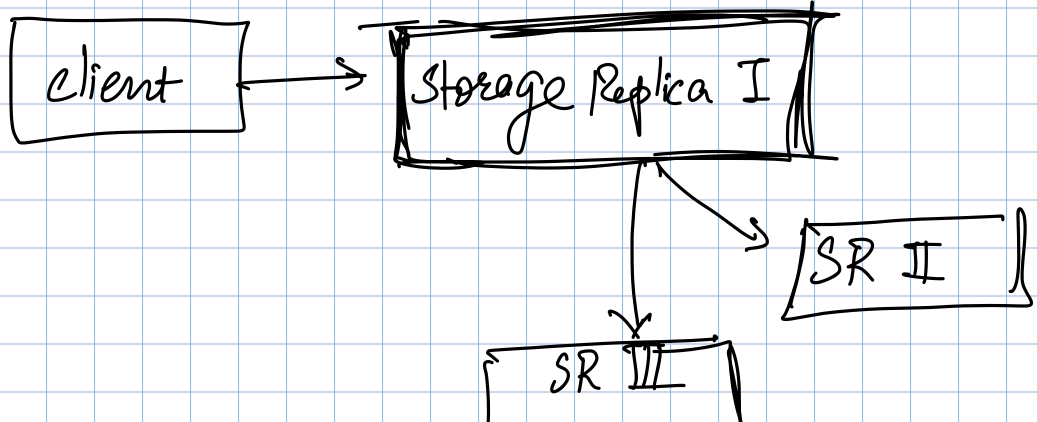
  - Responsible for detecting storage node failures
  - Ensuring replication is maintained.
  - System stores 3 copies of a chunk.
  - If a storage node goes down, the back up copy should have the retrival requests sent to.
  - Self heal by creating more copies on failure.
    (need to design an algo to determine placements)

Class disscussion for Controller:

- Controller doesn't see any file data to be stored.
- Controller will lay out a plan for the client wherein it describes:
    - Storage layout (includes replication plan)
    - Retrival information that would include replicati -on info.
    - Assumes client know nothing

~ List of chunks the client is looking for
~ Replicas (Helps client navigate)
~ locations

- Clients receive the storage layout generate by controller and breaks the file accordingly and sends it to one of the suggested replica hosts. The replica storage hosts are supposed to pipe the replicas to other responsible hosts.



- Make sure the load is spread out across the cluster
- Small files shouldn't be split.
    - Default could be 128MBs.

- Controller can be shut down, and list of files & list of storage nodes need to be stored on the disk.
    - The data doesn't necessarily have to be stored on disk for the controller, insted we can have the storage node send their index to the controller.

- DONT HAVE REPLICAS ON THE SAME MACHINE.

- Chunks will have corruptions during demo, so make sure there's a checksum check to check for corruption

- <u>Storage node:</u>
    - Responsibile for <u>storing & retriving</u> file chunks.
    - Stored files will be checksummed to prevent on disk corruption.
    - <u>On Corruption detection:</u>
        - Request a replica before fullfilling client req.
        - Store metadata like checksum on disk.

    - Send periodic heartbeat to the controller.
    - <u>Heartbeat contents</u>: (send every 5 seconds)
        - Free space available @ node.
        - Total num of requests processed (storage, retrievals, etc.)
        - New files stored @ the node

- Client:

  - Breaks the files into chunks
  - Asks the controller where to store them
  - Send file chunks to appropriate nodes.
    - Storage node passes the replicas not the client.

    - No duplicate files allowed. Client can remove the files.

    - There is a default chunk size, but the user can overwrite it.

  - Files should be retrived in parallel.
    - Each chunks gets requested & transferred in parallel.

  - Client can delete existing files.
  - Can 'ls' on the system.
  - Prints out: (Get's from the controller)
    - list of active nodes
    - Disk space in the cluster

- Num of requests handled by each node