**Design Document**

Team 6

Ana Taylor | Puma Damdinsuren | Mitchel Downey | Virmel Gacad

**Architecture of SUFS**

SUFS system will consist of three individual subsystems, communicating via HTTP calls with Flask servers. The client will be how the user interacts with the system. The NameNode will maintain a list of the contents of each DataNode and route read requests appropriately. The DataNodes will store blocks of files and periodically communicate to the NameNode their stored contents.

When writing a file to the system, the client will contact the NameNode to request the space for the file. The NameNode will reply with a list of DataNodes for the client to write to. The client will then break up the file according to the block size and send each block to each DataNode. The DataNodes will then store those blocks.

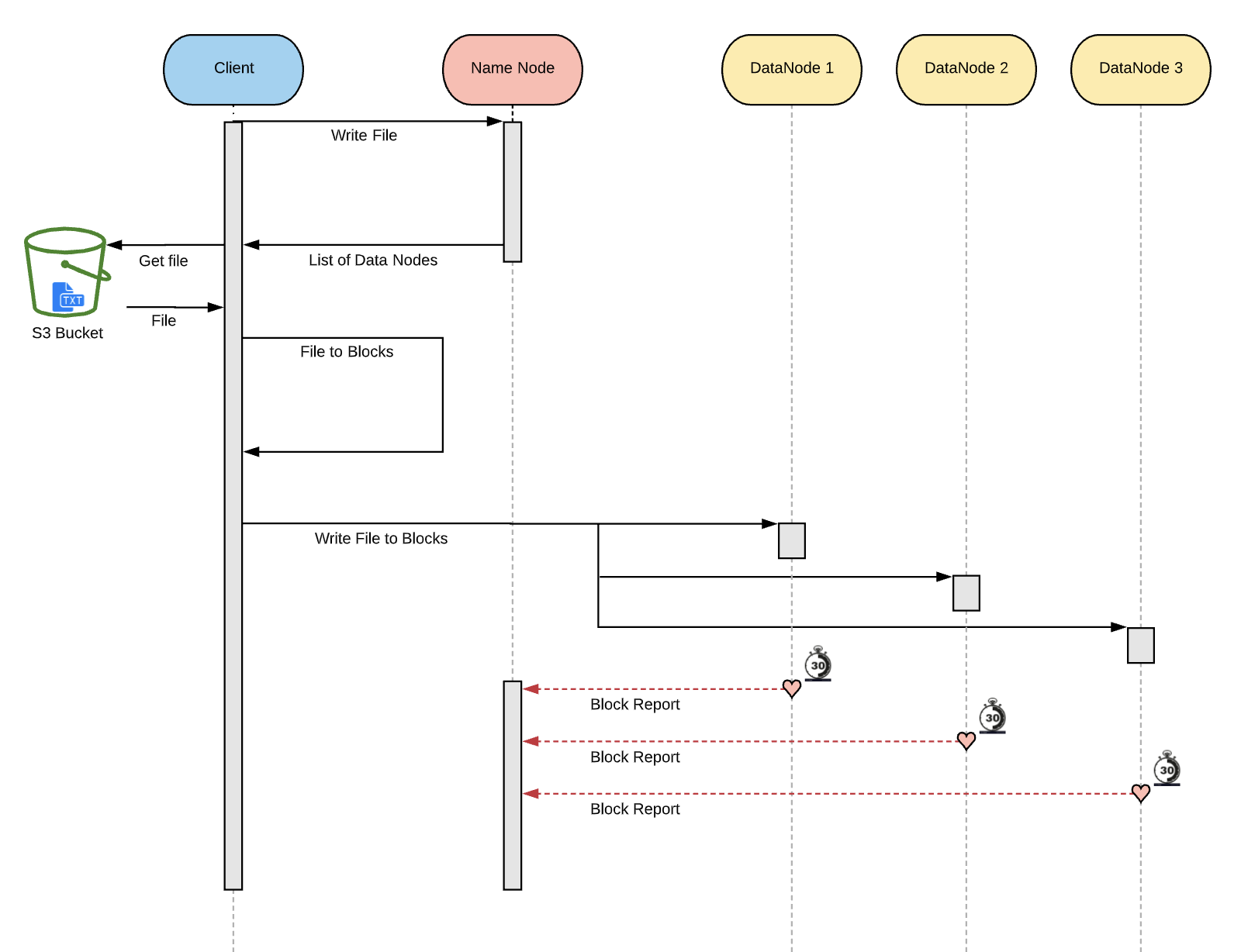
Similarly, when reading a file, the client will contact the NameNode to request a list of DataNodes that hold the blocks for the requested file. The NameNode will reply with a list of the block IDs that the file was broken into and the IP addresses of those DataNodes that store each block. The client will then request each block from one DataNode and combine those blocks back into the complete file.

