Component API Calls

We are using RESTful endpoints to communicate between the components. Only JSON is supported.

Name Node API

Create File Request

path: "/createFile"

```
{
    "FileName": string, // the filename SUFS will use
    "Size": string // this number of bytes in the file
}
```

Create File Response

```
{
    "BlockInfos": [ // in-order list of info for each block
        {
            "BlockId": string, // the internal id of the block
            "DataNodeList": [ // list of Data Nodes the block should be
stored on
                string // IP address and port of the Data Node (ex:
"10.0.0.1:8080")
            ]
        }
    ],
    "Error": string // description of the error, empty means no error
}
```

Get File Request

path: "/getFile"

```
{
    "FileName": string, // the filename in SUFS
}
```

Get File Response

```
{
    "BlockInfos": [ // in-order list of info for each block
```

```
        {
            "BlockId": string, // the internal ID of the block
            "DataNodeList": [ // list of Data Nodes the block should be stored on
                string // IP address and port of the Data Node (ex: "10.0.0.1:8080")
            ]
        }
    ],
    "Error": string // description of the error, empty means no error
}
```

Block Report Request

path: "/blockReport"

```
{
    "MyIp": string, // the public IP address of the sending Data Node
    "BlockIds": []string // the list of IDs of each block stored on the sending Data Node
}
```

Block Report Response

```
{
    "Error": string // description of the error, empty means no error
}
```

Heartbeat

FINISH ME

Data Node API

Store Block Request

path: "/storeBlock"

```
{
    "Block": string, // base64 encoded block data
    "DataNodeList": [ // list of Data Nodes the block should be stored on
                string // IP address and port of the Data Node (ex: "10.0.0.1:8080")
    ],
    "BlockId": string // the internal ID of the block
}
```

Store Block Response

```
{
    "Error": string // description of the error, empty means no error
}
```

Get Block Request

path: "/getBlock"

```
{
    "BlockId": string // the internal ID of the block
}
```

Get Block Response

```
{
    "Block": string, // base64 encoded block data
    "Error": string // description of the error, empty means no error
}
```

CLI

Each command can include the `-v` option. This turns verbose mode on. When verbose mode is on the CLI will output log statements as it performs the action.

Create File Command

```
/path/to/CLI create-file <name_node_address_and_port> <file_name> <s3_url>
```

- name_node_address_and_port
  - address is required
  - :port is optional
  - ex: "10.0.0.8", "10.0.0.8:8080"
- file_name
  - the name of the file in SUFS
- s3_url
  - the URL of the file to put into SUFS

Get File Command

```
/path/to/CLI get-file <name_node_address_and_port> <file_name>
<save_location>
```

