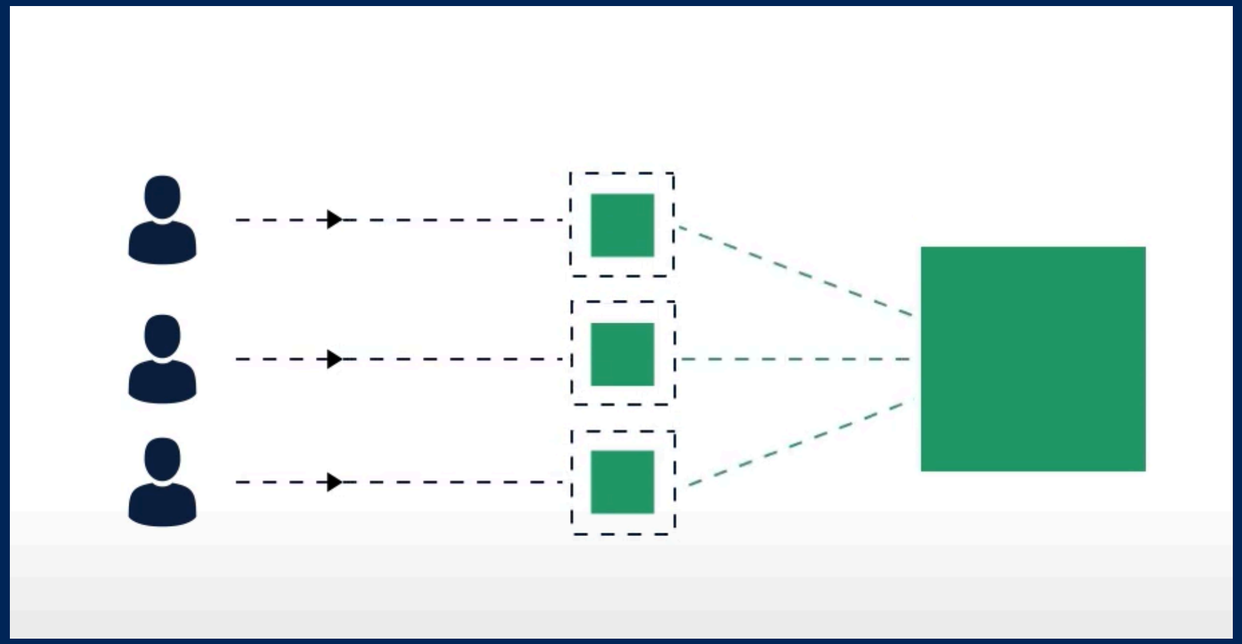
SHFS: A modern gRPC Distributed File System

Siddharth Narsipur
Henry Liu

University of Rochester

Background

- Modern computing increasingly relies on scalable, fault-tolerant systems to manage and transfer large volumes of data efficiently.

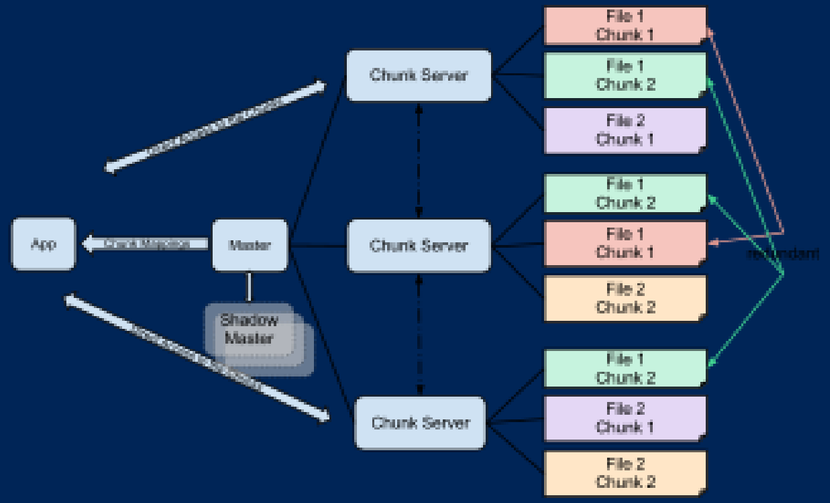


Background

- Distributed file systems play a critical role in environments where perfomance, reliability, and scalability are necessary.

