

Mimir

Design Review Document

Ryan McGovern

Chris Siebert

Advisor: Fred S. Annexstein

Goals

To create a distributed file storage and retrieval system capable of maintaining data integrity with 50\% data loss when total data storage is a factor of four larger than original file size. The system should also be capable of working at speeds fast enough to allow for real time use by a user.

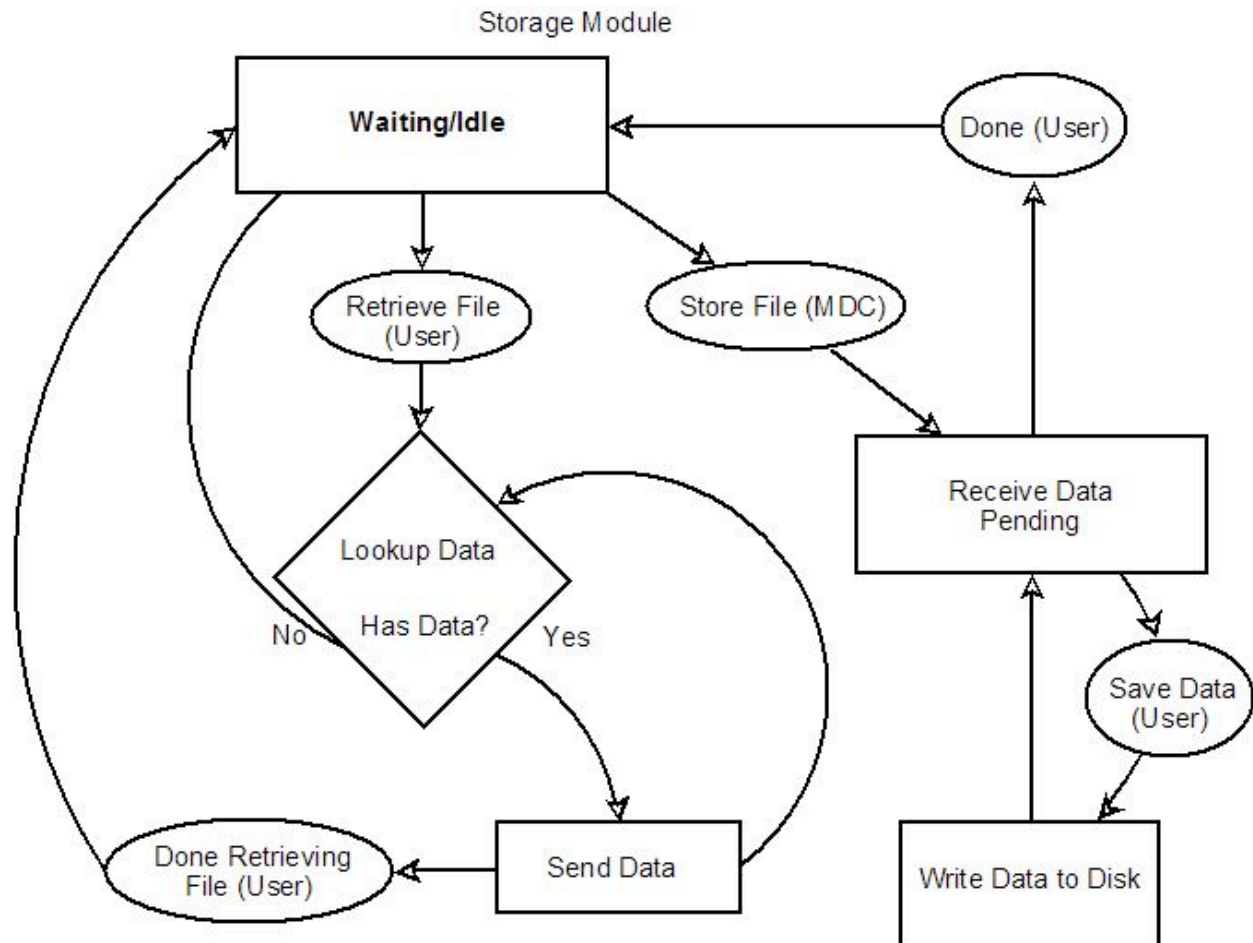
Design Specifications

Our project is will have three main parts, a Storage Module, a Client Module, and a Meta Data Controller(MDC). In the following diagrams we define how the first tow of these will interact with the network. Since our project uses Pastry we must conform their programming methodologies, however it does allow us to forgo creating a robust and adaptable peer to peer network. Also note that the MDC is not diagramed because the logic for how the MDC will service requests, as well as the crypto designs have not been fully flushed out.

- Rectangles are States or Actions taken by the application being represented.
- Ovals are incoming communications from other applications (source in parenthesis).
- Diamonds are application decision points based on internal data.
- Diagram labeled Network Member is applicable to all network applications in this project.

