Serialization Protocol

My RPC protocol uses a simple, length-prefixed binary format:

Request Structure (Client to server):
4-byte Length: Total message length (in network byte order), which equals the payload length plus 1 byte for the operation code.
1-byte Operation Code: Indicates which file operation is requested (e.g., OPEN_CODE, READ_CODE).
Payload: Operation-specific data (serialization) . For example:
open(): [4-byte file name length][4-byte flags][4-byte mode][file name (null-terminated)]

Response Structure (Server→Client):
Deserialziation specific to operation (parsing) on server side
4-byte Result Code: If negative, indicates an error; the subsequent 4 bytes contain the error code (errno).
Otherwise, for successful operations the result code contains the result:
For operations like read() and write(), the result code indicates the number of bytes read/written.

Directory Tree Serialization:
The directory tree is serialized recursively with:
4-byte Name Length
Name String: The directory name
4-byte Number of Subdirectories
Recursive Serialization of Each Subdirectory
This design was chosen for its simplicity and ease of parsing.

Other Design Decisions
Local FD Mapping:
Since the remote server's file descriptors are not meaningful on the client side, we maintain a mapping between a "local" FD (assigned by our library) and the actual remote FD returned by the server. This abstraction simplifies the client's interface.