Documentation of File Caching Proxy

Clients-Proxy Protocol:
The proxy emulates C file operation semantic by managing a data structure named Descriptors and implements open-close session on proxy side by another structure named FileStruct. The descriptors is actually an array where each entry stores a FileStruct. When a client asks to open a file, the proxy create a FileStruct instance, where stores the mode, version, pointer to original file, pointer to copied file, etc. And assigned that pointer to the *LOWEST FREE* entry in the descriptors, then return that number to client. So when next time the client wants to read or write a file, it passes that number and proxy fetch the corresponding pointer from Descriptors.

Open Close Session:
The proxy realized open-close semantic by replicate files in the cache. It made a directory for each clients with a unique name by keep tracking of how many clients connected to the proxy. For each WRITERS, they will have a replicated file for writing. After writing, the file will be directly pushed back to the server WITHOUT changing the cache. For the readers, the proxy will duplicate the original file in cache AT LEAST ONCE(I think straightly use the original file as reading file is a bad idea, the situation can be more complicated if that file need to be updated during reading), but will not be replicated again if another reader ask for the same file and has same version. All the read-file-replicas are stored in a unified directory.
All the temp file created for the clients will be cleaned up once all the related operation is done(Or at least try to clean up) to save cache space.

Conflict Naming Resolution:
Sometimes the files the clients want to fetch may have the same name. If same-naming is spotted when proxy tried to push file in the cache, it will be automated renamed. And the renamed file together with other information(such as the reference count, version, etc) will be stored in a FileInfo metadata, which is mapped to a simplified path name that the client originally passed to the proxy. This FileInfo meta-data is also used for cache eviction queue as well.

Eviction policy:
As LRU is the policy we need to implement here, a queue named cache_queue is created to store the original path name of the file. The field rfcnt will keep tracking how many users are using this file. And the file will only be add to the front of the queue only if the last user close the file. And each time the proxy need to evict files, the last one in the queue is deleted. If no file is available for freeing, the proxy throws EBUSY exception.

Server-Proxy Protocol:
The server keep a directory for each connected proxies to avoid blocking a file too

long. And it has a maximum bytes transferring for one RPC call. The server did that by let every proxy which want to connect call the connect() RPC first. In that way, the server can assign each clients(proxies) a unique service id. And then the server use this service_id to create sub directories for each clients for tempfiles.
The proxies also need to hand over that service_id for following operations, such as transferring file or pushing file. So the server can use that service_id and go to the specific directory to copy or overwrite the file for the proxy.