**N-Shortest Paths: A Dynamic Routing Algorithm for Energy Conservation**

Kurtis Bertauche

CPE 400-1001

**Project Topic**: Dynamic Routing mechanism design with a focus on energy conservation

**Overview of Protocol:**

In a network, a router should maintain the n shortest unique paths to other routers in the network. When a path is needed, the router will randomly choose one of the paths that it knows for the packet. By randomly selecting one of n shortest paths, the router will spread traffic across the network instead of always sending packets down the same shortest path.

For example, in the figure below, suppose A would like to send packets to B. The shortest path would send the packets from A to C to D to B. However, there is another path if the packets are sent from A to E to F to G to B. If all the packets are sent along the shortest path, routers C and D will lose energy while routers along the secondary path do not lose any energy. By utilizing both paths, the energy loss will be more evenly spread across the network, preventing one node from losing all its energy before others.

Diagram

Description automatically generated

Fig. 1: Example of Similar Paths

**Specification of Protocol:**

In general, a router in the network will keep track of the n shortest unique paths to other routers in the network. When a path is needed, the router will randomly choose one of the paths that it knows for the packet. If a router does not know a path, it will broadcast the packet to all its immediate neighbors.

In addition to the data being transmitted, a packet will also keep track of its path, the position in the path, the packet id, and two flags: one for route request and one to indicate if the path is known. An overview of what should happen in various events is shown below:

**When a router has a new packet that it would like to send:**

1. The router already knows at least one path to the destination:
   1. Set the route request flag to FALSE
   2. Set the path known flag to TRUE
   3. Randomly choose a path from the known paths and add it to the packet. Forward the packet to the next router along the path
2. The router does not know any paths to the destination:
   1. Set the route request flag to TRUE
   2. Set the path known flag to FALSE
   3. Forward the packet to all immediate neighbors

**When a router receives a packet, and the router is not the destination:**

1. Route Request flag is TRUE, and the path known flag is TRUE:
   1. This is an invalid state. Return an error message to the source router of the packet so that it can be retransmitted.
2. Route Request flag is TRUE, and the path known flag is FALSE:
   1. Check if the router knows a path to the destination, if so:
      1. Use the current path information in the packet to learn as many paths as possible
      2. Change Route Request flag to FALSE
      3. Change path known flag to TRUE
      4. Combine the path the packet has already taken with the rest of the path to the destination, this way further routers along the path as well as the destination will be able to learn the entire path
      5. Forward the packet to the next router in the path
      6. Send a new packet back to the source router with the full path from the destination to the source, allowing the source router to learn the full path that the packet took
   2. The intermediate router does not know a path to the destination:
      1. Use the current path information in the packet to learn as many paths as possible
      2. Append current router information
      3. Forward to all immediate neighbors
3. Route Request flag is FALSE, and the path known flag is TRUE:
   1. Use the current path information in the packet to learn as many paths as possible
   2. Forward to the next router in the path
4. Route Request flag is FALSE, and the path known flag is FALSE:
   1. This is an invalid state. Return an error message to the source router of the packet so that it can be retransmitted

**When a router receives a packet, and the router is the destination:**

1. Route Request flag is TRUE, and the path known flag is TRUE:
   1. This is an invalid state. Return an error message to the source router of the packet so that it can be retransmitted
2. Route Request flag is TRUE, and the path known flag is FALSE:
   1. Create a new packet to send to the source router so that it may learn the path:
      1. Set Route Request flag to FALSE
      2. Set Path Known flag to TRUE
      3. Set the path in the new packet to the reverse path of the packet that just arrived
      4. Forward to the next router in the path
   2. Use the path information from the packet that arrived to learn as many paths as possible
   3. Send packet contents to the transport layer
3. Route Request flag is FALSE, and the path known flag is TRUE:
   1. Use the path information from the packet that arrived to learn as many paths as possible
   2. Send packet contents to the transport layer
4. Route Request flag is FALSE, and the path known flag is FALSE:
   1. This is an invalid state. Return an error message to the source router of the packet so that it can be retransmitted

**Additional Notes:**

Routers should keep track of the most recently seen packet ids. Then, if a packet receives a duplicate packet that it has already seen, it can use the path information to possibly learn new paths but should not forward the duplicate packet. This is necessary to prevent broadcasted packets from continuously multiplying.

