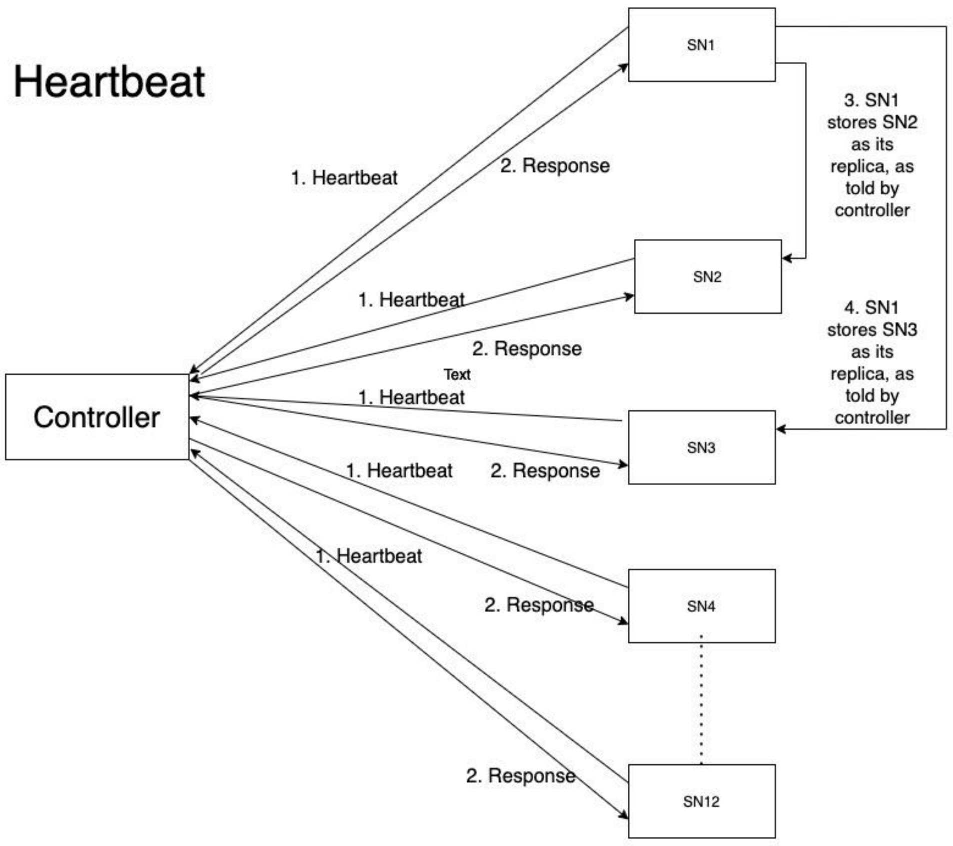
Design Document : Project 1 : CS677

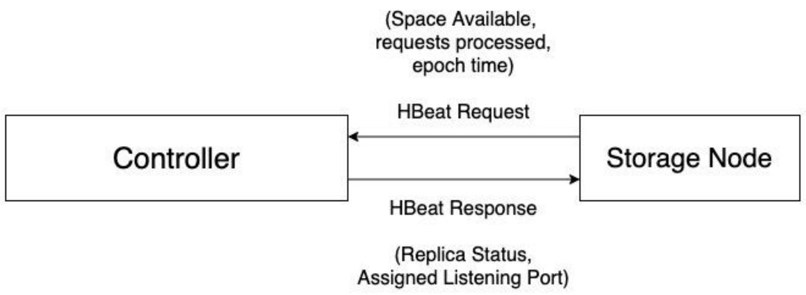
Team Members : Adarsh, Ayush

- **Components of your DFS (this includes the components outlined above but might include other items that you think you'll need)**
- **Messages the components will use to communicate**