A screenshot of a cell phone

Description automatically generated

**Sequence diagrams**

*Write File sequence with BlockReport/Heartbeat*

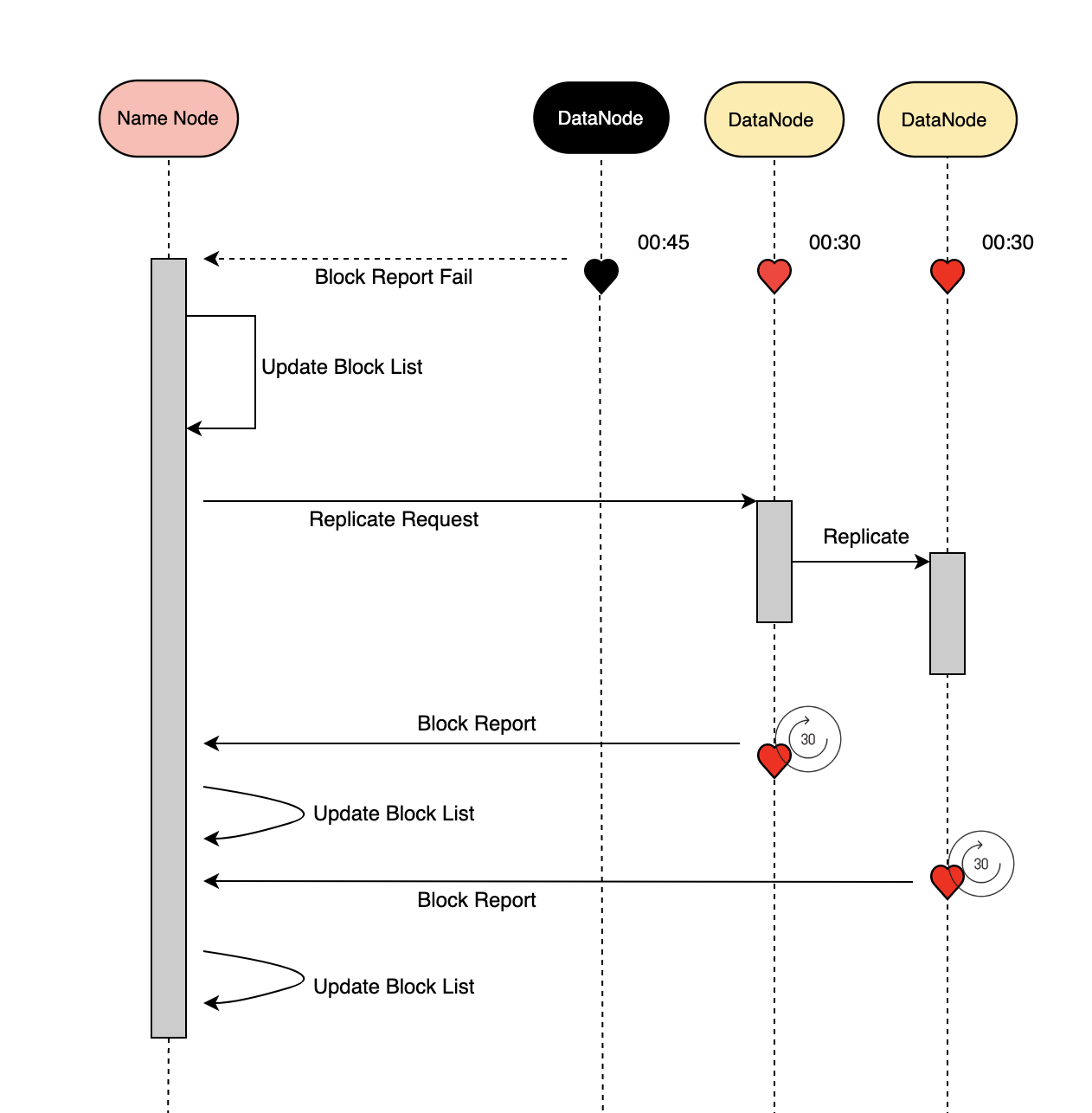


*Read File sequence with BlockReport/Heartbeat*

A screenshot of a cell phone

Description automatically generated

*Data Node failure sequence with BlockReport/Heartbeat*



**API documentation**

**NameNode.py:**

route:  **'/blocks', POST**

args: fileName - string of the name of the file to be written

fileSize - size of the file to be written, in bytes

{

“filename” : fileName,

“fileSize” : fileSize

}

returns: dictionary of BlockID’s and list of DataNodes the block should store this block,

where BlockID is represented as a number and DataNodeID is represented as IP.

{

0 : [DataNode0, DataNode1, DataNode2, ...],

1 : [DataNode0, DataNode1, …],

...

}

route:  **'/blockList/fileName', GET**

args: fileName - string of the name of the file to be read

returns: dictionary of BlockID’s and list of DataNodes that store this block,

where BlockID is represented as a number and DataNodeID is represented as IP.

{

0 : [DataNode0, DataNode1, DataNode2, ...],

1 : [DataNode0, DataNode1, …],

…

}

route:  **'/IP/', PUT**

args: Dictionary containing data node ID and its IP address

{

0 : [DataNode0 IP],

1 : [DataNode1 IP],

…

}

returns: None

route:  **'/blockReport', PUT**

args: Dictionary of data node's ID and an array of its held blocks

{

“ID”: DataNodeID,

“report” : [Block0, Block1,...]

}

returns: None

**DataNode.py:**

route: **‘/dataChunk/blockID’, PUT**

args: base64 encoded data chucks each of block size, evenly divided (32MB)

{

“encodedData” : base64EncodedData

}

returns: None

route: **'/data/blockID’, PUT**

args: blockID

returns: None

route: **'/storedData/blockID, GET**

args: None