Visual diagrams of the three events that a router could encounter are shown below:

Diagram

Description automatically generated

Fig. 2: Visual representation of protocol showing actions to be taken when a router has a new packet to send.

Timeline

Description automatically generated

Fig. 3: Visual representation of protocol showing actions to be taken when a router receives a packet, and it is not the destination router.

Timeline

Description automatically generated

Fig. 4: Visual representation of protocol showing actions to be taken when a router receives a packet, and it is the destination router

**Learning Paths:**

When learning paths, a router should dissect the path information that it receives to learn paths to as many other routers as possible. If the router has not learned n paths to that destination already and the path is unique, it should learn that path. If the router has already learned n paths to that destination, it should only learn that path if it is unique and shorter than one of the paths already known and evict the longer path in the process.

**Novel Contributions:**

Two novel concepts are introduced in this protocol, the ability for a router to remember more than one best path, and randomly choosing the path when more than one best path is known.

The parameter n controls the maximum number of best unique paths each router should remember. This parameter can be tuned for different network scenarios to provide optimal results for energy conservation. By allowing flexibility in the number of best paths each router will remember, the protocol could be useful in several different scenarios since it can be adapted to the network where it is being used.

Using random choices in choosing between shortest paths helps to ensure that packets get spread evenly across the network, as it is less likely that the same path will get continually chosen. While it would have also been possible to cycle through the known paths, this approach was not used due to the possibility of several nearby routers having similar paths chosen at the same time. A short burst of packets all using very similar paths could result in one or more nodes losing all energy before other nodes. By randomly choosing paths, the expected loss of energy is more evenly distributed across the network, making it less likely that one router runs out of energy much sooner than the others.

**Evaluation of Protocol:**

**Explanation of Code:**

To evaluate the effectiveness of the protocol, a simulation was created in Python to test the protocol in various networking scenarios. The simulation takes four inputs from the user:

1. A .csv file with a list of all routers and edges between the routers.
2. A .csv file with a list of packets. Each packet has a spawn time, source router, and destination router.
3. n – The number of shortest unique paths that each router should remember.
4. The default energy level of a router

The simulation reads in the network file and the packet file. The routers and their neighbors are stored while the packets are sorted based on spawn time. The simulation loop then begins. The simulation loop runs while there are: packets still in a buffer at any router **OR** packets that have not yet spawned **AND** the network is still alive. The network is considered alive if all the routers have an energy level of at least 1.

At every time point in the simulation, the following things happen: First, any packets that should spawn at that time are loaded into their source routers. The source routers add paths to the packets if known or prepare to broadcast them. Then, a list of routers that have at least one packet in their buffer is created. Every router in this list is then able to forward one packet, either to the next router in that packet’s path or by broadcasting it to all the router’s immediate neighbors if the path for the packet is not known.

If a router receives a packet, it processes the packet according to the protocol. Since time has not progressed at this point in the simulation loop, any received packets are added to the router’s buffer until the next cycle in the simulation loop. In addition to the information contained in the packet as specified, a few additional fields, such as hop count, arrival time, and creation time are maintained in each packet to be able to evaluate the protocol in this report.

After all the routers in the list have been allowed to process one packet, the network is scanned to ensure that it is still alive. The simulation loop continues until all packets have reached their destination or a router has run out of energy.

For this simulation, some assumptions have been made:

1. All packets take the same amount of energy to transmit (1 level of energy)
2. It takes the same amount of time to travel between every edge
3. The routers already know all their immediate neighbors
4. None of the routers or links fail during the simulation

Additionally, as the protocol specifies randomly choosing a path, the random seed has been set in the simulation so that results can be reproduced.

**Input Data for Evaluation:**

To test the protocol under different conditions, three different networks were created. A sparse network with fewer edges between nodes, a dense network with many edges between nodes, and a network with a moderate number of edges between nodes. Those networks are shown visually below in Figures 5, 6, and 7 and are stored in sparse\_network.csv, medium\_network.csv, and dense\_network.csv.

Diagram

Description automatically generated

Fig. 5: Sparse Network used in the simulation

Diagram

Description automatically generated

Fig. 6: Medium-density network used in the simulation

Diagram

Description automatically generated

Fig. 7: Dense network used in the simulation

An additional python script was created to handle the generation of packets to test in the network simulation. The additional script makePackets.py takes 4 arguments:

1. A network file in the same format that is used by the simulation script.
2. The requested number of packets to be generated.
3. The maximum spawn time of any of the packets.
4. The output file that the generated packets should be stored in.

