## Routing Protocol

#### Background Information

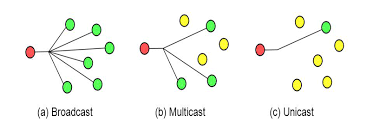
To fully understand the completed components of the Routing Protocol portion of Scenic Routing here is some basic information about routing, graph theory, and algorithms.

##### Routing

In computer science, routing refers to the process of learning all the routes, or paths through a network, and using them to forward data from one network to another or within the same network. This data is made up of a packet or several packets.

Packets can be routed through broadcast, multicast, or unicast.

1. **Broadcast** (One to All) - a single of a packet sent by a single sender to the broadcast address, which is listened to by all hosts in the network.
2. **Multicast** (One to Many) - a single packet sent by a single sender to the multicast address, which is only listened to by hosts in the multicast group.
3. **Unicast** (One to One) - a single packet sent by single sender to a single recipient.

[](https://www.google.com/imgres?imgurl=https%3A%2F%2Fradiocrafts.com%2Fwp-content%2Fuploads%2F2020%2F03%2Fmulticast-feature-image-webpage.png&imgrefurl=https%3A%2F%2Fradiocrafts.com%2Fwhy-is-multicasting-becoming-essential-for-mesh-networks%2F&tbnid=Tjzb2M-vT1cUHM&vet=12ahUKEwiriomS2fnvAhX_GDQIHcghAScQMygBegUIARDPAQ..i&docid=dOxzSlfeRaQU3M&w=803&h=292&q=multicast&ved=2ahUKEwiriomS2fnvAhX_GDQIHcghAScQMygBegUIARDPAQ)

[[photo credit](https://www.google.com/url?q=https://www.google.com/imgres?imgurl%3Dhttps%253A%252F%252Fradiocrafts.com%252Fwp-content%252Fuploads%252F2020%252F03%252Fmulticast-feature-image-webpage.png%26imgrefurl%3Dhttps%253A%252F%252Fradiocrafts.com%252Fwhy-is-multicasting-becoming-essential-for-mesh-networks%252F%26tbnid%3DTjzb2M-vT1cUHM%26vet%3D12ahUKEwiriomS2fnvAhX_GDQIHcghAScQMygBegUIARDPAQ..i%26docid%3DdOxzSlfeRaQU3M%26w%3D803%26h%3D292%26q%3Dmulticast%26ved%3D2ahUKEwiriomS2fnvAhX_GDQIHcghAScQMygBegUIARDPAQ&sa=D&source=editors&ust=1618966137751000&usg=AOvVaw1dipykeK3ZlDSUk84PID3P)]

A routing protocol is a set of processes, algorithms, and messages that are used to exchange routing information to determine the paths in the network, often the best paths. Routing protocols are used to facilitate the exchange of routing information between routers. [[link](https://www.ciscopress.com/articles/article.asp?p=2180210&seqNum=4)]

A routing protocol will build and update a routing table. A Routing table is a data table stored in a router or a network host that lists the routes to particular network destinations, and in some cases, metrics associated with those routes.

The basic components of a routing protocol are as follows. The bolded components on the list are what has been implemented thus far in Scenic Routing.

* **Routing Algorithm**  - Determines the path from one network node (router or host) to another.
* **Hello Protocol** - Ensures that the communication between routers is bidirectional by periodically sending packets to neighboring routers, then updating the list of router neighbors depending on if the packets it sends are acknowledged.
* Flooding Protocol - Updates all routers in the network with information about changes to each router’s neighbor list until all routers have the same information.
* Calculating Metrics - Determines how to calculate how expensive it is to forward packets from one router to another using throughput, bandwidth, reliability, etc.
* Routing Table Creation - Uses the neighbor list information, metrics, and routing algorithm to create a routing table.

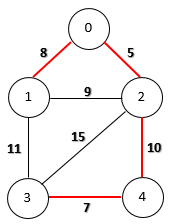
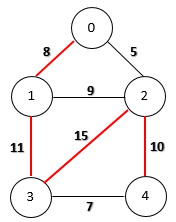
##### Graph Theory and Algorithms.

These are the graph theory concepts and algorithms used in the implementation of the Scenic Routing protocol’s routing algorithm.