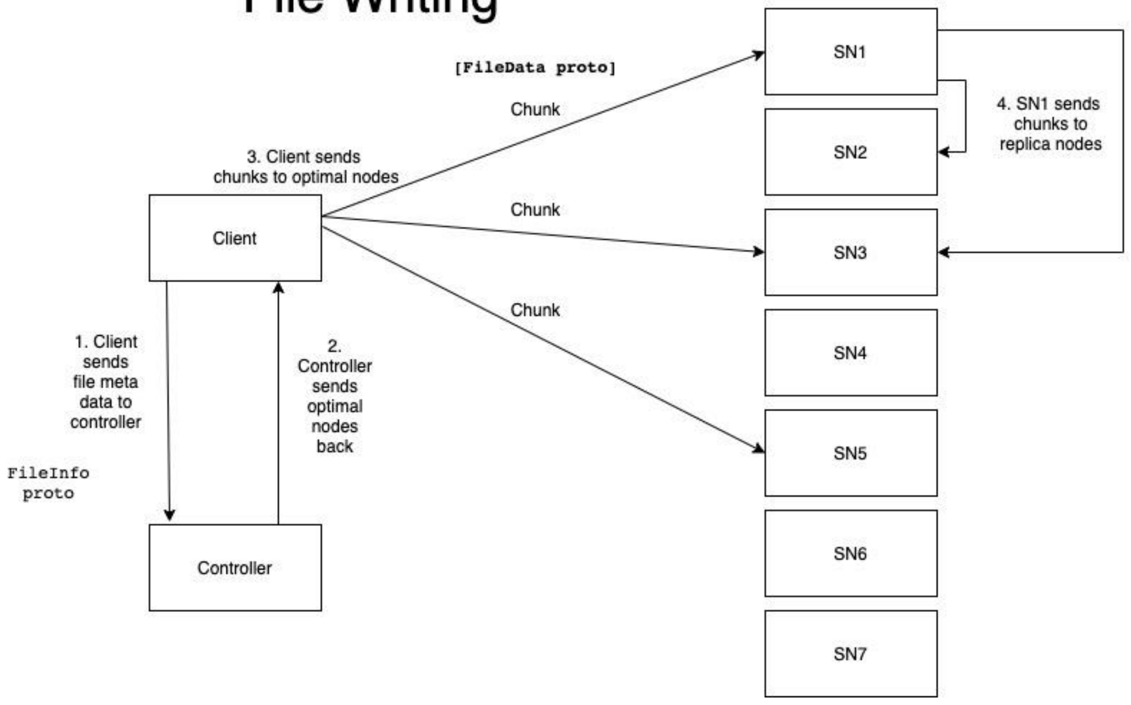
The names of protocol buffer messages are provided in the following diagrams. We have also explained the design and flow of our project implementation in the diagrams below.