This script will generate the requested number of packets by randomly choosing a source router, randomly choosing a destination router, and randomly choosing a spawn time within the given range. The script will ensure that the source and destination routers are not the same. The generated packets are written to a .csv file that can then be used in the simulation.

Three unique sets of packets were generated:

1. A “small” set of packets – 500 packets with a maximum spawn time of 250
2. A “medium” set of packets – 1000 packets with a maximum spawn time of 500
3. A “large” set of packets – 1500 packets with a maximum spawn time of 750

The random seed is not controlled in the makePackets.py script. Due to this, the packet sets that were used for the simulations for this report have also been included in the following files: smallPacketSet.csv, mediumPacketSet.csv and largePacketSet.csv

Each network was tested with each set of packets with the following values of n: 1, 2, 3, 4, and 5. Additionally, the simulation was always initialized with a default energy level of 1000 in every simulation that was performed.

**Performance Metrics:**

Delay – Delay is measured as any time that a packet is in a router and not immediately sent to the next router. For example, if router A receives a packet, but already has one other packet in its buffer, the packet would receive a delay of 1 as there is one packet ahead of it that must be transmitted first.

Delay is measured separately for packets that are loaded into the simulation and packets that are generated during the simulation as route responses. This allows the average delay for data packets to be calculated, the average delay for route response packets, and the overall average delay for all packets. A smaller value of delay is preferable.

Energy – The energy level of each router at the end of the simulation is recorded. As conserving energy is the goal of the protocol several different metrics are calculated from the remaining energy levels to obtain a thorough understanding of the performance of the protocol:

* Highest Remaining Energy – The remaining energy level of the router with the highest remaining energy level. A higher value is preferred, although does not necessarily represent the best performance in terms of conserving energy, as a small number of heavily used routers would not necessarily lower this value but would result in poorer energy conservation.
* Lowest Remaining Energy – The remaining energy level of the router with the lowest remaining energy level. A higher value is preferred, as a lower value indicates that one node is being favored more in the simulation and is likely to run out of energy sooner.
* Average Remaining Energy – The average remaining energy level across all the routers. A higher value is preferred, although does not necessarily represent the best performance in terms of conserving energy, as a small number of heavily used routers would not necessarily lower this value but would result in poorer energy conservation.
* Median Remaining Energy – The median remaining energy level across all the routers. The Median is also included in the case of an outlier affecting the average. A higher value is preferred, although does not necessarily represent the best performance in terms of conserving energy, as a small number of heavily used routers would not necessarily lower this value but would result in poorer energy conservation.
* Variance – The variance in energy levels across the routers. A lower value is preferred, as this indicates that the packets are more evenly distributed across the network.

As the goal of the protocol is to conserve energy across the network, more focus will be given to variance and the lowest remaining energy level. Focus is given to variance as this represents how well the packets were spread across the network. A higher value for a variance would indicate that some nodes are being preferred over others while a lower value for variance indicates a better job at spreading packets more evenly across the network. Focus is given to the lowest remaining energy level as this shows the remaining energy level of the router which transmitted the largest number of packets (i.e., the most preferred node). The higher the lowest remaining energy level is, the better job the protocol did at spreading packets across the network.

While average remaining energy, median remaining energy, and highest remaining energy are reported, they are weighted much less heavily in evaluating performance, as a small number of heavily used nodes would not have a large effect on these metrics but would have a poor performance in overall energy conservation across the network.

For each simulation, the same network file and packet file are used with varying values of n, the number of shortest unique paths each router should remember. This allows for investigation of the benefits or drawbacks of choosing to remember more than one path to destination routers in terms of both energy conservation and the effects of delay.

