Assignment 6 Part 2
Leslie Xie | Bojun Lin

Design:

serial.h is essentially a library with the following functions:

char* serialize(StringArray* val)
char* serialize(DoubleArray* val)
char* serialize(Message* val)
StringArray* deserialize_sarray(char* val)
DoubleArray* deserialize_darray(char* val)
Message* deserialize_msg(char* val)

These functions encapsulate the requirements for being able to serialize and deserialize the classes outlined in supported.h.

Overview:

Serialization functions use a buffer to load fields into a char*, with each field separated by a '%' symbol. This symbol is used later in deserialization to be able to separate fields that were of the same type (size_t) and preserve the original values. The way fields are loaded into the buffer are handled based on their types and their neighbor's types (Message has three size_t fields after the MsgKind field, which we can load quickly using printf formatting). Sometimes intermediate buffers are created, loaded with data, then strcat tacks them onto the main buffer, based on the fields. We attempt to keep fields in order of their declarations.

The serialize function for the StringArray* uses the String pointer in its field to access the size_ and cstr_ member fields, and casts these to char* by using sprintf for size_ and memcpy for cstr_ to the buffer.

The serialize function for the DoubleArray* accesses the double* field and uses sprintf to cast its value to the buffer.

The serialize function for the Message* classes handle them by a type-by-type basis (Ack, Nack, Put, Reply, Get, WaitAndGet, and Kill function as the same, since they don't have differences in member fields).

Use cases:

The use cases are fairly simple. We make a StringArray, DoubleArray, and Messages of each type (other than Ack, Nack, Put, Reply, Get, WaitAndGet, and Kill), and load them with arbitrary data in their fields. We serialize them, and then deserialize them.