###### Graph Theory Concepts

A computer network can be represented as a graph. To be more specific, a computer network can be represented by an undirected, cyclic graph. This means that all of the edges between nodes are bidirectional and it is possible that the graph has loops.

A minimum spanning tree is a subset of the edges of a connected, edge-weighted undirected graph that connects all the nodes together, has no cycles, and uses the minimum possible total edge weight. A maximum spanning tree is the same except it uses the maximum possible total edge weight. Below you will find the minimum spanning tree (left) and maximum spanning tree (right) for the same graph.

[[photo link](https://www.google.com/url?q=https://www.google.com/imgres?imgurl%3Dhttps%253A%252F%252Fwww.baeldung.com%252Fwp-content%252Fuploads%252F2019%252F12%252Fminimum_spanning_tree.png%26imgrefurl%3Dhttps%253A%252F%252Fwww.baeldung.com%252Fjava-spanning-trees-kruskal%26tbnid%3DisFbJCUqcD-F8M%26vet%3D12ahUKEwjG17imxvrvAhX5ATQIHRpbC6sQMygMegUIARDNAQ..i%26docid%3D2cYQbrUc52ya2M%26w%3D170%26h%3D224%26q%3Dminimum%2520vs%2520maximum%2520spanning%2520tree%26ved%3D2ahUKEwjG17imxvrvAhX5ATQIHRpbC6sQMygMegUIARDNAQ&sa=D&source=editors&ust=1618966137760000&usg=AOvVaw2y1amfklym3_3UcNCHyffT)][[photo link](https://www.google.com/url?q=https://www.google.com/imgres?imgurl%3Dhttps%253A%252F%252Fwww.baeldung.com%252Fwp-content%252Fuploads%252F2019%252F12%252Fmaximum_spanning_tree.png%26imgrefurl%3Dhttps%253A%252F%252Fwww.baeldung.com%252Fjava-spanning-trees-kruskal%26tbnid%3DqadmVShhqPVOZM%26vet%3D12ahUKEwjG17imxvrvAhX5ATQIHRpbC6sQMygGegUIARDBAQ..i%26docid%3D2cYQbrUc52ya2M%26w%3D176%26h%3D222%26q%3Dminimum%2520vs%2520maximum%2520spanning%2520tree%26ved%3D2ahUKEwjG17imxvrvAhX5ATQIHRpbC6sQMygGegUIARDBAQ&sa=D&source=editors&ust=1618966137758000&usg=AOvVaw0WpNWd5WG70ukLPZcjbHYY)]

###### Kruskal’s Algorithm

Kruskal’s algorithm is a greedy algorithm that finds a minimum spanning tree of an undirected edge-weighted graph. This algorithm works as follows:

1. Sort all the edges from low weight to high
2. Take the edge with the lowest weight and add it to the spanning tree. If adding the edge created a cycle, then reject this edge.
3. Keep adding edges until we reach all vertices.

###### Breadth First Search

Breadth First Search is an algorithm used to search or traverse a graph. It does this by starting from a root node (which can be any node in the graph) and exploring all of the neighbor nodes at its present depth before moving onto the next level of nodes. The algorithm stops when all of the nodes in the graph have been visited. This can be modified to stop when a specific node has been visited if looking to find a path between two nodes.

In the diagram below, the Breadth First Search algorithm would visit all of the nodes in the orange level, then the green level, and finally the blue level.

###### 

[[photo link]](https://www.google.com/url?q=https://www.google.com/imgres?imgurl%3Dhttps%253A%252F%252Fmiro.medium.com%252Fmax%252F4788%252F1*VM84VPcCQe0gSy44l9S5yA.jpeg%26imgrefurl%3Dhttps%253A%252F%252Fmedium.com%252Fbasecs%252Fbreaking-down-breadth-first-search-cebe696709d9%26tbnid%3DnuYYeP7XNiVPbM%26vet%3D12ahUKEwiR-Y_ByfrvAhVjAzQIHRSrBg0QMygBegUIARDPAQ..i%26docid%3DdNmI-Dfm3bcXcM%26w%3D1824%26h%3D1216%26q%3Dbreadth%2520first%2520search%26ved%3D2ahUKEwiR-Y_ByfrvAhVjAzQIHRSrBg0QMygBegUIARDPAQ&sa=D&source=editors&ust=1618966137763000&usg=AOvVaw1pT4JcH6PRdaHonJMDgdwA)