**Results:**

**Sparse Network:**

Summary of Delay

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Number of Packets | n | Number of Route Response Packets | Total Number of Packets | Average Delay of Data Packets | Average Delay of Route Response Packets | Overall  Average Delay |
| 500 | 1 | 88 | 588 | 8.37 | 17.56 | 9.74 |
| 500 | 2 | 86 | 586 | 6.15 | 17.37 | 7.85 |
| 500 | 3 | 86 | 586 | **5.90** | **17.33** | **7.61** |
| 500 | 4 | 86 | 586 | 6.15 | **17.33** | 7.83 |
| 500 | 5 | 86 | 586 | 6.00 | **17.33** | 7.70 |
| 1000 | 1 | 80 | 1080 | 9.14 | 17.38 | 9.75 |
| 1000 | 2 | 83 | 1083 | 18.92 | 16.99 | 18.77 |
| 1000 | 3 | 83 | 1083 | **6.85** | **16.93** | **7.62** |
| 1000 | 4 | 83 | 1083 | 9.63 | **16.93** | 10.19 |
| 1000 | 5 | 83 | 1083 | 9.23 | **16.93** | 9.82 |
| 1500 | 1 | 50 | 1550 | **1.82** | **5.58** | **1.95** |
| 1500 | 2 | 52 | 1552 | 2.88 | 5.60 | 2.97 |
| 1500 | 3 | 50 | 1550 | 2.01 | 5.70 | 2.13 |
| 1500 | 4 | 50 | 1550 | 2.01 | 5.70 | 2.13 |
| 1500 | 5 | 50 | 1550 | 2.01 | 5.70 | 2.13 |

Table 1: Delay of Sparse Network Simulation – For each packet size input, the best value is shown in bold

Summary of Energy Loss:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Number of Packets | n | Average Remaining Energy | Highest Remaining Energy | Lowest Remaining Energy | Median Remaining Energy | Variance |
| 500 | 1 | **886.43** | **929** | 744 | **895** | 1566.44 |
| 500 | 2 | 867.96 | 904 | **774** | 879 | 1187.50 |
| 500 | 3 | 869.83 | 914 | 772 | 874 | **1180.60** |
| 500 | 4 | 868.48 | 916 | 770 | 875 | 1314.44 |
| 500 | 5 | 869.74 | 911 | 770 | 876 | 1243.66 |
| 1000 | 1 | **796.17** | **878** | 515 | **810** | 6082.51 |
| 1000 | 2 | 756.22 | 834 | 458 | 769 | 6057.91 |
| 1000 | 3 | 766.09 | 861 | **531** | 778 | 5337.72 |
| 1000 | 4 | 755.17 | 841 | 511 | 763 | **4969.97** |
| 1000 | 5 | 756.65 | 850 | 513 | 762 | 5216.60 |
| 1500 | 1 | **719.61** | **859** | 384 | **748** | 12400.61 |
| 1500 | 2 | 659.65 | 807 | 351 | 699 | 13344.69 |
| 1500 | 3 | 678.35 | 836 | **435** | 716 | **12352.78** |
| 1500 | 4 | 677.52 | 831 | **435** | 715 | 12370.53 |
| 1500 | 5 | 678.17 | 836 | **435** | 715 | 12415.51 |

Table 2: Energy Loss in Sparse Network Simulation – For each packet size input, the best value is shown in bold

Chart, diagram, line chart

Description automatically generated Chart, line chart

Description automatically generated

Figures 8 & 9: Variance and Lowest Remaining Energy Levels plotted as the value of n differs in the sparse network simulation

In the sparse network, we find that the lowest remaining energy level is improved with n being greater than 1, and this is more pronounced with the larger data sets. Across all data sets, the variance is improved when n is greater than 1. The average delay is better when n is greater than 1 for the small and medium datasets, but not for the large dataset. This could indicate that in the long run, energy conservation is better when n is greater than 1 but at the expense of more delay.

**Medium Density Network:**

Summary of Delay

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Number of Packets | n | Number of Route Response Packets | Total Number of Packets | Average Delay of Data Packets | Average Delay of Route Response Packets | Overall  Average Delay |
| 500 | 1 | 87 | 587 | **4.84** | **11.02** | **5.76** |
| 500 | 2 | 87 | 587 | 5.23 | 11.62 | 6.18 |
| 500 | 3 | 84 | 584 | 5.93 | 11.76 | 6.77 |
| 500 | 4 | 84 | 584 | 6.65 | 11.79 | 7.40 |
| 500 | 5 | 84 | 584 | 6.50 | 11.80 | 7.27 |
| 1000 | 1 | 103 | 1103 | **3.34** | 11.04 | **4.06** |
| 1000 | 2 | 98 | 1098 | 4.23 | 10.40 | 4.79 |
| 1000 | 3 | 100 | 1100 | 3.50 | 10.16 | **4.06** |
| 1000 | 4 | 99 | 1099 | 3.95 | **10.13** | 4.51 |
| 1000 | 5 | 99 | 1099 | 3.83 | **10.13** | 4.40 |
| 1500 | 1 | 74 | 1574 | **1.63** | 3.38 | **1.71** |
| 1500 | 2 | 68 | 1568 | 2.13 | 3.68 | 2.19 |
| 1500 | 3 | 65 | 1565 | 2.03 | 3.42 | 2.09 |
| 1500 | 4 | 63 | 1563 | 1.93 | **3.14** | 1.98 |
| 1500 | 5 | 64 | 1564 | 2.08 | 3.36 | 2.13 |

