The client will calculate all commands needed to process a given set of input parameters

Then the client will compose a single message with all of the commands and send this message to the middleware.

Command message is formatted as follows:

[ID ][# messages][message index][robot ID][command body (format below)]

[4 bytes][4 bytes ][4 bytes ][variable][variable ]

The standard header comes first. It is followed by the total number of messages being sent, since it may be the case that the command message is too large for one packet. The function of the index is to allow for messages to be re-ordered if they are sent out of order. Once all of the messages have arrived, the server can then parse the body into a queue of commands to be sent to the robots.

Command body is formatted as follows:

[command length][command text][command length][command text]....[last command text][NULL ]

[1 byte ][variable ][1 byte ][variable ]....[variable ][1 byte]

The command body consists of the length of a command, followed by the command with that length. The end of the body is terminated with a NULL byte which follows the last command. This would be interpreted by the middleware as the length of the next command. Since this length is 0, it is a clear indication that there are no more commands.

The purpose of sending all the commands at once is so the middleware can control the sending and receiving of data to and from the client and the robot in an efficient manner. Previously, the client controlled all of this in a stop and go pattern.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The middleware controls message flow between the client and the robots. Because of this, the client does not know what data it is receiving from the middleware. To allow for the client to correctly identify the type of data being received, an extra field must be added to the header of messages sent to the client:

[client ID][# UDP messages][message index][command index ][message data]

[4 bytes ][4 bytes ][4 bytes ][4 bytes ][variable ]

[ header defined in former protocol ][our addition ][ ]

A command index of *n* would indicate the response to the nth command sent to the client.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Formerly, the client handled the calculations for the wait times needed for MOVE and TURN commands. Since the server will be handling these commands now, and the server does not know the length or number of sides requested, more information must be added to the MOVE and TURN commands that are sent between the client and the middleware.

Former MOVE and TURN formats:

MOVE [velocity]

TURN [velocity]

New MOVE and TURN formats:

MOVE [velocity] [length]

TURN [velocity] [degrees]

Just as before, the client will be responsible for determining the length and degrees needed by each command.