README - Distributed Systems Project

**File System Features**

From the 7 available options:

* Distributed Transparent File Access(Mandatory)
* Security Service
* Directory Service
* Replication
* Caching
* Transactions
* Lock Service

The following 4 features were incorporated into the distributed file system for this project:

* Distributed Transparent File Access(Mandatory)
* Directory Service
* Caching
* Lock Service

**Starting the File System**

A batch file called ‘StartProject.bat’ has been created to start up all the necessary servers for this project in one simple click. This contains a list of commands to open all the servers in separate command windows. Each new command window is labelled for identification purposes and runs on the local machine.

The two main commands that are offered for a given client are: Read & Write. These are the main operations carried out by most clients of basic file systems.

For example:

**To Read**

In order to Read a file from the database, the client must type: read=<filename>

Where the <filename> describes the name of the file they want to read (e.g. ‘test.txt’).

**To Write**

In order to Write a file to the database, the client must type: write=<filename>

Where the <filename> describes the name of the file they want to write (e.g. ‘test.txt’).

This simple message protocol is defined in general terms as: <command>=<filename>.

Explanation of Main Components:

Storage File Server:

o   If a request is made to access a file from the client proxy, the file server sends a request to the lock server with the file ID and the client ID. The lock server will respond to let the file server know if the requested file is available to that client (Is it locked? If so does that client own the lock?) If available the file server will carry out the client’s request, and respond to the client

Lock Server:

o   Will receive requests from the file server containing a file ID and a client ID. The locking server keeps track of which files are locked and who owns the lock. Only a client who owns a lock may read/write to the file or unlock it. The lock server will respond to the file server if that client can have access, or if the file is unlocked it will make the client the owner of the locker. The locking system implemented is a simple read/write lock, which doesn’t distinguish between read and write requests. To improve this, a separate read and write lock system could be incorporated.

 Client-Side Cache:

o   A local cache to the client will keep a copy of the more recently accessed file. It will also keep track of the timestamp (when they were updated). This is important for consistency, as when a request is made to read a file from the cache, the timestamp must be compared to the timestamp on the original file (i.e. ‘last modified’ field in file system). If the original file has been updated by another client after the cached file was, this cached file is no good and the most recent file must be sought.

Client-Side Proxy:

o   The proxy helps to maintain transparency by handling requests from the client. It confers with the cache if a copy is available and obtains a timestamp from the file server to compare to the timestamp on the copy. If the copy is old the proxy will then request the original file from the file server. The proxy acts as a gateway for the client to access each of the servers available in this distributed file system.