Table 3: Delay of Medium Density Network Simulation – For each packet size input, the best value is shown in bold

Summary of Energy Loss:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Number of Packets | n | Average Remaining Energy | Highest Remaining Energy | Lowest Remaining Energy | Median Remaining Energy | Variance |
| 500 | 1 | **901.57** | 970 | **792** | **910** | **3208.17** |
| 500 | 2 | 886 | 970 | 772 | 888 | 3329.36 |
| 500 | 3 | 889.48 | 970 | 775 | 896 | 3388.35 |
| 500 | 4 | 886.26 | 970 | 772 | 892 | 3491.57 |
| 500 | 5 | 886.83 | 970 | 771 | 894 | 3533.33 |
| 1000 | 1 | **826.48** | 941 | **637** | **866** | 11129.53 |
| 1000 | 2 | 800.48 | 941 | 550 | 810 | 12688.53 |
| 1000 | 3 | 808.52 | 941 | 633 | 843 | **9905.62** |
| 1000 | 4 | 795 | 941 | 577 | 814 | 11371.82 |
| 1000 | 5 | 798.39 | 941 | 589 | 819 | 11313.98 |
| 1500 | 1 | **758.91** | 938 | 436 | **858** | 26752.54 |
| 1500 | 2 | 714.91 | 938 | 392 | 742 | **22864.63** |
| 1500 | 3 | 717.74 | 938 | 417 | 747 | 27729.47 |
| 1500 | 4 | 713.90 | 938 | **440** | 766 | 26963.57 |
| 1500 | 5 | 711.91 | 938 | 422 | 747 | 27296.08 |

Table 4: Energy Loss in Medium-Density Network Simulation – For each packet size input, the best value is shown in bold

Chart, line chart

Description automatically generated Chart, line chart

Description automatically generated

Figures 10 & 11: Variance and Lowest Remaining Energy Levels plotted as the value of n differs in the sparse network simulation

In the medium-density network, the lowest remaining energy level is not best with the value of n greater than 1 except for in the large data set. The variance is best for the small dataset when n is 1. For the medium and large datasets, variance is best when n is 3 and 2 respectively, but this result is not a trend, as the variance for other values of n that are greater than 1 is worse than the variance when n is 1 for these datasets. This suggests that it may be important to tune the value of n to a specific network, if possible, as an arbitrary value of n greater than 1 may not produce better results, but only a specific value of n. The delay for data packets is consistently worse when n is greater than 1.

**High-Density Network:**

Summary of Delay

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Number of Packets | n | Number of Route Response Packets | Total Number of Packets | Average Delay of Data Packets | Average Delay of Route Response Packets | Overall  Average Delay |
| 500 | 1 | 76 | 576 | **2.35** | 10.62 | **3.44** |
| 500 | 2 | 83 | 583 | 3.21 | **10.15** | 4.21 |
| 500 | 3 | 76 | 576 | 2.84 | 10.87 | 3.90 |
| 500 | 4 | 76 | 576 | 3.51 | 10.83 | 4.49 |
| 500 | 5 | 76 | 576 | 3.03 | 10.84 | 4.07 |
| 1000 | 1 | 94 | 1094 | **1.89** | **6.76** | **2.31** |
| 1000 | 2 | 90 | 1090 | 2.38 | 6.98 | 2.76 |
| 1000 | 3 | 92 | 1092 | 2.0 | 7.58 | 2.46 |
| 1000 | 4 | 92 | 1092 | 2.54 | 7.72 | 2.98 |
| 1000 | 5 | 92 | 1092 | 2.20 | 7.64 | 2.66 |
| 1500 | 1 | 81 | 1581 | **1.28** | **4.93** | **1.47** |
| 1500 | 2 | 76 | 1576 | 1.53 | 4.99 | 1.70 |
| 1500 | 3 | 84 | 1584 | 1.54 | 5.30 | 1.74 |
| 1500 | 4 | 81 | 1581 | 1.42 | 5.11 | 1.61 |
| 1500 | 5 | 81 | 1581 | 1.49 | 5.09 | 1.68 |