###### Graph Coloring Method

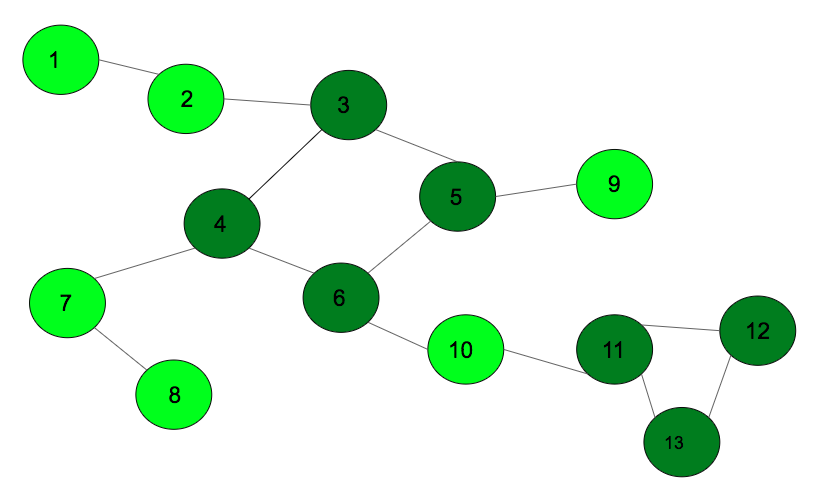
The Graph Coloring Method is used to find all the cycles in a simple, undirected graph. Coloring refers to marking the nodes based on if they are visited, partially visited, or unvisited using black, gray, and white respectively.

The algorithm works as follows:

1. Call Depth First Search traversal for the graph which can color the nodes.
2. If a partially visited node is found, backtrack till the node is reached again and mark all nodes in the path with a counter which is cycle number.
3. After completion of traversal, iterate for cyclic edges and push them into a separate adjacency list.
4. Print the cycles number wise from the adjacency list.

[[content and image source](https://www.google.com/url?q=https://www.geeksforgeeks.org/print-all-the-cycles-in-an-undirected-graph/&sa=D&source=editors&ust=1618966137764000&usg=AOvVaw2MB6VpZ4hRiGlL2jHWjFiZ)][[content source](https://www.google.com/url?q=https://www.geeksforgeeks.org/print-all-the-cycles-in-an-undirected-graph/&sa=D&source=editors&ust=1618966137764000&usg=AOvVaw2MB6VpZ4hRiGlL2jHWjFiZ)]

This image below uses the Graph Coloring method to detect cycles, the cycles noted in dark green.



#### What is the Scenic Routing Protocol?

The Scenic Routing routing protocol will be a link-state, Interior Gateway Protocol (IGP) written using C/C++. This means that the protocol program will run on every router within a closed, isolated network and dynamically change routes based on updates in the network. The protocol’s algorithm will aim to route the packets in the longest possible path to their destination while avoiding infinite loops.

#### Finished Routing Protocol Components

As mentioned before, the completed, fully documented components of the routing protocol includes the Routing Algorithm and the Hello Protocol.

##### Routing Algorithm

As previously mentioned, the goal of the Scenic Routing routing protocol is to route packets the longest route possible through the network. This path will be determined by the routing algorithm. However, finding the longest path is not as simple as finding the shortest.

In theoretical computer science, the longest path problem is the problem of finding a simple path of maximum length in a given graph, simple meaning that the path does not have any repeated nodes. The length of a path is measured by either the number of its edges, or by the total weight of its edges This is otherwise known as the “Travelling Salesman Problem” or the “Hamiltonian Cycle Problem” [[content source](https://en.wikipedia.org/wiki/Longest_path_problem)]

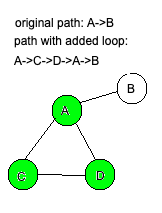
The longest path can be found of a graph that is directed and acyclic (has no cycles), but a computer network is not directed or strictly acyclic.

For this reason, our algorithm will not be truly the longest but “longish”.

###### Design

The design of the routing algorithm is a combination of Kruskal’s algorithm, Breadth First Search, and the Graph Coloring Method. By negating the edge weights of the graph, Kruskal’s algorithm will produce a maximum spanning tree. Breadth First Search will then be used to find the path from one node on the maximum spanning tree to another. This ensures that the path between the two nodes will be made up of the largest edge weights possible, therefore potentially having a larger total weight.