Related Work & Inspiration

- Google File System (2003)



- CMU Course Assignment (2020)

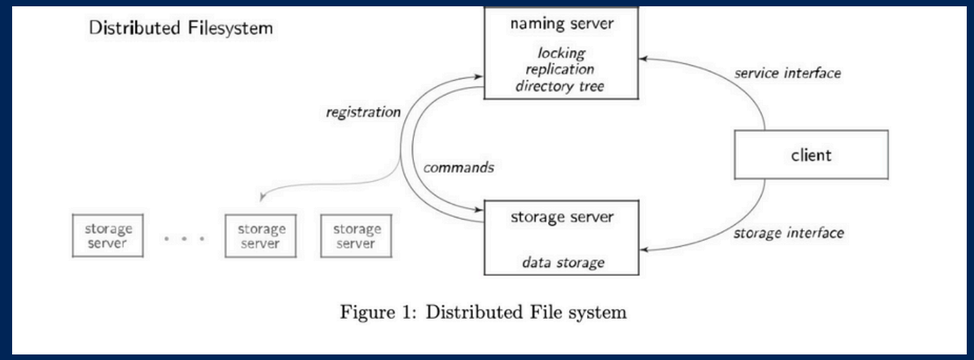
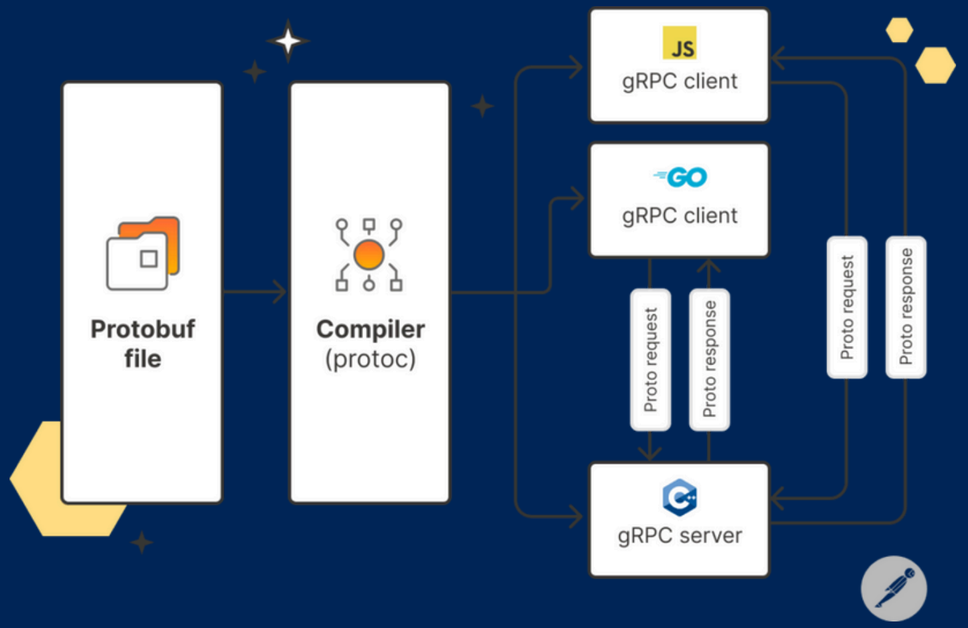


