Hello Protocol Project Guide

A guide to the download, use, and code of this project.

# Introduction

This project is a demonstration of the hello protocol that will be used in the routing protocol for neighbor recovery. So far, this project is able to create a Scenic Routing hello packet to send to a multicast group and receive acknowledgements back.

The project was created in Eclipse using C++ (C++11 and above) and tested on a linux machine set up with the same tools as the OpenWRT Build System machine, then tested again as a package cross compiled and run on an OpenWrt Router.

This guide will walk through the steps to downloading this project from GitHub, how to build and run the project correctly, and a summary of the classes and functions within the code.

# Download

1. Go to <https://github.com/crumbj/Scenic-Routing>
2. Click the green Code ↓ button
3. Either select Download Zip or Copy the SSH line. Enter the following line into the terminal where you want to download the folder to.

* git clone [git@github.com](mailto:git@github.com):crumbj/Scenic-Routing.git

1. The project will be located in the **Scenic Routing/Routing Protocol/Hello Protocol** folder within the downloaded project.

# How to Use

You can either run the program on your development machine or your OpenWrt router as long as you have set up your machine to use multicasting. If you have not done so already, please refer to “Multicasting on OpenWrt“ in the OpenWrt Documentation folder to do so.

### Run on OpenWrt

Copy the .ipk file and the configuration file from the “Hello Protocol Project” or folder to the /tmp folder of the router:

cd “Hello Protocol Project”

scp routerInterface\_1.0-1\_x86\_64.ipk root@<ROUTER IP ADDRESS>:/tmp

scp routerConfig.txt root@<ROUTER IP ADDRESS>:/tmp

From the root directory, install the package:

cd ..

opkg install /tmp/routerInterface\_1.0-1\_x86\_64.ipk

From the root directory run the command:

routerInterface

You can of course, copy the “Hello Protocol Project” to the buildroot directory on your build system to test out building and deploying your own package. You will need to rename the “Hello Protocol Project” folder to “routerInterface” unless you want to edit the makefile. Follow the steps in the OpenWrt HeloWorld documentation or OpenWrt Cross Compilation Guide for more information.

### Run on Linux machine

Open the terminal from the project folder and run the following commands:

cd Debug

make clean

make

./routerInterface

In order for this to work, in the getRouterIP () function in configUtil.cpp, you will need to change the device to the name of your machine’s device. This can be found by entering the command ip addr and looking for the device that has the IP address of your machine (not 127.0.0.1). It will often be something like **“eth0” or “eth1”**

# Configuration Code Explanation

The file configUtil.cpp contains functions used to help configure the router utilizing the routerConfg.txt file. In the code, this file is currently located in the tmp directory, but this can be changed by the user.

### Functions

**std::string getRouterIP ( )**

* This function utilizes the ifaddr struct to get the IP address of the current device (router). It is currently configured to use “br-mng” as the device. This can be changed to another device if the name of your router is different. It returns the IPv4 address of the router as a string.

**std::string getIPFromUnsignedInt (unsigned int ip)**

* This function converts an unsigned int to an IP address using bit shifting. This needs to be done because the IP address is stored in the graph nodes (routers, hosts) as an unsigned int, but we need to print it and include it in packet headers.

**void readConfigFile (Graph \*pcGraph)**

* This is where the configuration file is read in from /tmp/routerConfig.txt and into a graph of the network where the routers are the nodes. The file is in the format:

number of nodes

router1name ipaddress

router2name ipaddress

..

number of edges

leftrouterIP rightrouterIP

..

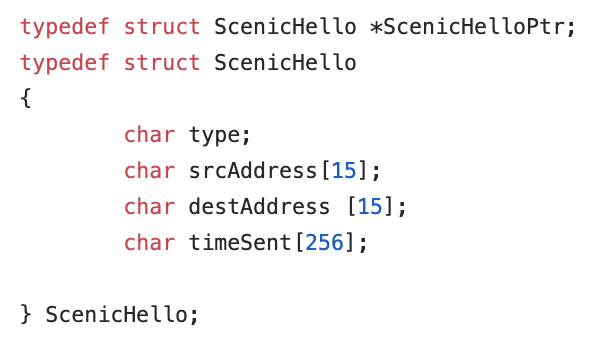
**vector<Node \*> getNeighbors (Graph \*pcGraph, std::string routerIP)**

* This function finds all the neighboring routers of the router who’s IP address is passed into the function and returns them in a list.

# Multicasting Code Explanation

## Hello Packet Construction

The file helloPacket.cpp defines the functions that construct a hello protocol packet. The packets are defined as follows:



type - refers to whether or not this packet is acknowledging another hello packet

srcAddress - the IPv4 address of the router sending the hello packet

destAddress - the IPv4 address of the router or multicast group receiving the packet

timeSent - the time the packet was sent.