This design so far still leaves room for our “longish path” to be the same as the shortest path. For example, if the two nodes are adjacent to each other, even after the original graph is transformed into a maximum spanning tree, then both the longest and shortest path will be the same. This can be avoided if a loop is attached to the front of the path that includes the source node, but does not include the destination node. Take the following diagram for example:



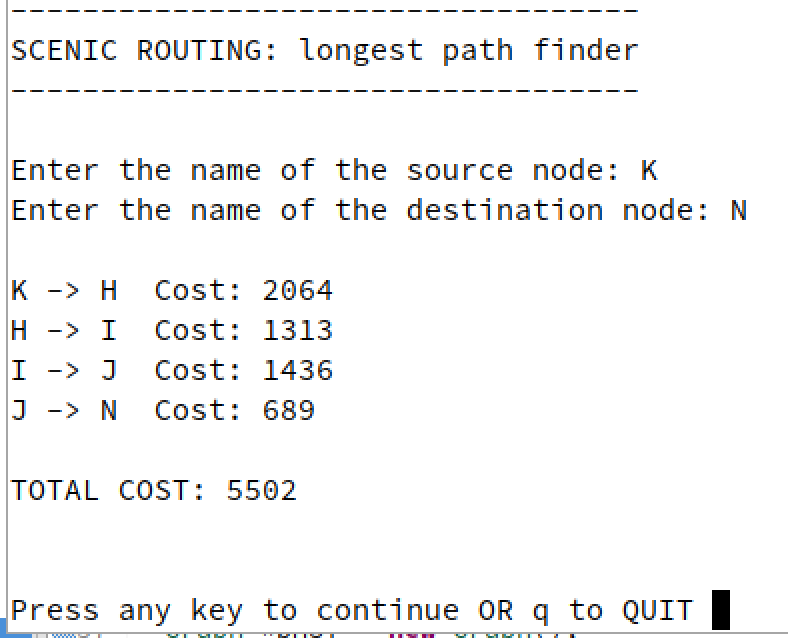
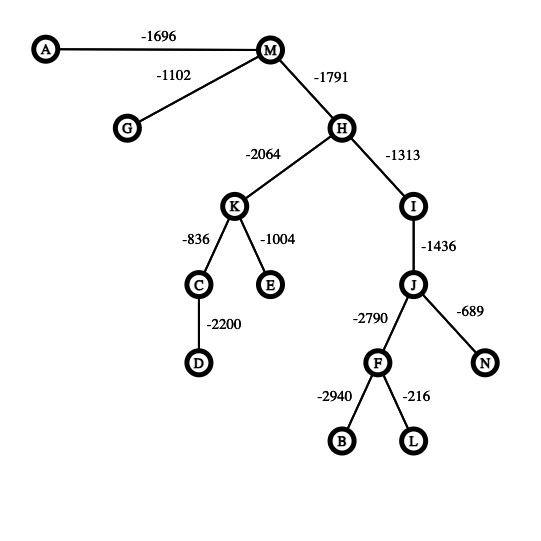
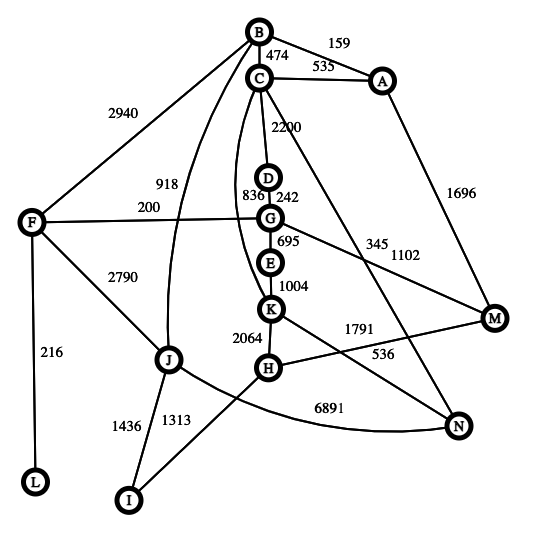
This is achieved by using the Graph Coloring method to find all the loops in the original graph then storing them to be accessed later if the nodes are adjacent to one another.