Figure 1: Distributed File system

Our Contribution

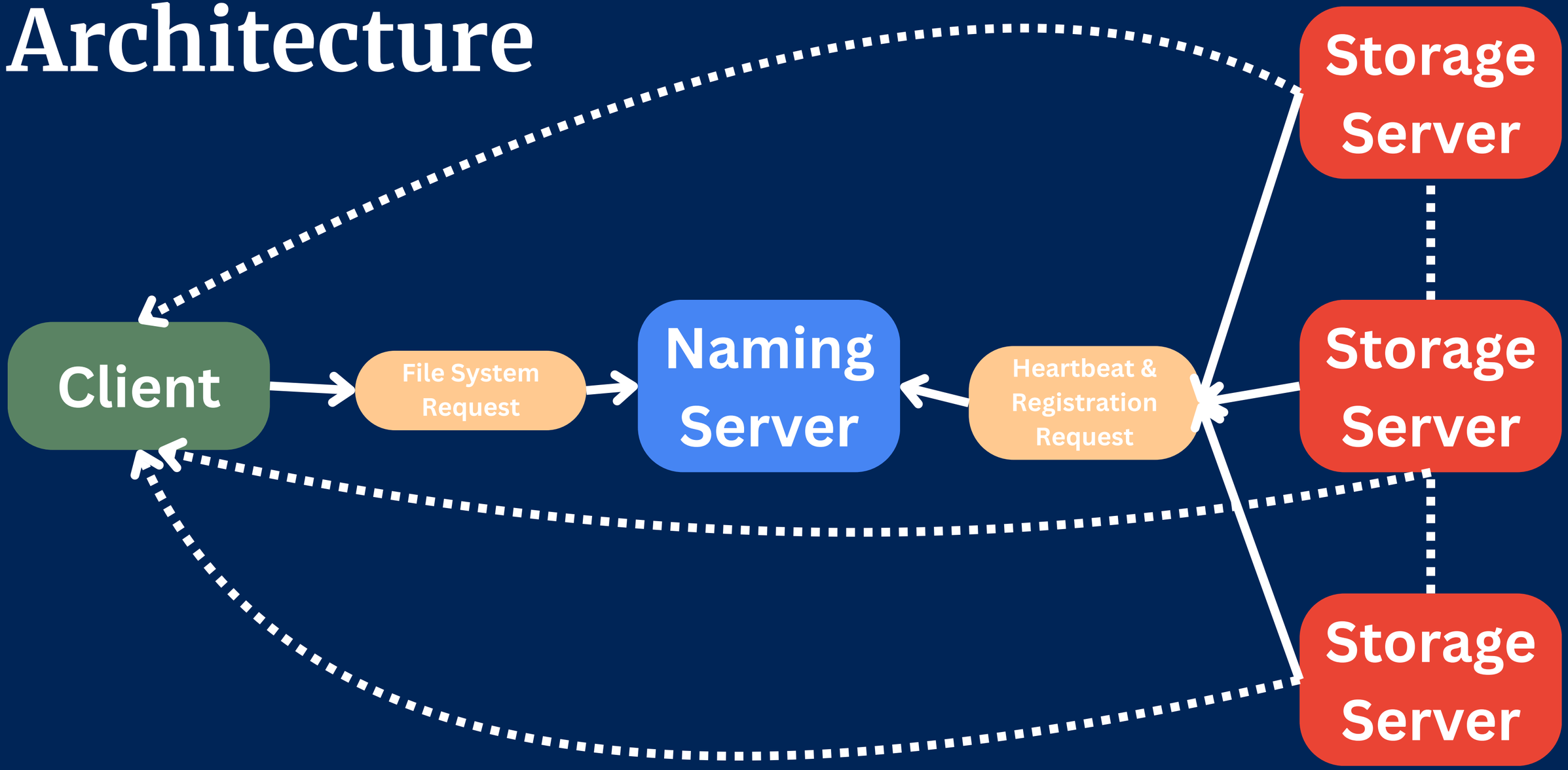
- Use gRPC, a high-perfomance, open-source universal RPC framework as our communications framework.