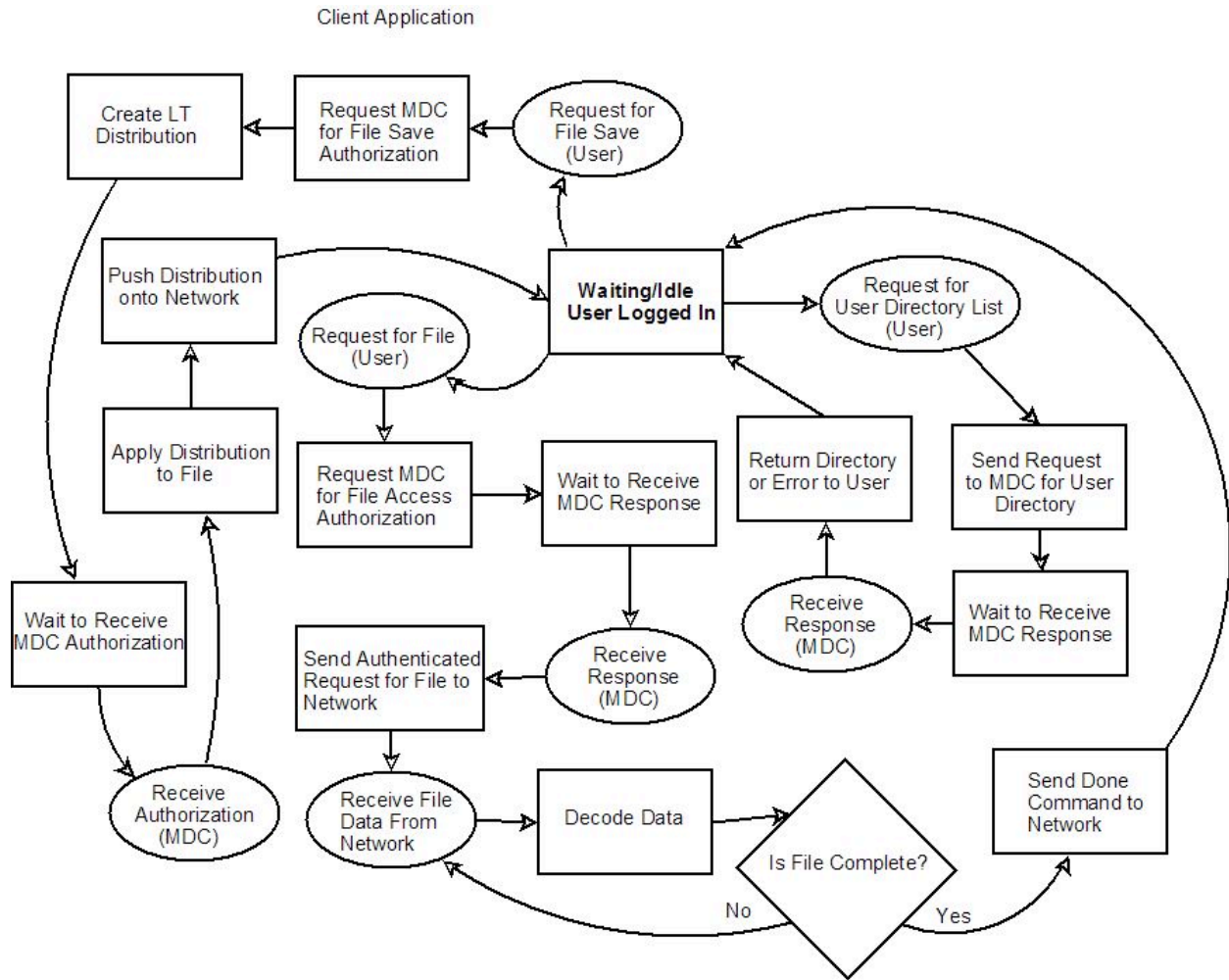
Storage Module

The storage module has only two main tasks to receive data from the network and save it to disk, and to respond to requests for user data by sending it to the requesting node.