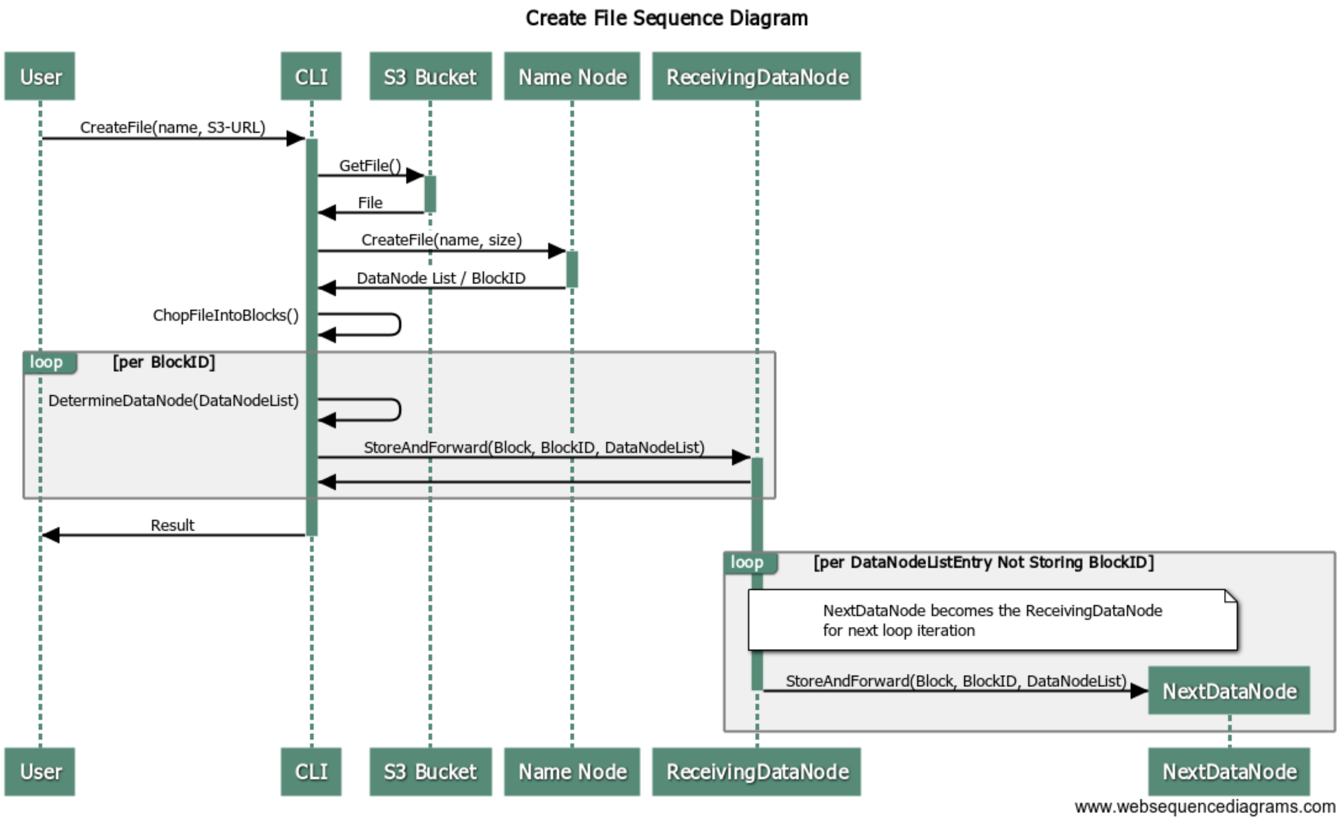
- name_node_address_and_port
    - address is required
    - :port is optional
    - ex: "10.0.0.8", "10.0.0.8:8080"
- file_name
    - the name of the file in SUFS
- save_location
    - the location on the local host to save the file