SHFS Features

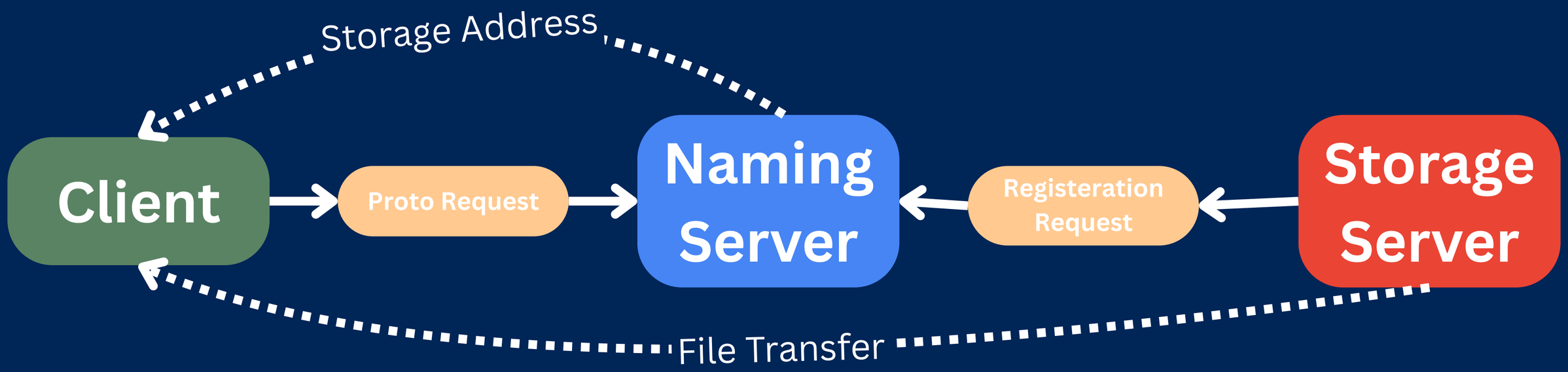
- Dynamic rebalancing for resource efficiency
- Streaming and parallelism for fast data transfer
- Fault tolerance through replication
- Scalability through concurrency and decentralization

Architecture



Architecture

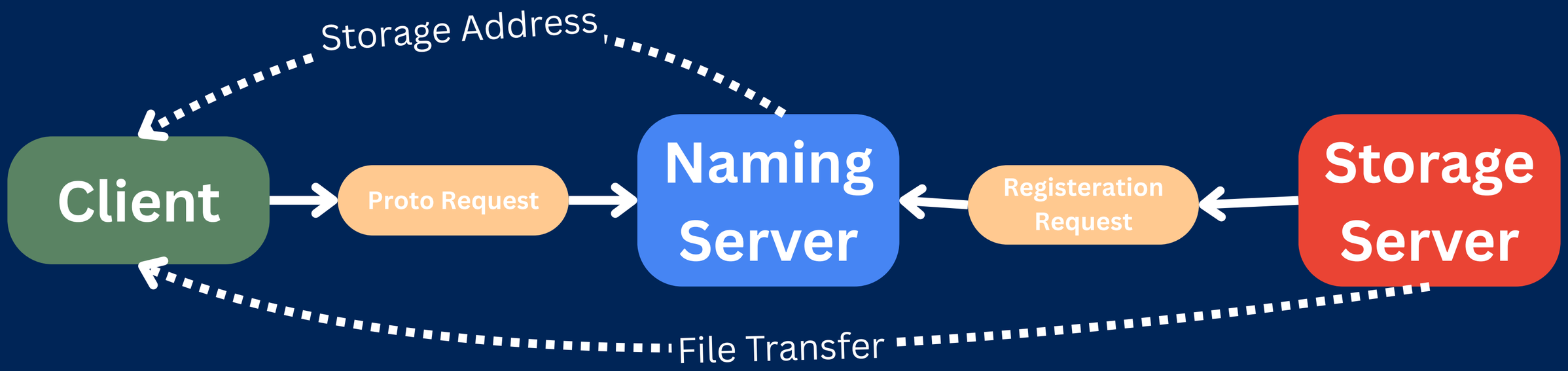
- The Naming Server is the main point of contact between clients and the SHFS system.
- Maintains medata of the file systems and coordinates interactions with the storage servers.



File Operations

- When a client requests a file operation, the naming server returns a list of storage servers that maintains consistent load and the required replication factor.

