# Abstract:

*Semi-static routing algorithms attempt to combine many of the benefits of both Static and Dynamic routing algorithms to better suit the needs of the overall networking environment. The Semi-Static routing algorithm offers adaptive protocols to static environments to be able to respond to stresses within the network. The purposed implementation reveals the focus on reducing the number of packets dropped while also providing a more balanced overhead cost when compared to Dynamic routing algorithms.*

# Introduction:

In networking, routers are necessary to create paths so data can move from one point to another. Routers use various algorithms, each with its unique strengths and weaknesses, to accomplish this. These algorithms fall under 3 possible categories: static, dynamic, or semi-static. This project will involve the implementation of a semi-static algorithm in the form of a modified version of Dijkstra’s Shortest Path algorithm. The purpose of this paper is to explain and show the strengths of semi-static routing in comparison to static and dynamic routing. A major challenge for this project is updating the routing table in real-time and reducing network overhead costs.

# Static Routing:

Static routing (or non-adaptive routing) is a form of routing where the network administrator manually sets the routing table. The routing table doesn’t change unless the administrator changes it manually. Due to the manual nature of static routing, it doesn’t use complex algorithms and is simple to implement. Static routing is best used in smaller networks because the routing table must be set manually. Some benefits of static routing over dynamic routing are that it has higher security and requires less bandwidth. A major weakness of static routing is that a link failure can disrupt the routing table. The routing table can’t be updated automatically so if a link fails data can’t be re-routed until the administrator updates the routing table. One algorithm used for static routing is called flooding.

The flooding algorithm works by sending all incoming packets to all outgoing links except the one they arrived from. It always chooses the shortest path because it chooses every path. A consequence of flooding is that it creates many duplicate packets. There are numerous ways this can be dealt with; however, one way is to use a hop counter. A hop counter is a number put in the header of each packet and it is decremented each time the packet is sent to a router. If the packet is not at its destination when the hop counter reaches 0 then the packet is discarded. The hop counter should either be initialized to the shortest possible path from source to destination or the full diameter of the subnet. The second way to manage duplicates is sequence number-controlled flooding. A unique sequence number is put in the header of each packet when it is sent into the network. Each router keeps a list of sequence numbers and if a router receives a packet that it has already received and sent out then that packet is discarded. The third way to manage duplicates is to use selective flooding. This algorithm will send packets along paths that are heading in the right direction. For example, if a packet is heading eastbound it will be sent along links heading eastbound.

# Dynamic Routing:

Dynamic routing is a technique of routing that adjusts accordingly to real-time logical network layout changes. Dynamic routing uses complex routing algorithms that help it determine different routes it can take within the topology to get to the destination. This is significantly more efficient than static routing, especially when it comes to larger networks since a network administrator does not need to manually set the routing table. Since the network administrator is less involved and it automatically adjusts this makes dynamic routing less error-prone. It is significantly easier to make changes in the topology since it will automatically adjust, unlike static routing which requires a network administrator. Unfortunately, unlike static routing, dynamic routing requires the use of more resources and is also less secure. Two of the most popular algorithms that are used in dynamic routing are RIP (Routing Information Protocol) and OSPF (Open Shortest Path First). Routing Information Protocol was one of the first routing protocols implemented on TCP/IP. RIP is a dying routing protocol due to its scalability issues. Unfortunately, this routing protocol has a maximum hop count of 15 which means it does not optimize the best purpose of having a dynamic routing protocol that is using it for a large-scale topology. Therefore, this protocol works best for small-scale networks that plan on using dynamic routing. Instead, the better protocol which is just as popular as RIP is OSPF which is used in most large-scale networks. This protocol works side by side with the link-state algorithm to determine the best route. This protocol sends IP packets by using the link-state data to map out the topology. Dijkstra's algorithm is then used with the mapping of the topology to find the shortest path. Dijkstra’s Algorithm uses the edges that connect the different nodes (routers in this context) and their weight to establish the shortest path from the starting node to the ending node. Unlike Routing Information Protocol, Open Shortest Path First Protocol is much more complex and involves establishing a more significant number of parameters.