###### Implementation

The routing algorithm was implemented in C++. The program takes in a file with each node, edge, and edge weight needed to build the original graph. The algorithm is then applied and the user is able to enter a source and destination node. After printing out each edge in the determined path, the total cost of the path is displayed.

The following images are an example of the original graph (A), corresponding maximum spanning tree (B), and user input and program output (C) for a path with a non-adjacent source and destination.

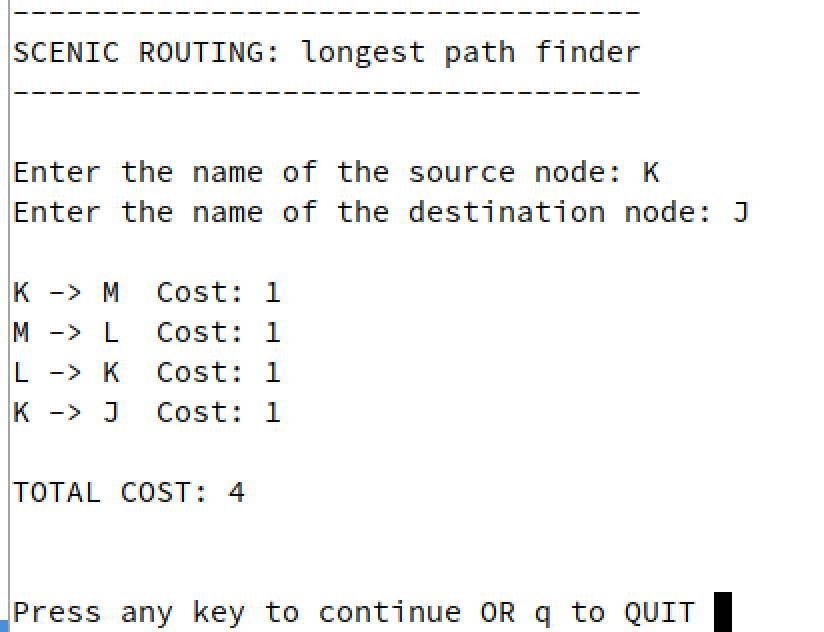
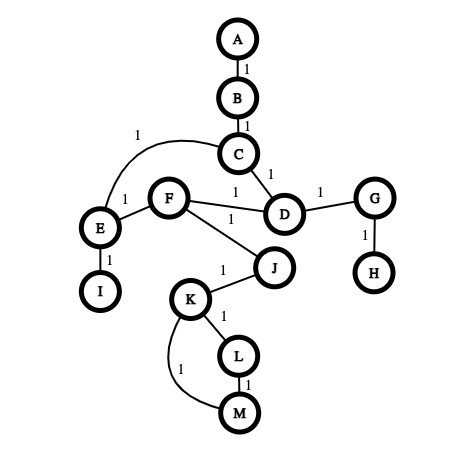
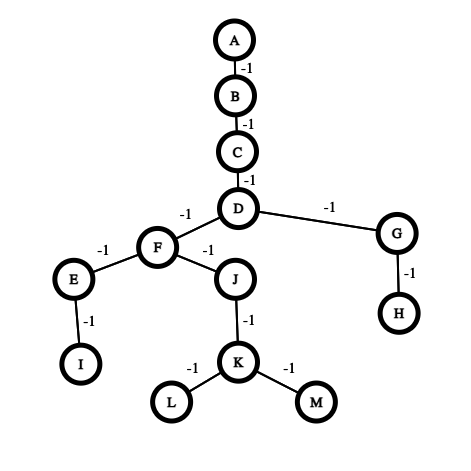
A B C



In the original graph, the nodes K and N are adjacent to each other, so the shortest path is a single edge with a total weight of 536. After the algorithm is applied to the original graph, the new path found is KHIJN for a total weight of 5502, which is much longer.

The next set of images is an example of the original graph (D), corresponding maximum spanning tree (E), and user input and program output (F) for a path with an adjacent source and destination that can have the loop added on.

D E F



In the original graph, the nodes K and J are adjacent to each other and remain adjacent to each other in the maximum spanning tree. So, the path in the original graph is a single edge with a total weight of 1. After the algorithm is applied to the original graph, the new path found includes the loop MLK, making the new path KMLKJ with a weight of 4.

##### Hello Protocol

###### What is a Hello Protocol?

The purpose of a Hello Protocol is to establish and maintain neighbor relationships. It also ensures that communication between neighbors is bidirectional.

