**General Client Flow**

1. Client refresh timeout expires.
2. Client FileNodeMonitor scans directory tree for directory/file changes. Client asks server for list of updated files.
3. Modified files are added to a watch list.
4. When a file in the modified-file watch list hasn’t been modified for a while (we can work with the file without user interruption), ask the server for changes to that file (update before commit). If the files has changed, backup the file and then Rsync the server changes into the file without destroying the new content (is this possible? Actually, it seems unnecessary since the modified blocks are already on the server; that’s the whole point of rsync afterall. But how do we retain and merge client data?).
5. When server file changes have been merged, segment the file and send it to the server. Segment one file at a time. When the segmenter has finished segmenting the file, the segment list is sent to the assembler and the segment report (bit packed version of the segment list) is sent to the server.
6. Server hashes the segments and segments the server file version. Server compiles instruction list containing server data segments and client segment IDs (when a match is found). Packed reassembly instructions are sent back to the client.
7. Client reassembles file, per the server instructions, into a stage.
8. When the stage is completed, the Assembler checks whether the file on disk has been changed locally since it was segmented. If it has, repeat #4 in the opposite direction (wait until the user is no longer modifying the file; note that this also naturally means that we’ll have to send the updates to the server). Otherwise, lock the file and push the stage to disk as quickly as possible.

Other thoughts:

* To find out about changes, client has to ask server. Server replies with list of changed file ids. If the client doesn’t have the changed file, it asks the server for the entire file. If the client has an old, unmodified version of the file, it initiates a classic rsync transaction.
* If the client version of the file has been updated, but the server version has not been modified, the client initiates a reverse version of the rsync transaction.
* What do we do if both files have changed? Do we need to enforce in order updates (i.e. the client that modified logically first will merge with an earlier version of the file, which is then merged with more logically-recent updates from another client?)