# **GETTING STARTED**

A reference implementation of the HELPER network stack is available as open source software hosted on Github (https://github.com/ANDROComputationalSolutions/HELPER-node). The reference implementation includes many components that were utilized in our implementation of the HELPER network stack including physical layer, MAC layer, and message structures. Baseline implementations of network layer, service layer, and applications are provided to allow for a complete network stack.

**Goal:** Out of the box, the reference implementation allows a user to transmit messages (packets) from source application to destination application across two HELPERs (within communication range) over-the-air. The developer can build on this reference implementation and develop his/her own networking protocols within the Helper network stack. The goal is not to share the exact same applications and algorithms that we have developed, but rather provide the platform that will encourage the development of new applications, algorithms and protocols that can be evaluated on our HELPER network.

## **Raspberry Pi Setup**

The reference implementation has been developed and tested using Raspbian Stretch Lite operating system. The Raspbian Stretch Lite operating system image is available for download at https://www.raspberrypi.org/downloads/raspbian/.

The Lora transceiver breakout needs to be wired to the Raspberry Pi. Information about the Lora transceiver breakout is available at https://www.adafruit.com/product/3072?gclid=Cj0KCQjwjN7YBRCOARIsAFCb936\_JUJftnJBQTzqC5m3BoLv6wjOvZYwLvghfHhElgVuzSA2p-92RuoaAmgqEALw\_wcB.

## **Software Dependencies**

The following are dependencies for the reference implementation: RadioHead, Bcm2835, ZMQ, and RapidJson. Instructions on how to install each dependency is explained below.

The RadioHead library is managed as a Git submodule within the HELPER-node repository. Instructions on how to initialize the submodule are in the following section.

The bcm2835 library is built from source on the Raspberry Pi using the commands below.

*wget www.airspayce.com/mikem/bcm2835/bcm2835-1.56.tar.gz*

*tar zxvf bcm2835-1.56.tar.gz*

*cd bcm2835-1.56*

*./configure*

*make*

*sudo make check*

*sudo make install*

The ZMQ library can be installed with apt-get using the below command.

*sudo apt-get install libzmq3-dev*

The RapidJson library can be installed using apt-get using the below command.

*sudo apt-get install rapidjson-dev*

## **Building the Reference Implementation**

Once the Raspberry Pi is configured as described above and the Lora transceiver module is properly wired, the user can begin using the reference implementation.

The user must download the source code from Gitlab, initialize the RadioHead git submodule, and then build reference implementation, which is shown in the commands below.

*git clone https://github.com/ANDROComputationalSolutions/HELPER-node.git*

*cd HELPER-node/*

*git submodule update --init*

*make*

If the build was successful, the following executables should be present in the current working directory: helperd\_benchmark, benchmark\_send\_text, and benchmark\_recv\_text.

## **Testing the Reference Implementation**

Two HELPERs are required to test the reference implementation. On each HELPER, the program helperd\_benchmark must be executed. The usage for helperd\_benchmark is shown below.

*sudo ./helperd\_benchmark <nodeid>*

The parameter node id assigns an identifier to the HELPER. The identifier is used for network and MAC layer addressing. The node id is an unsigned 8-bit integer which allows for 256 addresses in a HELPER network. For this test, simply assign the first HELPER node an id of “1” and the second node an id of “2”, as shown in the commands below.

*sudo ./helperd\_benchmark 1*

*sudo ./helperd\_benchmark 2*

Each HELPER will also run a text application. The first node will run benchmark\_send\_text and the second node will run benchmark\_recv\_text. On the second node, start the receive application as follows.

*./benchmark\_recv\_text*

On the first HELPER, start the sending application as follows. The command-line argument “2” is used to specify the destination node id.

*./benchmark\_send\_text 2*

Figures 1-4 provide example standard output when running these programs.