Hello packets are sent periodically out all router interfaces. When a neighboring router receives a hello packet, it sends a new hello packet as an acknowledgement. Bidirectional communication is indicated when the router sees itself listed in the neighbor's Hello Packet.

[[content source](https://www.google.com/url?q=https://www.freesoft.org/CIE/RFC/1583/28.htm&sa=D&source=editors&ust=1618966137765000&usg=AOvVaw3oxnr6JUamczcQqCyoOkG5)]

For the purposes of this project, the network is a broadcast network, meaning that multicasting and UDP are used in the Hello Protocol.

###### Design

Scenic Routing implements a much simpler Hello Protocol than normally used in other routing protocols. This is because neighbor discovery is not being implemented, rather a file containing the nodes and edges in the network is read from on each router and is used to determine neighbors. The hello protocol is being used to check if the router’s neighbors are still up and able to communicate with.

The hello packet contains:

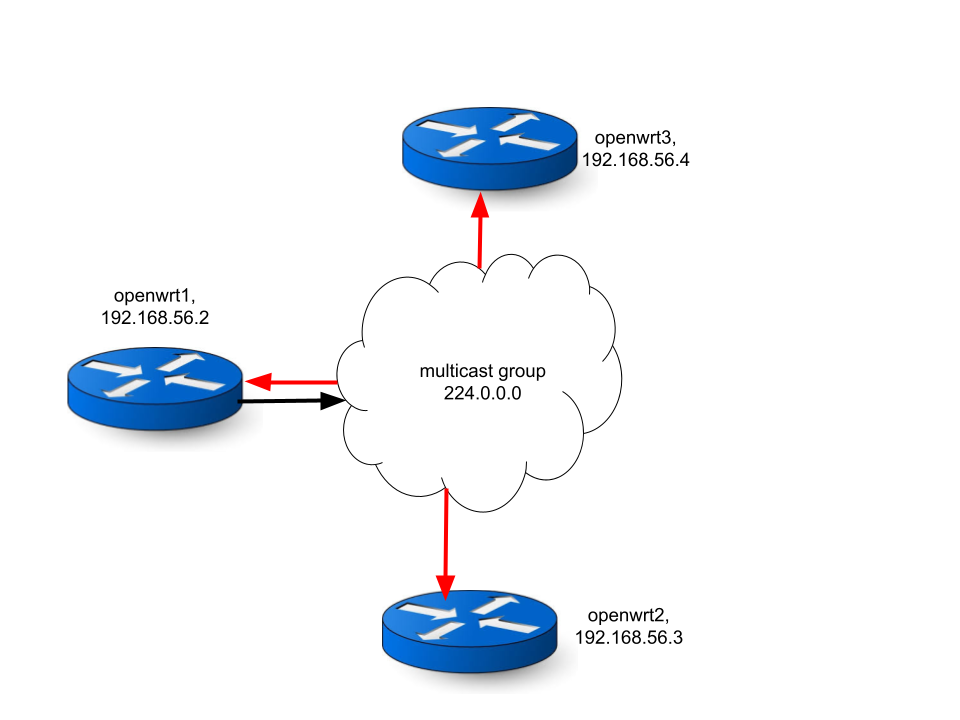
* The IP address of the router sending the hello
* The IP address of either the destination router or the multicast address.
* The time the packet was constructed
* A character to represent whether the packet was an acknowledgment or not.