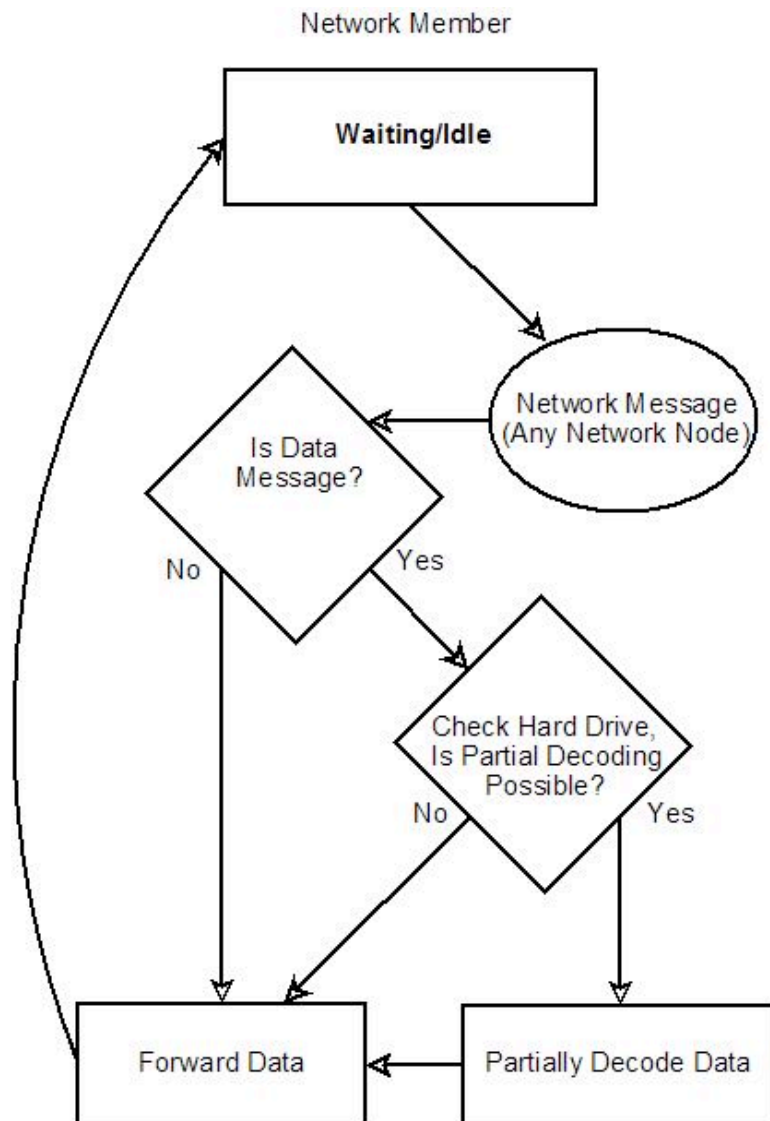
The Client Module

This module is responsible for performing all user initiated tasks such as listing a directory, requesting a file from the network, and saving a file onto the network.



All Network Nodes (Client, Storage, MDC)

This is the callback that is invoked when any message is traveling through the current node. Here is where we are able to do any parallel decoding for data retrieval messages.



Tasks and Milestones	Start	End
Implement Network Interface		Feb 11
Implement Client Network Interface	Dec 10	Feb 11
Implement Storage Node Network Interface	Dec 10	Feb 11
Implement Metadata Network Interface	Dec 10	Feb 11
Implement User Interface		Feb 11
Configuration management	Feb 1	Feb 11
User Storage Interface(FTP)	Jan 20	Feb 11
Metadata Administrator Interface	Feb 1	Feb 11
Metadata Controller		Feb 11
Create metadata database	Jan 1	Jan 21
Implement data maintenance routines	Jan 21	Feb 11
Implement Security systems	Jan 14	Feb 11
Storage Node		Jan 14
Implement storage management functions	Jan 1	Jan 14
Digital Fountain Codes		Feb 18
Create Distribution	Jan 1	Jan 30
Create Distribution Tests	Jan 1	Jan 18
Implement Encoder	Feb 11	Feb 18
Implement Decoder	Feb 11	Feb 18
Test Suite		Mar 24
Implement Unit Tests	Feb 11	Mar 1
Implement Functional Tests	Feb 11	Mar 1
Implement Data Integrity Tests	Feb 29	Mar 24
File Manipulation		Feb 25
Implement File Retrieval	Feb 11	Feb 25
Implement File Storage	Feb 11	Feb 25

Current Progress

The original goal of our distribution was to be capable of reliably sustaining 50% losses with a storage requirement of four times the non-encoded file size. This means that with a file of size n we should always be able to rebuild after receiving $2n$ encoded blocks. Our progress of adapting the Luby Transform to our project currently rebuilds successfully after receiving $1.25n$. We have also implemented a 'spike' optimization designed by Helsinki University of Technology which has reduced the standard deviation of the average required data by up to 80%, which improves the reliability. The result is that 99.74% of all distributions decode using less than $2n$ of data. It is our goal by the end of the month to have 100% of the distributions decode using less than $2n$ of data.

Progress Assessment

Our project is on schedule and possibly ahead of schedule for completing all objectives. This is due to discovering that pastry provides both a persistent storage class as well as a in memory cache class. Both of these meet our requirement that users can set a maximum amount of space to be used by the system in memory and for storage. In addition we were able to do much of the research into implementing the Luby Transform over break including some investigation on ways to improve it for our application.