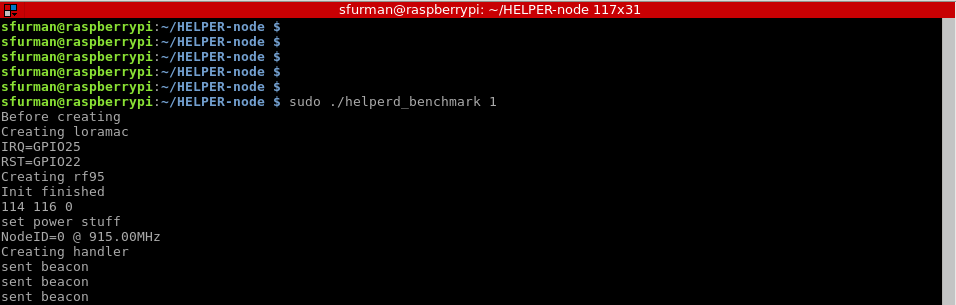


Figure 1. Example helper benchmark output

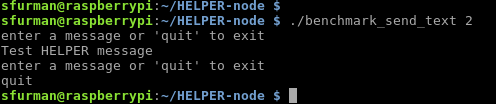


Figure 2. Example send application output

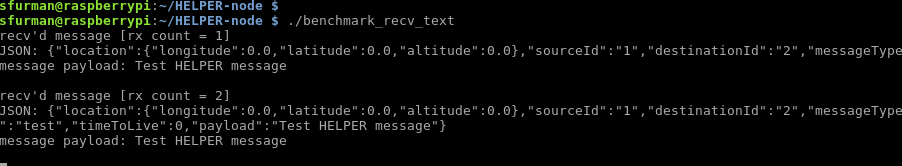


Figure 3: Example receive application output

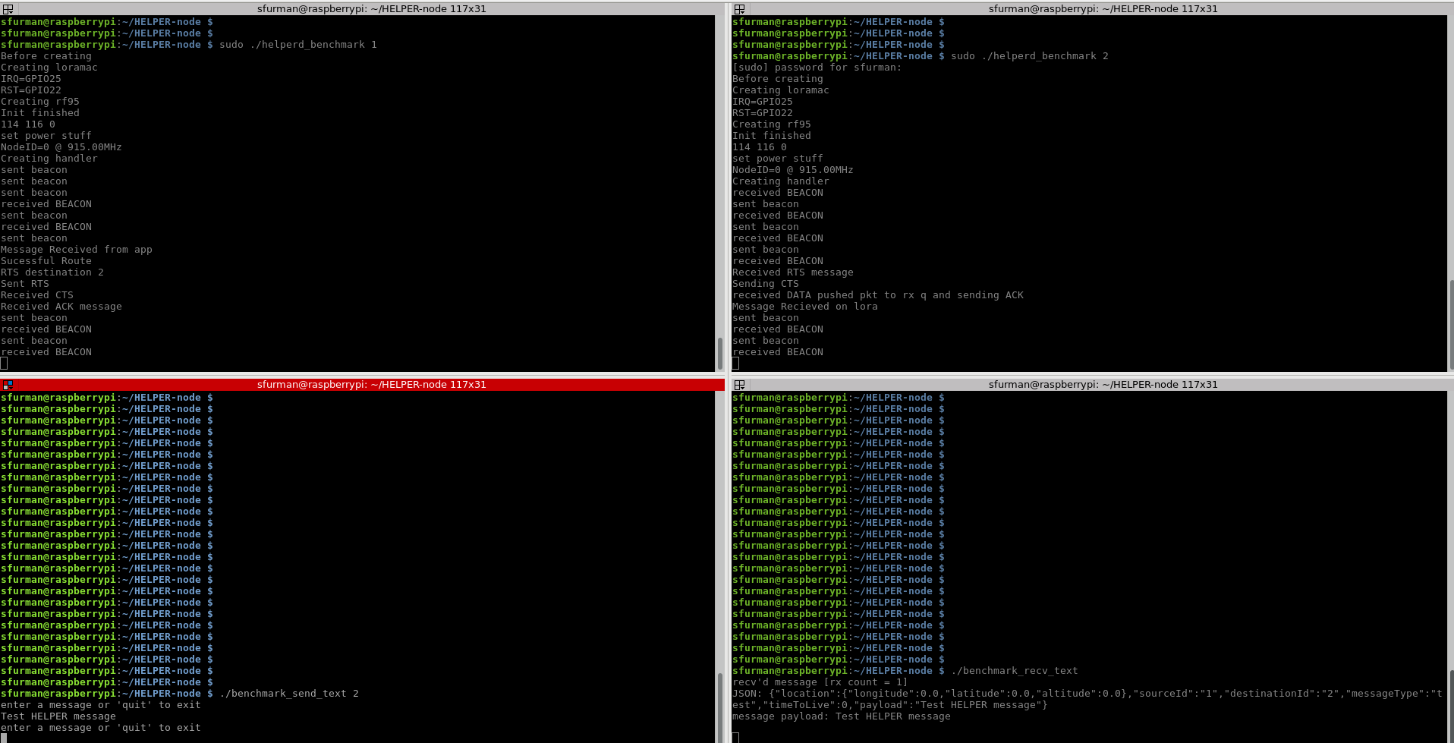


Figure 4. Testing Screenshot

## **Development Next Steps**

The developer can build off the provided reference implementation to implement networking protocols of his/her choice. The user can implement his/her own network layer by following the interface defined in the file RoutingApi.h, an example of this is shown in the files RoutingBenchmark.h and RoutingBenchmark.cpp. The user would need to modify the Makefile so that his/her routing implementation is used rather than the benchmark. The key functions in the RoutingApi to focus on are: send\_msg, mac\_recv, route, process\_beacon\_data, and write\_beacon\_data. The functions send\_msg and mac\_recv are where network layer packet fields are written/read and packets are queued. The route function is used to execute the routing algorithm implemented in the network layer. The process\_beacon\_data and write\_beacon\_data functions are used to share control information between the network nodes to facilitate distributed routing algorithms. Currently, control information fields (location, utility, battery life, queue lengths) are included in the helper packet structure.