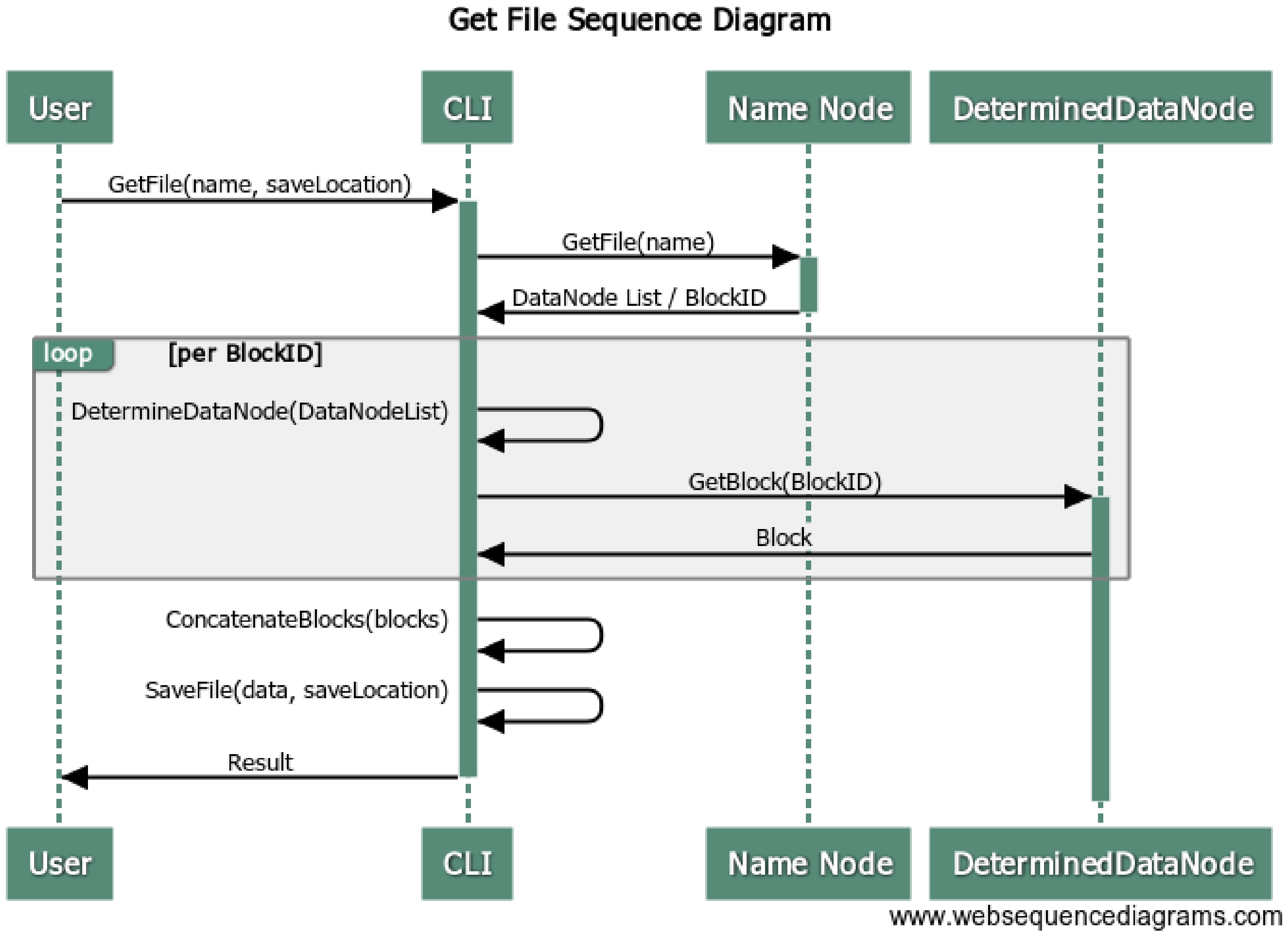
List Data Nodes File Command

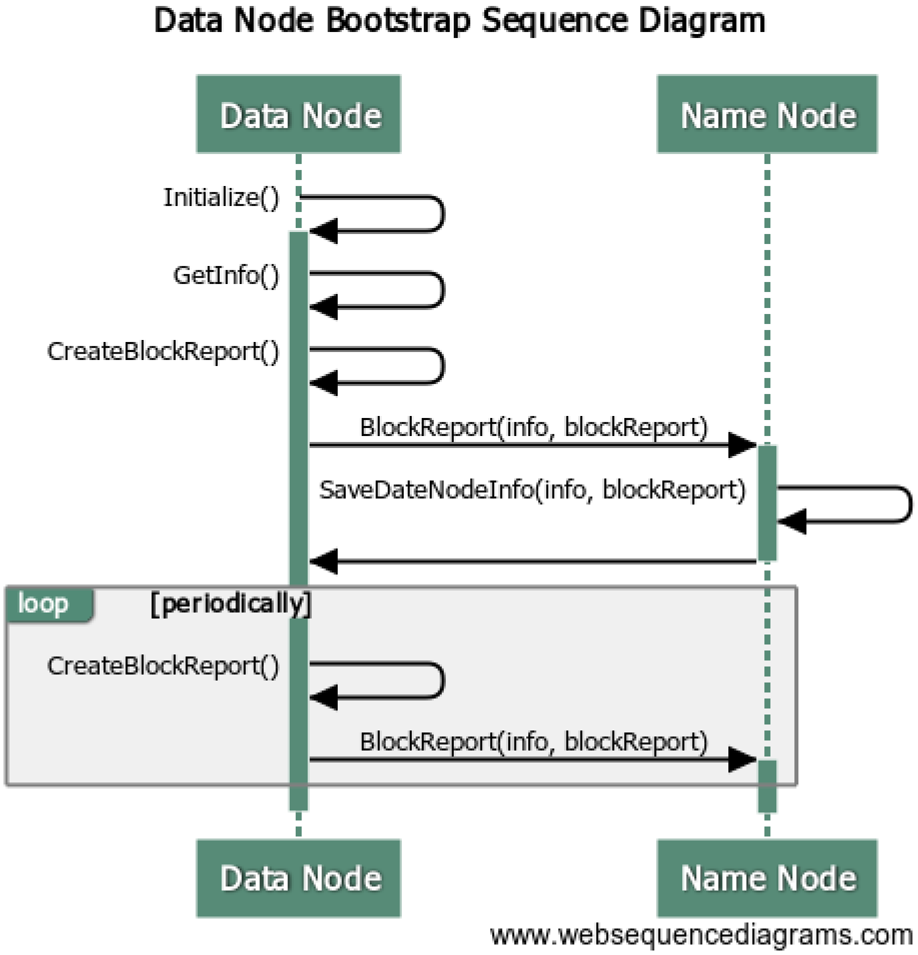
```
/path/to/CLI list-data-nodes <name_node_address_and_port> <file_name>
```

- name_node_address_and_port
    - address is required
    - :port is optional
    - ex: "10.0.0.8", "10.0.0.8:8080"
- file_name
    - the name of the file in SUFS

System Design


Create File Sequence Diagram
User
CLI
S3 Bucket
Name Node
ReceivingDataNode
CreateFile(name, S3-URL)
GetFile()
File
CreateFile(name, size)
DataNode List / BlockID
ChopFileIntoBlocks()
loop
[per BlockID]
DetermineDataNode(DataNodeList)
StoreAndForward(Block, BlockID, DataNodeList)
Result
loop
[per DataNodeListEntry Not Storing BlockID]
NextDataNode becomes the ReceivingDataNode for next loop iteration
StoreAndForward(Block, BlockID, DataNodeList)
NextDataNode
User
CLI
S3 Bucket
Name Node
ReceivingDataNode
NextDataNode
www.websequencediagrams.com


Get File Sequence Diagram
User
CLI
Name Node
DeterminedDataNode
GetFile(name, saveLocation)
GetFile(name)
DataNode List / BlockID
loop
[per BlockID]
DetermineDataNode(DataNodeList)
GetBlock(BlockID)
Block
ConcatenateBlocks(blocks)
SaveFile(data, saveLocation)
Result
User
CLI
Name Node
DeterminedDataNode
www.websequencediagrams.com


Data Node Bootstrap Sequence Diagram
Data Node
Name Node
Initialize()
GetInfo()
CreateBlockReport()
BlockReport(info, blockReport)
SaveDateNodeInfo(info, blockReport)
loop
[periodically]
CreateBlockReport()
BlockReport(info, blockReport)
Data Node
Name Node
www.websequencediagrams.com

Technologies and Tools Used

- Go
    - Using `http` library for all REST calls
- Git
    - Storing all code, documents, and images in a private repo
    - GitHub Repository

System Parameters

- Block Size: 64MB
- Replication Factor: 3
- Total DataNode: 4

Project State

Completed

- System design
- Tools and technologies decided
- Basic implementation of creating and getting files
    - CLI
    - Name Node
    - Data Node

In Progress

- Block Report
    - timeouts in Name Node
    - send block report from Data Node

Needs to be started

- Heartbeat from Data Nodes to Name Node
- AWS Setup
- End-to-end testing