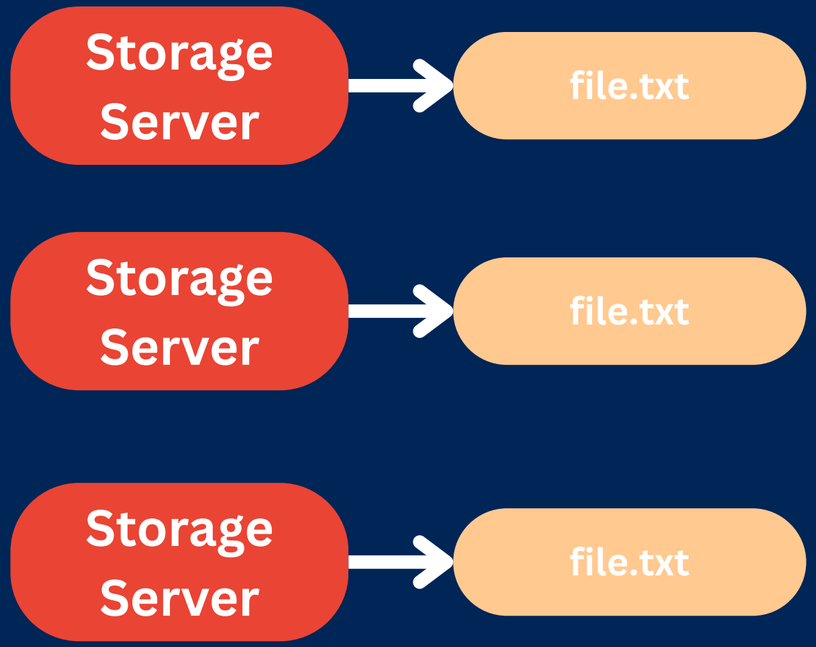
- The client then contacts the storage server directly to complete the operation; preventing the naming server from becoming a bottleneck.



Replication

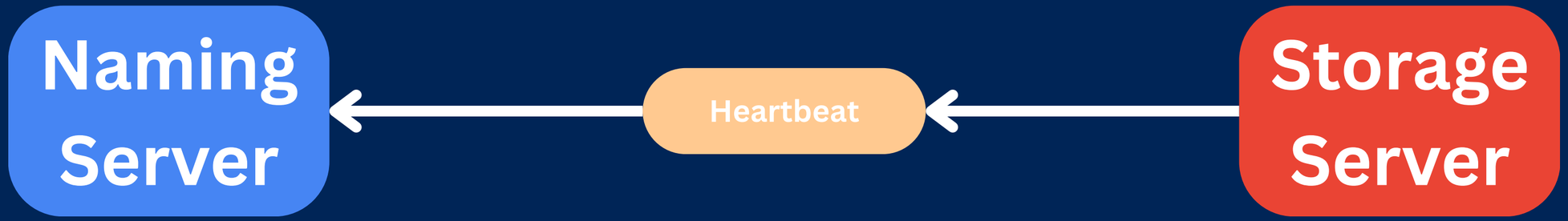
- Supports configurable file replication to ensure high availability and durability.

- If a storage server goes down, replication is automatically initiated by the naming server



Heartbeat Detection

- Storage server independently and periodically send heartbeat messages to the naming server.
- Servers that are silent are marked inactive and replication in initiated.
- Active servers will receive a list of replication tasks and initiate replication automatically



Dynamic Rebalancing

- When a new storage node joins the systems, the naming server initiates a rebalancing process.

- New file uploads are spread across servers systematically to spread file load.

Client Interface

```
~/Projects/DFS/cmake-build-default/src/client git:(main) $ client upload henry.txt
Successfully uploaded to: localhost:8000
Successfully uploaded to: localhost:9000
Successfully uploaded to: localhost:7000
~/Projects/DFS/cmake-build-default/src/client git:(main) $
```

```
~/Projects/DFS/cmake-build-default/src/client git:(main) $ client info
file | servers
----------------
henry.txt | localhost:7000, localhost:8000, localhost:9000
```

```
~/Projects/DFS/cmake-build-default/src/client git:(main) $ client remove henry.txt
Succesfully Removed File
~/Projects/DFS/cmake-build-default/src/client git:(main) $ client info
file | servers
---------------
```

Demo