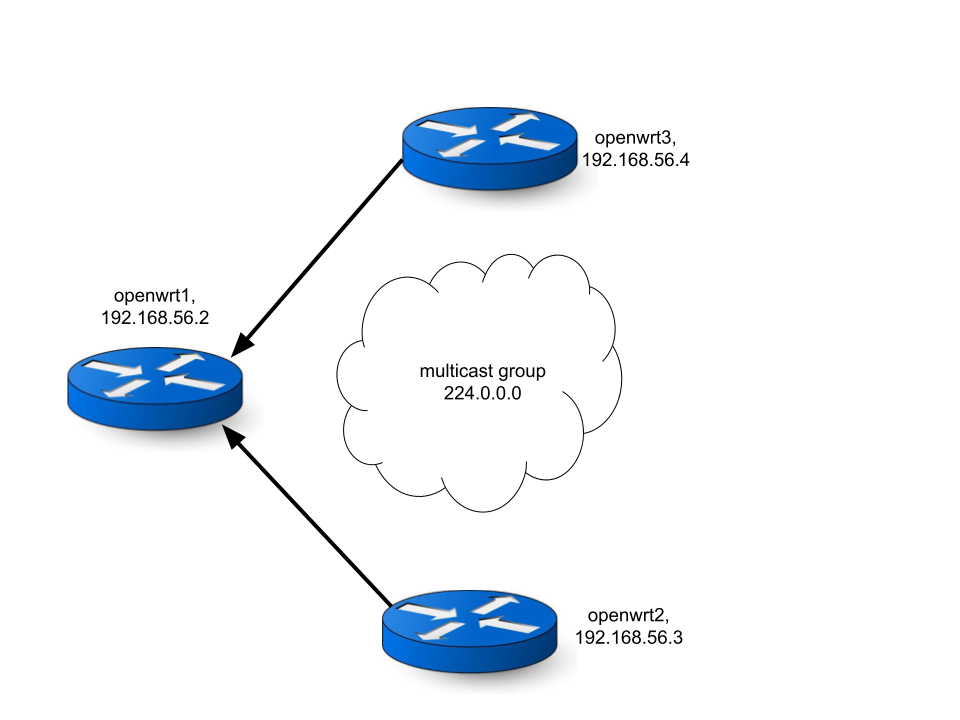
So, instead of the router waiting to see itself in another hello packet’s list of neighbors, the router keeps track of the time it last received an acknowledgement from one of its neighbors. If a certain amount of time has passed before receiving acknowledgement, the router is marked as down. Alternately, if a router suddenly receives an acknowledgment from a previously down router, it is marked as a neighbor again.

###### Implementation

The Hello Protocol was implemented in a combination of C and C++ and then cross-compiled to run on OpenWrt routers. The program itself utilizes the node and graph classes from the routing algorithm program to build the original list of neighbors. The program takes a single argument, its own IP address.

Threads are used to implement the periodic sending of hello packets to the multicast address, listening on the multicast address to receive multicast packets then send acknowledgements, and receive acknowledgement packets. This is to ensure that each of these tasks can happen concurrently and have access to the container that holds the previous time of acknowledgement for each of the router’s neighbors. Below is a set of diagrams depicting a router sending a hello packet to the multicast address and having the routers in the multicast receive that packet and the acknowledgements being sent back to that router.





The program currently sends and receives all hello packets, but the updating of the acknowledgements has yet to be implemented. In the future the program will also be able to find its own IP address without needing an argument.

#### Open Source Documentation

Here is the current list of topics covered by the documentation on GitHub:

* Routing Basics
* OpenWrt Router Build System Setup
* OpenWrt Cross Compilation Guide
* OpenWrt Router Setup Through VirtualBox
* OpenWrt Multicasting Setup
* Longest Path Algorithm Design
* Longest Path Algorithm Pseudocode
* Longest Path Algorithm Project Guide
* Hello Protocol Basics
* Hello Protocol Project Guide
* Socket Programming

#### Technologies Used

##### C/C++

The code for both the Routing Algorithm and Hello Protocol was written in a combination of C and C++ in the IDE Eclipse and Geany.

##### OpenWrt

OpenWrt (open wireless router) is an open-source project for embedded operating systems based on Linux, primarily used on embedded devices to route network traffic. The main components are Linux, util-linux, musl, and BusyBox.

In this project, OpenWrt is used as a router that can install user-defined firmware packages. This allows the user to write their own code and run it on a router to act as an interface.

##### OpenWrt Build system

The OpenWrt build system is a set of Makefiles and patches that automates the process of building a cross-compilation toolchain and then using it to build pieces of software to run OpenWrt on a specific device or software to run on an OpenWrt router. A typical toolchain consists of:

* a compiler, such as gcc
* binary utilities such as an assembler and a linker; for example binutils
* a C standard library, such as glibc, musl, uClibc or dietlibc

For this project, the OpenWrt Build System is used to cross-compile source code from a development machine or environment to an .ipk file (software) that can be installed and run on the OpenWrt Router.

##### Virtual Box

For this project VirtualBox was used to create virtual machines to run the x86-64 OpenWrt images and an OpenSUSE Linux development machine.