Table 5: Delay of High-Density Network Simulation – For each packet size input, the best value is shown in bold

Summary of Energy Loss:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Number of Packets | n | Average Remaining Energy | Highest Remaining Energy | Lowest Remaining Energy | Median Remaining Energy | Variance |
| 500 | 1 | 905.78 | **967** | **822** | 897 | **1481.36** |
| 500 | 2 | 899.61 | 966 | 793 | **902** | 2083.43 |
| 500 | 3 | 897.30 | 966 | **822** | 898 | 1483.13 |
| 500 | 4 | 891.26 | 966 | 792 | 888 | 1971.47 |
| 500 | 5 | 893.04 | 966 | 818 | 892 | 1593.23 |
| 1000 | 1 | 836.39 | **943** | 694 | **828** | 6113.25 |
| 1000 | 2 | 821.52 | 940 | 677 | 819 | 7040.44 |
| 1000 | 3 | 821.30 | 941 | 706 | 809 | **5438.68** |
| 1000 | 4 | 803.30 | 939 | 687 | 817 | 6580.49 |
| 1000 | 5 | 811.91 | 939 | **707** | 815 | 5569.08 |
| 1500 | 1 | 765.96 | **925** | 488 | 750 | 16897.68 |
| 1500 | 2 | 741.36 | 920 | 470 | **769** | 17187.25 |
| 1500 | 3 | 744.57 | 921 | 456 | 734 | 16727.26 |
| 1500 | 4 | 730.35 | 919 | **513** | 724 | **13840.60** |
| 1500 | 5 | 736.65 | 919 | 475 | 727 | 15522.96 |

Table 6: Energy Loss in High-Density Network Simulation – For each packet size input, the best value is shown in bold

Chart, line chart

Description automatically generated Chart, line chart

Description automatically generated

Figures 12 & 13: Variance and Lowest Remaining Energy Levels plotted as the value of n differs in the sparse network simulation

In the high-density network, we observe similar results to the medium-density network. The variance is improved when n is greater than 1, but not consistently. Similarly, the lowest remaining energy is improved when n is greater than 1, but not consistently, only for specific values of n, otherwise, a value of 1 for n produces the best results. The average delay is consistently worse when n is greater than 1.

**Discussion:**

While specific values of n greater than 1 were able to provide improvements in energy conservation in the medium and high-density networks, this was only achieved with very specific values of n, not any value of n greater than 1. While the small packet input did not see the same benefits in the medium and high-density networks, this could mean that the energy conservation benefits are only seen in the long run and that for smaller numbers of packets, no benefit is seen. This can be explained by the fact that for a smaller packet input, there will be fewer opportunities for routers to learn multiple unique paths.

The most consistent benefits of using a value of n greater than 1 were shown in the sparse network, whereas in most simulations results for energy conservation were consistently better when the value of n was greater than 1. This could be explained since, in a sparse network, the best paths for multiple sources and destination router combinations are more likely to overlap, whereas in a dense network it is less likely that the shortest paths between many source and destination router combinations will overlap, as there are more edges for paths to use.

Across the simulations, any benefit in energy conservation came at the cost of incurring an additional delay. Additionally, the extra time spent transmitting packets that use alternative slightly longer paths as opposed to the shortest paths was not tracked, but it is possible that the overall average time for packets to reach their destinations increased when multiple paths were used.

Another consideration that was not accounted for in this simulation is the extra resources necessary to make using more than one path possible. To use a value of n greater than 1, every router would need to have more memory to store the additional paths. Additionally, there would be an additional amount of nodal processing delay to randomly choose a path, this was not accounted for in the simulation. However, as the delay was already worse when n was greater than 1, the additional delay from nodal processing would not change the evaluation of delay in using the value of n set to 1 versus the value of n greater than 1.

**Conclusion**

The n shortest paths protocol has been evaluated in the simulation by using three different networks, a sparse network, a dense network, and a medium-density network. For each network, three packet-size inputs were tested, small, medium, and large, to allow testing over various lengths of time. For each input on each network, five values of n were evaluated, 1, 2, 3, 4, and 5, representing only remembering one shortest path up to remembering 5 shortest paths.