# Semi-Static Routing:

Although the definitions for both Static and Dynamic routing are more clearly defined, there is no clear definition for Semi-static routing due to resembling either Static or Dynamic at any given time with the overall network. In the related work by Alippi and Vanini in routing optimization in wireless sensor networks, they define semi-static for a network where “… nodes are … moving very slowly to reduce overhead [cost] associated with the routing tables…”. Although the work by Alippi and Vanini does represent their work within wireless sensor networks, we have chosen to define Semi-static routing as a semi-adaptive routing that is capable of self-adjusting to the networking conditions. Although there are many attempted implementations for Semi-static routing, these attempts focus on specific areas of application such as in wireless sensor networks or prioritizing their effectiveness within clusters of sensors to improve power efficiency using various protocol implementations.

Through the following sections, we will analyze the implementation and results gathered as part providing a concrete example of the definition of Semi-static routing defined above.

# Implementation Explanation:

The implementation of Semi-static routing required the use of Dijkstra’s routing algorithm as the main means of gaining the path from source to destination based upon the given graph. The full simulation program is implemented in four source files and three header files. The following will provide a high-level description of the function of the source code, which is linked in the Appendix [1].

* 1. The DijkstraRouting.cpp contains all functions and classes involved with running Dijkstra’s algorithm as well as storing all relevant information about the paths found from the source node. This code is also accessed for all update procedures during execution to facilitate the rerouting procedure based upon the updated graph.
  2. The SemiStaticRouting.cpp provides two functions to the running simulation that allow the non-static simulations to update the current graph with the given snapshot of the network (i.e., the initial graph and the queues as each of the connections within the graph). The second function searches the current path and returns the Boolean value of whether the node is found on the path or not.
  3. Next is the actual SimpleSimulation.cpp file that runs all of the disparate parts of the application using the given graph, node queue lengths, transmission rates as well as the source and destination nodes and the message size from the source node. The running simulation runs in sequential order in a single-threaded environment and indicates to the console the nodes that are being accessed and the queues at those nodes and other relevant data or information to the understanding of the simulation.
  4. Lastly, there is the main source code in FDCRoutingAlgorithms.cpp which is the entry point for the running simulation that passes the initial values necessary for the simulation object as well as returning the found simulation run times and the number of packets dropped by the current routing protocol’s simulated run.

Each of the above source code files provides a portion of the overall running simulation that allows us to understand the effectiveness of Semi-static Routing protocols more accurately.

# Results:

Using the code outlined above, our simulations were executed on the same system in a single-threaded environment to provide a direct comparison for a worst-case scenario for a Semi-static routing environment. The initial starting information of the graph — figure 1, queues at each node — table 1, and transmission rates for each node — table 2, were used for both the Static simulation and the Semi-static simulation to ensure there is no difference in setup for the given run environments.

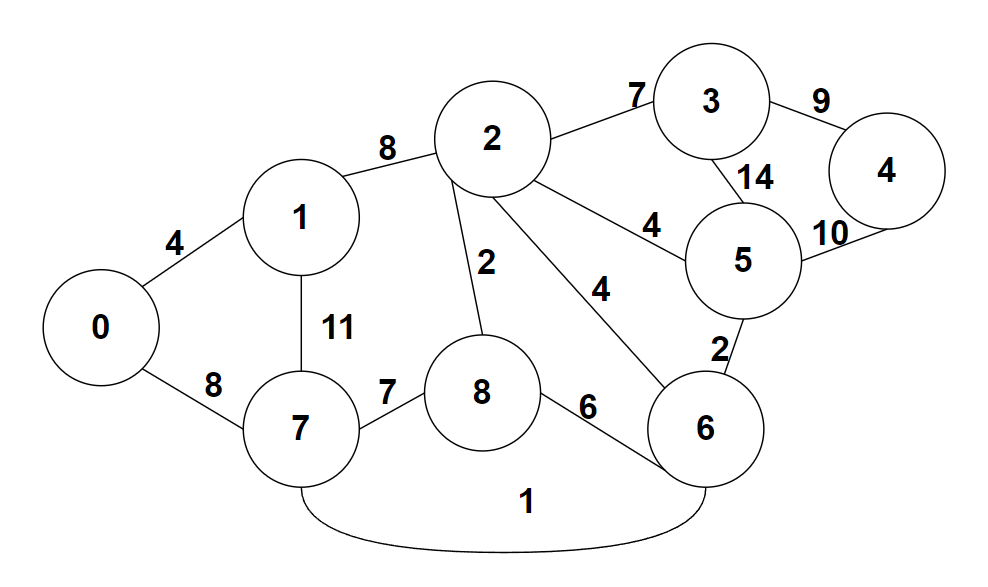


Figure 1: Initial Graph

The only distinction between the Static and Semi-static simulation runs is the Boolean value in the constructor which determines whether the current run is a static run or not. The program handles the rest by ensuring that certain features of the program are not accessed to ensure the run is as precise as possible for the given run conditions.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table 1 | | | | | | | | |
| N0 | N1 | N2 | N3 | N4 | N5 | N6 | N7 | N8 |
| 1000 | 100 | 50 | 70 | 45 | 80 | 1000 | 100 | 90 |

The queues themselves are a consistent value set for each connection that a node possesses (i.e., node 3 would have a queue of 70).