### Functions

**constructPacket (ScenicHelloPtr pPacket, char \*pszSrcAddress, char \*pszDestAddress, char helloType)**

* This function takes in a pointer to a ScenicHello struct and sets its source address, destination address and type. The function itself determines the time in the timeSent value using the seconds and useconds attributes of a timeval struct to calculate the milliseconds, then converting that into a string.

## Multicasting Code

The file helloMulticast.cpp implements the functions needed in order to send and receive hello packets through multicasting ip datagrams. There are also the functions needed to implement multicasting with threads. Using threads allows for concurrency and for all threads to share access or update the information on how long it has been since the last acknowledgement from a neighbor. This was created with resources from <https://www.tenouk.com/Module41c.html>

### Functions

**void sendHelloMulticast (std::string routerIP)**

* This function constructs and sends a single packet to the multicast address. It does this by utilizing group sockaddr struct and a local interface.

**ScenicHelloPtr recvHelloMulticast (char \* routerIP)**

* This function listens on the multicast address for a hello packet, then returns the packet that was received.

**ScenicHelloPtr recvACK (char \* routerIP)**

* This function listens on its normal IP address to receive hello packet acknowledgements from other routers. These acknowledgements are not sent to the multicast address because they are for an individual router.

**void \* sendHelloLoop (void \* routerIP)**

* This function is made to be passed to a thread that will continuously loop to send multicast packets periodically. It currently sends them at an interval of 15 seconds. This interval can be changed in helloMulticast.h on line 14.

**void \* recvHelloLoop (void \* routerIP)**

* This function is made to be passed to a thread that continuously listens on the multicast address to receive hello packets, then on the reception of a hello packet, constructs and sends an acknowledgement to the source of the hello packet.

**void \* recvACKLoop (void \* routerIP)**

* This function is made to be passed to a thread that continuously listens for acknowledgment hello packets sent to its normal IP address.

## Startup Code

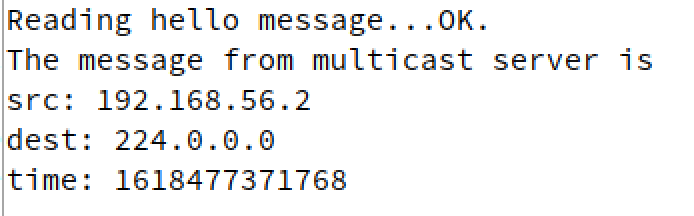
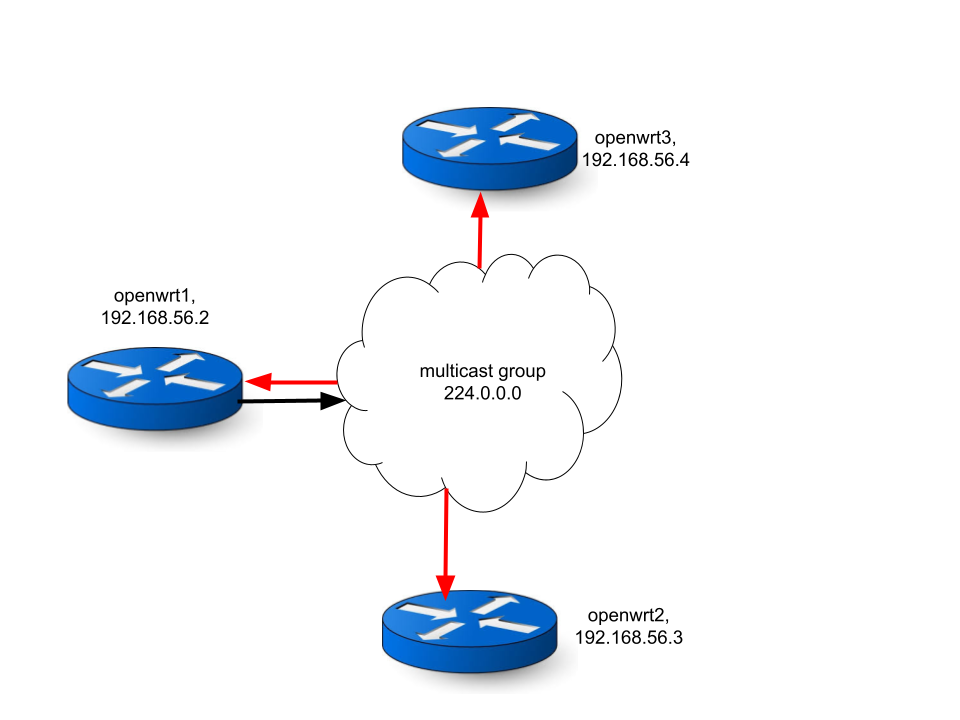
The main driver is in this file. Here are the tasks that this file completes:

1. Get the router’s IP address.
2. Read in the configuration file routerConfig.txt to set up the graph to be used by the routing algorithm and hello protocol.
3. Get a list of the router’s neighbors from the graph.
4. Start the threads used to concurrently send and receive hello packets on the multicast address and send and receive acknowledgements.