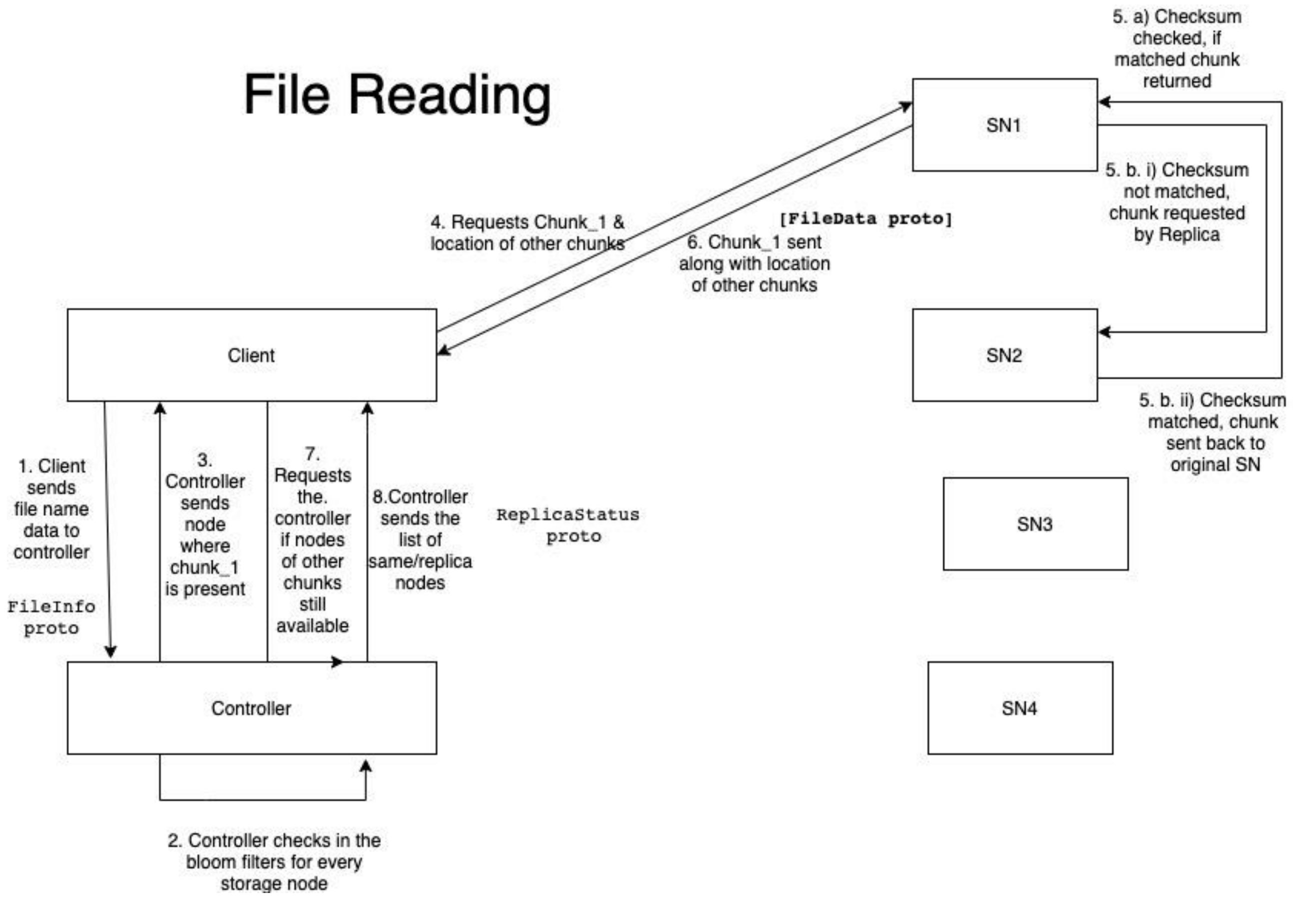
Heartbeat proto


(Space Available, requests processed, epoch time)
HBeat Request
Controller
Storage Node
HBeat Response
(Replica Status, Assigned Listening Port)

