Protocol/Method for Communication between the Simulator and Arbiter

We have three programs that communicate with each other: the simulator, the arbiter, and the entity. The simulator communicates only with the arbiter, and the entities also only communicate with the arbiter. For the purposes of MVWT II, the testbed replaces the simulator. Our interface, therefore, should communicate with the arbiter using the same protocol as the simulator.

The simulator communicates with the arbiter through use of its Networking and Server classes. Its method of communication is to send messages in the form of strings of bytes, which are sent using the function

```
bool Server::send( int clientIndex, void* message, int size);
```

There is a similar Server::receive function for receiving data. The message is given in the form of a void pointer, and the send function converts the message to a char string. The same process is run in reverse for the Server::receive function. The receive function receives a char string and converts the string to a void pointer.

The Server functions are called by the Networking class, particularly in a function

```
void Networking::initialize(PhysicsWorld &pw);
```

that sets up the network connections. This function sets up a struct RawVision that is sent as the argument to the message (which can handle any class, as the send function takes a void pointer.) There are two other structs, SimulatorCommands and ObjectCommands, which are used to take inputs (in particular, velocities) from the arbiter. For our interface, we need to copy the RawVision, SimulatorCommands, and Object Commands structures from the simulator, and use those structures to receive and send data; we don't need to do any extra formatting. We should probably also continue to use (with small modifications) the Networking and Server classes.