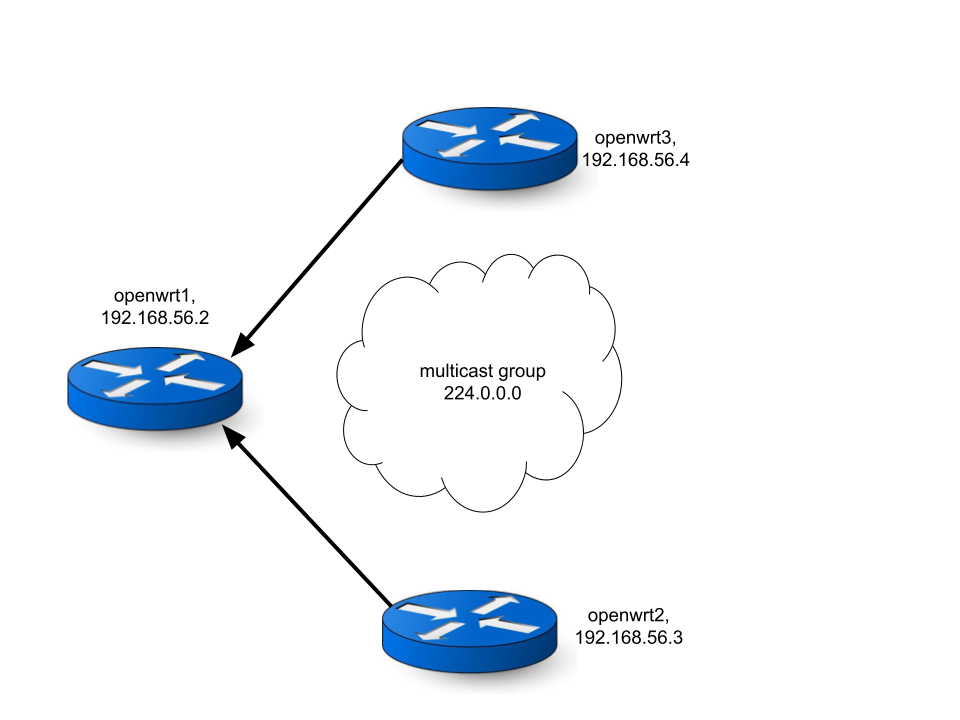
# Sample Output

Here are some diagrams depicting what is happening in the network with corresponding output from this program.

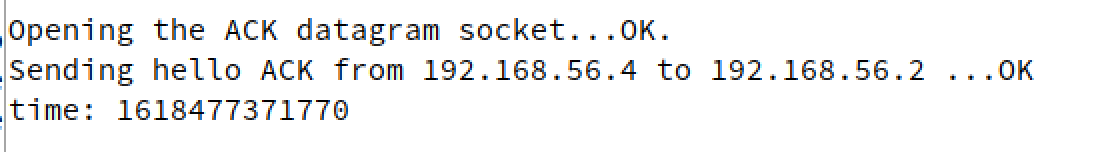
This first image depicts one router sending a hello packet to the multicast address (black arrow), and each router listening to the multicast group receiving the hello packet (red arrow) This is followed by the output from the program which is the packet that was sent to the multicast address. The source address is 192.168.56.2, which is the ip address of openwrt1 who sent the packet, and the destination is 224.0.0.0 which is the multicast address.

.

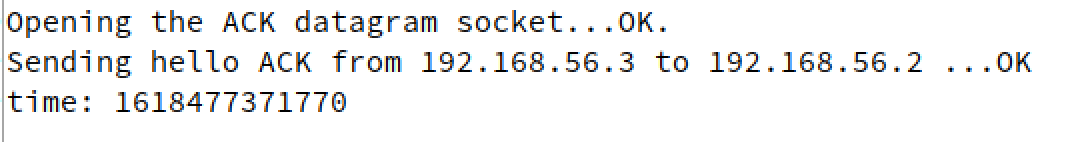
This next image depicts the two routers that received the packet sent to the multicast sending back acknowledgements to the source, openwrt1. This is followed by the output from the program.



Output from openwrt3 sending its acknowledgement to openwrt1.



Output from openwrt2 sending its acknowledgement to openwrt1



Output from openwrt1 receiving acknowledgements from both openwrt2 and openwrt3