File Writing

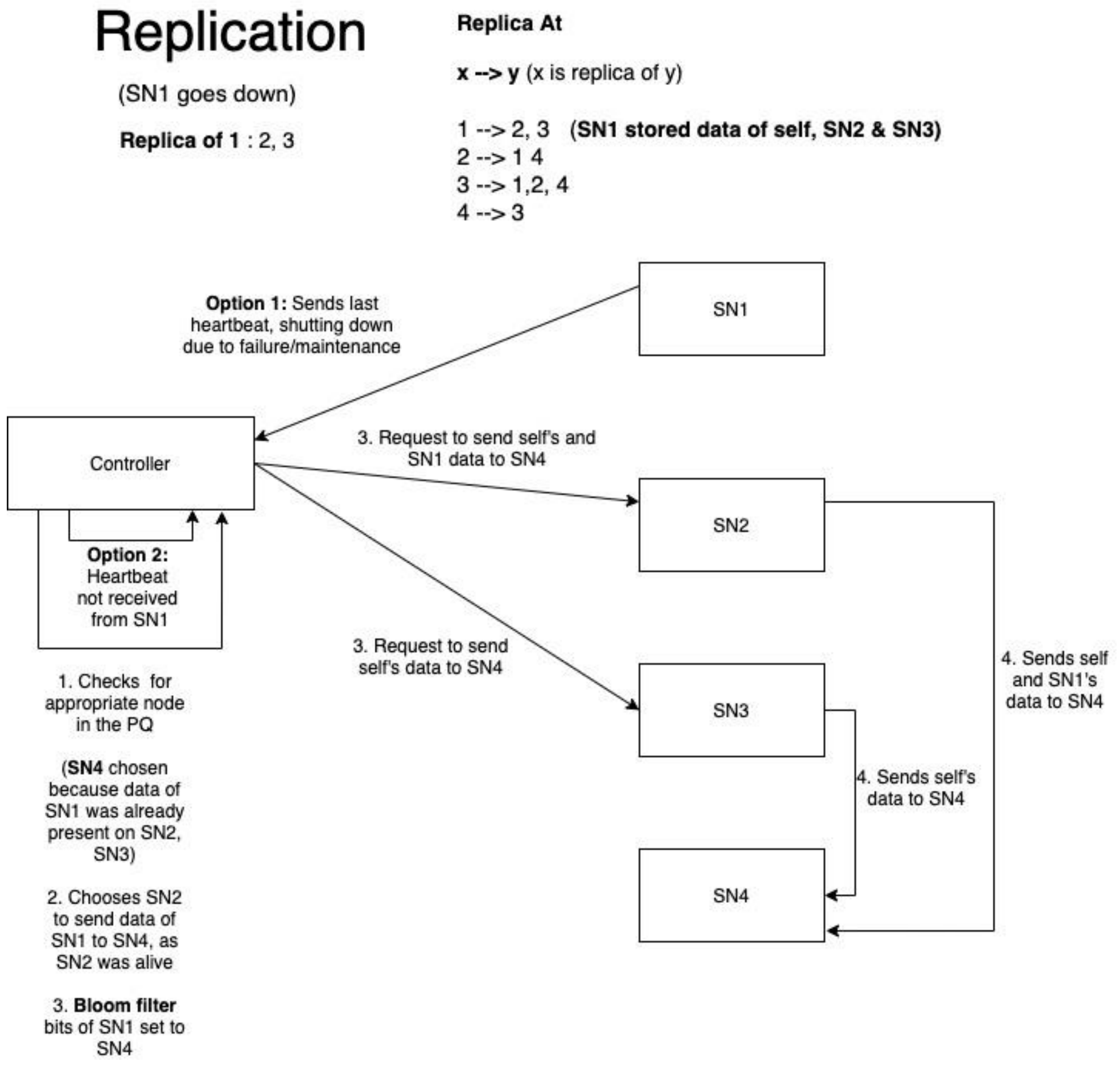

[FileData proto]
Chunk
3. Client sends chunks to optimal nodes
Chunk
Chunk
Client
1. Client sends file meta data to controller
FileInfo proto
2. Controller sends optimal nodes back
Controller
SN1
SN2
SN3
SN4
SN5
SN6
SN7
4. SN1 sends chunks to replica nodes

File Reading


5. a) Checksum checked, if matched chunk returned
SN1
5. b. i) Checksum not matched, chunk requested by Replica
4. Requests Chunk_1 & location of other chunks
[FileData proto]
6. Chunk_1 sent along with location of other chunks
Client
SN2
5. b. ii) Checksum matched, chunk sent back to original SN
1. Client sends file name data to controller
FileInfo proto
3. Controller sends node where chunk_1 is present
7. Requests the. controller if nodes of other chunks still available
8.Controller sends the list of same/replica nodes
ReplicaStatus proto
SN3
Controller
SN4
2. Controller checks in the bloom filters for every storage node

- **Design decisions (how big the chunks should be, how you will place replicas, etc…)**
  The considered chunk size is 100MB, because for files larger than 1GB, smaller chunk size will create loads of chunks, leading to high latency due to excessive communication between storage nodes and client.