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table 2 | | | | | | | | |
| N0 | N1 | N2 | N3 | N4 | N5 | N6 | N7 | N8 |
| 10 | 8 | 12 | 9 | 7 | 10 | 6 | 7 | 13 |

The transmission values determine how fast information can be transferred between any two nodes. Much like the queues, each link from a particular node maintains the same transmission value (i.e., N7 has a transmission rate of 7 for all nodes it is connected to).

Text

Description automatically generated

Figure 2: Run Results

As seen by the above running results, the Semi-static routing simulation has a significant reduction in the number of packets dropped by the network due to an overallocation of the queue for a particular node. Although it does not assist in recovering dropped packets, it attempts to respond to the dropped packet by rerouting the path from source to destination to provide the best current path from source to destination. Even though the completion times from the Static and Semi-static routing simulation runs are drastically different, it is important to remember that the simulation is currently in a worst-case setup since the run environment does not support multithreaded processing.

# Conclusion:

Network routing is a critical part of transferring information from its source to its destination. Being that it is a critical part of data transfer, there are many times where the network needs more responsiveness than a Static routing algorithm provides while also reducing the amount of overhead cost associated with Dynamic routing algorithms. The purposed Semi-static routing algorithm attempts to showcase the capabilities of modifying the established Static or Dynamic routing algorithms to fit the needs of the network. Although there are many benefits to modifying the established routing algorithms, there needs to be more work accomplished to properly analyze the work of Semi-static routing algorithms of multi-source/multi-destination network trafficking as well as providing a multithreaded environment for execution. These expansions on the purposed solution as well as detailed comparisons between Semi-static and Dynamic routing algorithms provide a better dataset to accurately draw a conclusion of which algorithm is best for a given situation.

# Works Cited

*Static routing vs Dynamic Routing: Top 10 differences you should know*. EDUCBA. (2021, March 16). Retrieved December 1, 2021, from <https://www.educba.com/static-routing-vs-dynamic-routing/>

Alippi, Cesare and Giovanni Vanini. *Application-based routing optimization in static/semi-static Wireless Sensor Networks.*IEEE Xplore. (2006). <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1604792>

Deepali and Padmavati. *Improved Energy Efficiency semi-static routing algorithm using sink mobility for WSNs.*IEEE Xplore. (2014, March 8). <https://ieexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6799602>

*Advantages of dynamic routing over static routing: Advertise only the directly connected networks. updates the topology changes dynamically. administrative. - ppt download*. SlidePlayer. (n.d.). Retrieved December 2, 2021, from <https://slideplayer.com/slide/6923610/>

*What is dynamic routing?* Educative. (n.d.). Retrieved December 6, 2021, from <https://www.educative.io/edpresso/what-is-dynamic-routing>.

*Difference between static and dynamic routing*. GeeksforGeeks. (2020, June 15). Retrieved December 2, 2021, from <https://www.geeksforgeeks.org/difference-between-static-and-dynamic-routing/>

*Non-Adaptive Routing Algorithms. Non-adaptive routing algorithms. (2021, February 3). Retrieved December 6, 2021, from* [*https://www.tutorialspoint.com/non-adaptive-routing-algorithms*](https://www.tutorialspoint.com/non-adaptive-routing-algorithms)*.*

*FaadooEngineers. (n.d.). Flooding. Flooding, static routing algorithm, Computer Networks Lecture notes. Retrieved December 6, 2021, from* [*http://www.faadooengineers.com/online-study/post/cse/computer-networks/563/flooding*](http://www.faadooengineers.com/online-study/post/cse/computer-networks/563/flooding)*.*

# Appendix:

[1] Semi-static Routing Simulation: <https://github.com/JoelSmi/FDC-SemiStaticRouting>