returns: base64 encoded data chucks each of block size or smaller (128MB)

{

“encodedData” : base64EncodedData

}

**Name Node Data Structures**

**fileList = {}**

* + Dictionary of file names and their associated block IDs
  + Key: name of the file

Value: Block IDs that store chucks of this file, represented as a number (0, 1, 2…)

{

fileName1 : [Block0, Block1, Block2, ...],

fileName2 : [Block3, Block4, Block5...],

…

}

**DataNodeIP = {}**

* + Dictionary of DataNode IDs and their IP addresses
  + Key: DataNode ID represented as number (0, 1, 2, 3, …)
  + Value: DataNode IP represented as an IP address (1.2.3.3)

{1 : "34.219.170.244", 2 : "54.200.100.114", …}

**blockStatus = {}**

* + Dictionary of DataNode IDs and the blockIDs which are stored on it
  + Key: DataNodeIDs represented as number (0, 1, 2, 3, …)
  + Value: list of blockIDs stored in this data node represented as a number (0, 1, …)

{

DataNode1: [Block0, Block1, Block2,...] ,

DataNode2: [Block0, Block1, Block2,...] ,

…

}

**timeRecord = {}**

* + Dictionary of data node IDs and the last time they have sent blockReport
  + Key: DataNodeIDs represented as number (0, 1, 2, 3, …)
  + Value: time (when the block report is last sent) represented as seconds since the epoch

{

{0: 30, 1: 25, ... }

}

**Data Node Data Structures**

**heldBlocks= []**

* + List of blockIDs that contains the file (or part of file if it has been divided due to its’ size)
  + It represents which file blocks each DataNode contains
  + BlockIDs represented as numbers (0, 1, 2, 3, …)

**Technologies/Tools**

Programming language: Python

Networking Framework: Flask

AWS SDK: AWS Boto3

Servers: AWS EC2, 1 NameNode, >=N DataNodes, 1 Client

File storage: AWS S3

Version Control: AWS CodeCommit

**System parameters**

Block size = 128MB

Replication factor N = 3

Block Report/Heartbeat frequency = 30 secs