  We will consider the next two consecutive, nodes as the replica using round robin. Under replication is handled by Priority Queue, used as a max heap; node with largest space available is chosen first.


Replication
(SN1 goes down)
Replica of 1 : 2, 3
Replica At
x --> y (x is replica of y)
1 --> 2, 3 (SN1 stored data of self, SN2 & SN3)
2 --> 1 4
3 --> 1,2, 4
4 --> 3
SN1
Option 1: Sends last heartbeat, shutting down due to failure/maintenance
Controller
3. Request to send self's and SN1 data to SN4
SN2
Option 2: Heartbeat not received from SN1
3. Request to send self's data to SN4
SN3
4. Sends self and SN1's data to SN4
4. Sends self's data to SN4
SN4
1. Checks for appropriate node in the PQ
(SN4 chosen because data of SN1 was already present on SN2, SN3)
2. Chooses SN2 to send data of SN1 to SN4, as SN2 was alive
3. Bloom filter bits of SN1 set to SN4

Retrospective Questions

Q - How many extra days did you use for the project?

A – We didn't use any extra days for the project. We submitted the project before 11:59p on 10/11/2020.

Q - Given the same goals, how would you complete the project differently if you didn't have any restrictions imposed by the instructor? This could involve using a particular library, programming language, etc. Be sure to provide sufficient detail to justify your response.

A – We didn't feel any restrictions with respect to programming language or libraries, as the allowed libraries and JAVA were sufficient to complete the project effectively.
We thought of implementing RAFT early in the project and have even written code about it which is commented in our repo currently. We decided to discontinue with it for now but have plans to implement it sometime later in the future.

Q - Let's imagine that your next project was to improve and extend P1. What are the features/functionality you would add, use cases you would support, etc? Are there any weaknesses in your current implementation that you would like to improve upon? This should include at least three areas you would improve/extend.

A - We thought of implementing multiple controllers which could have been coordinated by something called the Mega-Controller. Replication Management would then have to be handled among the controllers as well. Also, we prefer to add proper routing tables for each node, with Chord implemented to route through the tables.
We also thought about creating an API, where a third-party user could utilize our DFS, along with the CLI. Also, this could us to use our DFS for some other project.

Q - Give a rough estimate of how long you spent completing this assignment. Additionally, what part of the assignment took the most time?

A – We spent around 16 days of time on the project, with each of us spending around 5 hrs a day on average. The part which took the most time was replication management.

Q - What did you learn from completing this project? Is there anything you would change about the project?

A – In addition to better learning previously learned concepts like multithreading, file handling, use of data structures like max heap, we learned a lot about the Distributed file system. It was a hugely different experience applying the concepts on the project than reading the HDFS paper, which nevertheless had set the ground foundation for us. Apart from introducing file locking, currently, we don't feel that we would change anything about the project. Maybe later as we learn new concepts, we could think of something.

Q - If you worked as a group, how did you divide the workload? What went well with your team, and what did not go well?

A: The Design document was handled by Ayush. This retrospective Document was handled by Adarsh. Other than that, our group didn't really divide things. We were always on a Zoom call, and discussed the logic and wrote code with alternate pushing on GitHub. We did the controller, the client and the storage node together, but tested them simultaneously on each of our computers. The reason we did in this way as the flow of the project is very much interconnected. We felt the if we give for example Client to one person and Storage Node to the other person, it would be very difficult to co-ordinate the logic and design of the project.
What didn't went well, was that we were always at conflict with some of the design decisions. Ayush wanted things to be done in a perfect and efficient way and Adarsh wanted things to be done first and sooner and concentrate on the efficient part later. In our group, we both wanted to do things and learn so it created some conflicts. We have decided to work alone on the rest of the projects.