**Message Specification**

The message specification must define the requirements of the message packets which are to be sent between nodes within the distributed system. These messages will be created, sent and received by the Message Passing and Queuing layer.

**Assumptions:**

The Message Passing and Queuing Layer will sit upon the IP Layer. This enables existing IP technologies to be employed for error checking, data retransmission and routing.

The Message Queuing/Passing Layer will utilise a publisher/subscriber to some extent.

**Requirements:**

The messages must be able to be sent asynchronously and must be able to be queued.

The messages must be able to be routed using the IP Protocol.

The messages must be able to send multiple or individual variables.

The messages must support procedure/data advertisement or listing.

The messages must be variable in size.

The client sending the messages should not rely on a reply once the message has been sent.

The messages must contain enough information for remote procedure execution w/ parameter passing.

The messages must be able to contain the results of remote procedure calls.

They must be easy to construct and deconstruct.

The data should be able to be validated using CRC or similar.

The messages should be able to be used in a publish/subscribe model.

**Beneficial features:**

A solution in which the contents of the messages are somewhat human readable for ease of debugging and construction(ie. XML or JSON)

A solution which would be compatible with existing message queuing and passing libraries (ie. ZeroMQ or RabbitMQ)

**Research:**

Investigation into existing Remote Procedure Call libraries for Python led to the discovery of two technologies.

[1] ZeroRPC

And

[2] RabbitMQ RPC

Zero RPC is a thin and reliable library for distributed communication between server-side processes. It builds on top of [ZeroMQ](http://www.zeromq.org/) and [MessagePack](http://msgpack.org/).

[Note: MessagePack is a library which handles variable size message sending using compressed JSON]

One of the interesting features of ZeroRPC, besides it’s speed, low overhead and vast number of existing features is the support for streamed responses from procedure calls. This could be hugely beneficial to the ROVER project considering the vast quantity of data which might need to be retrieved at various points in time. ZeroRPC also has built in support for method/procedure advertisement when probed.

RabbitMQ RPC shares the same remote procedure execution principles as ZeroRPC, however it is more of a “shoe-horned” in feature which is not a fully supported library such as ZeroRPC. Streamed responses are not supported by default, and method advertisement is not available by default.

**Chosen Solution:**

ZeroRPC has been chosen as the platform. This is because it fully supports the requirements as laid out in the Message Specification. It integrates well with existing Message Passing Technologies (ZeroMQ) and meets all functional requirements listed. The utilisation of ZeroRPC also makes the Message Layer almost invisible to the programmer as the perception of ZeroRPC is simple method execution on a remote machine.

**Deadline:**

There will be a timeframe of 4 weeks for the subgroup to complete the requirements for the message specifications, and to combine it with the message passing and queueing subgroup to create the finalised high level communications system.