In several of the simulation scenarios, n-shortest paths produced favorable results as far as energy conservation is concerned, by producing a smaller variance and higher lowest remaining energy levels. However, these results were not consistent for every value of n greater than 1, as some values of n greater than 1 produced worse results than when n was set to 1. There was an increased delay consistently across the simulations when n was greater than 1. The best results were experienced in the sparse network.

The n-shortest paths protocol could provide benefits for energy conservation in specific network scenarios. Sparse networks are more likely to see benefits in energy conservation. Denser networks may also see improvements with n greater than 1, but it is more likely that the value of n would need to be specifically tuned to the network to provide any benefit.

*(Instructions on running the simulation are shown on the next page)*

**Running the Simulation:**

The following commands were used to produce the results shown in this report:

py makePackets.py sparse\_network.csv 500 250 smallPacketSet.csv

py makePackets.py sparse\_network.csv 1000 500 mediumPacketSet.csv

py makePackets.py sparse\_network.csv 1500 750 largePacketSet.csv

(All networks use the same 23 nodes, allowing the packet sets to be the same for each network)

py project.py sparse\_network.csv smallPacketSet.csv 1 1000

py project.py sparse\_network.csv smallPacketSet.csv 2 1000

py project.py sparse\_network.csv smallPacketSet.csv 3 1000

py project.py sparse\_network.csv smallPacketSet.csv 4 1000

py project.py sparse\_network.csv smallPacketSet.csv 5 1000

py project.py sparse\_network.csv mediumPacketSet.csv 1 1000

py project.py sparse\_network.csv mediumPacketSet.csv 2 1000

py project.py sparse\_network.csv mediumPacketSet.csv 3 1000

py project.py sparse\_network.csv mediumPacketSet.csv 4 1000

py project.py sparse\_network.csv mediumPacketSet.csv 5 1000

py project.py sparse\_network.csv largePacketSet.csv 1 1000

py project.py sparse\_network.csv largePacketSet.csv 2 1000

py project.py sparse\_network.csv largePacketSet.csv 3 1000

py project.py sparse\_network.csv largePacketSet.csv 4 1000

py project.py sparse\_network.csv largePacketSet.csv 5 1000

py project.py medium\_network.csv smallPacketSet.csv 1 1000

py project.py medium\_network.csv smallPacketSet.csv 2 1000

py project.py medium\_network.csv smallPacketSet.csv 3 1000

py project.py medium\_network.csv smallPacketSet.csv 4 1000

py project.py medium\_network.csv smallPacketSet.csv 5 1000

py project.py medium\_network.csv mediumPacketSet.csv 1 1000

py project.py medium\_network.csv mediumPacketSet.csv 2 1000

py project.py medium\_network.csv mediumPacketSet.csv 3 1000

py project.py medium\_network.csv mediumPacketSet.csv 4 1000

py project.py medium\_network.csv mediumPacketSet.csv 5 1000

py project.py medium\_network.csv largePacketSet.csv 1 1000

py project.py medium\_network.csv largePacketSet.csv 2 1000

py project.py medium\_network.csv largePacketSet.csv 3 1000

py project.py medium\_network.csv largePacketSet.csv 4 1000

py project.py medium\_network.csv largePacketSet.csv 5 1000

py project.py dense\_network.csv smallPacketSet.csv 1 1000

py project.py dense\_network.csv smallPacketSet.csv 2 1000

py project.py dense\_network.csv smallPacketSet.csv 3 1000

py project.py dense\_network.csv smallPacketSet.csv 4 1000

py project.py dense\_network.csv smallPacketSet.csv 5 1000

py project.py dense\_network.csv mediumPacketSet.csv 1 1000

py project.py dense\_network.csv mediumPacketSet.csv 2 1000

py project.py dense\_network.csv mediumPacketSet.csv 3 1000

py project.py dense\_network.csv mediumPacketSet.csv 4 1000

py project.py dense\_network.csv mediumPacketSet.csv 5 1000

py project.py dense\_network.csv largePacketSet.csv 1 1000

py project.py dense\_network.csv largePacketSet.csv 2 1000

py project.py dense\_network.csv largePacketSet.csv 3 1000

py project.py dense\_network.csv largePacketSet.csv 4 1000

py project.py dense\_network.csv largePacketSet.csv 5 1000

Finally, the line graphs shown in this document were produced using the analyzeResults.R file, which can be run with the R statistical software.