Explanation of System Responsibilities

For: Network Library

Created by: Josh Lee on 11/30/15

Last Modified by: Josh Lee on 1/24/16

# Overview:

The Network Library is meant to allow other systems the ability to send and receive data across a network as well as act as a regulated entry point for all network traffic. The Network Library has three primary responsibilities:

1. Receive raw data from the attached socket and parse it into the appropriate message
2. Notify the appropriate subscribers when a pertinent message has been received
3. Mediate the sending of messages by the other attached systems

# Responsibilities of Current Interfaces

Currently the Network Library contains 4 interfaces. Please note that all interfaces in this library should not be modified unless absolutely necessary and only if express permission is given from a system architect.

## Message

Message is an interface that all other “messages” that can be sent from and to the system must implement. The Message interface guarantees that all messages implement methods that allow the system to get basic information from the message that will allow identification of the specific type of message, where it was sent from or where it should be sent, and access to the raw unparsed message.

## ExchangeSubscriber

The ExchangeSubscriber is an interface that must be implemented by any objects that wish to subscribe and receive information from the SubscribableExchange. The ExchangeSubscriber forces the inheriting class to implement methods that will allow the ExchangeSubscriber to check if the subscriber is interested in the received message and allow the ExchangeSubscriber to add the message to that subscriber’s message queue.

## MDS\_Interface

The MDS\_Interface is the general interface used by the message delivery systems. This interface forces all message delivery systems to implement a single method that allows a Message to be queued for transmission over the network.

## SystemInterface

The SystemInterface is a general interface that is inherited by all subsystems and is used for starting and stopping the subsystems.

# Responsibilities of Current Classes

Currently the Network Library consist of 3 concreate classes. Two of these classes deal with message distribution and sending messages over the network while one deals with parsing and encoding messages.

## MessageDeliverySystem

The MessageDeliverySystem or MDS is a singleton that is meant to act as the central entry point for network communications. The responsibilities of the MDS are:

1. Create and maintain a thread for the system to run on and stop the thread when requested.
2. Maintain the network socket attached to the MDS.
3. Safely destroy the attached network socket when a new socket, or a null, is attached to the MDS through the use of the += operator or the attachInterface method, or when the singleton is disposed of.
4. Provide a callback/handler method that can be passed to the socket that will take in the raw message and the address of the sender, and create the correct message and notify exchange subscribers that new data was received.
5. Regulate the transmission of messages over the network from different systems.

## SubscribableExchange

The SubscribableExchange is a concreate class whose responsibilities are:

1. Allow classes that implement the ExchangeSubscriber to subscribe and receive updates from the exchange.
2. Keep a list of all current subscribers.
3. Update subscribers when new information relevant to the subscriber is received.

## OpcodeOnlyMessage

The OpcodeOnlyMessage is a concreate class used to encode a single integer value, aka opcode, to a raw byte message or take a raw byte message and parse it to its opcode. For details on formatting see the network protocol document.

## MotorDir16Message

The MotorDir16Message is a concreate class used to encode what direction the attached motors should rotate to a raw byte message or take a raw byte message and parse it into the direction the attached motors should rotate. The MotorDir16Message stores the direction information in a 16bit value with each motor represented by two bits. This allows the message to control up to 8 motors. For details on formatting see the network protocol document.

## SetSpeedByteMessage

The SetSpeedByteMessage is a concreate class used to encode the information for the speed that a certain motor should run at to a raw byte message or take a raw byte message and parse it into the motor’s number and the speed it should be set to. This message has an 8bit resolution for its speed setting and can support up to 65535 motors. For details on formatting see the network protocol document.

<NOTE: As of 1/24/16 the network protocol document has been rewritten and, while not fully complete, contains all the commands for manual control. Please refer to it when working on any network messages for the message’s structure.>