Improving Concurrency in a 3D Multihop Wireless Sensor Network Protocol

COMP 4203 - Dr. Christine Laurendeau

Gabriel Jensen
Alex Gao

Table of Contents

1 Introduction **1**

2 Background **1**

2.1 3DMA . . . 1

2.2 ONAMA . . . 2

3 Problem and Solution **3**

4 Analysis **6**

5 Conclusion **23**

6 List of Acronyms **24**

7 References **25**

1 Introduction

Wireless sensor networks are a continuously growing area in the field of networking and communications. They face many issues, including the need to acknowledge a third dimension, manage limited power sources, and utilize new routing protocols. This project examines a proposed three-dimensional routing protocol that addresses the void node and looping problems, 3DMA, in conjunction with a concurrent distributed scheduling protocol, ONAMA, in order to produce a concurrent variant of 3DMA [1], [2].

This paper is organized as follows. Section 2 details the 3DMA routing protocol and ONAMA scheduling protocol. Section 3 discusses implementing the routing and scheduling protocols together, how the simulation was modelled, issues encountered, and how results collection was performed. Section 4 analyses the results collected from our tests and compares them against the original papers' results. Finally, Section 5 wraps up the paper by providing a brief conclusion of the work as we discuss if 3DMA performed better with ONAMA, without ONAMA, or if no discernible change was detected.

2 Background

2.1 3DMA

[1] presents a three-dimensional (3D) space routing protocol for stationary multi-hop wireless networks. In a wireless network, when long range transmission is required and the nodes are outside of the transmission range of each other, an intermediate node (IM) is required in order to forward the data (hop) towards its destination [1, p. 1]. For practical use, real-world networks are 3D, such as sensor networks with ocean, pollution, or climate monitoring [1, p. 2]. The goal of this protocol is to overcome the Void Node Problem (VNP) and the looping problem [1, p. 2]. The void node problem is an issue associated with IMs, where the sending node attempting to forward packets to the destination is unable to find any further nodes to forward to [1, p. 2]. The looping problem occurs when the forwarded nodes begin to follow a loop and never reach the destination node [1, p. 2].

[1] proposes the routing protocols 3D Minimum Angle-Centralized (3DMA-CS) and 3D Minimum Angle-Distributed (3DMA-DS). 3DMA-CS finds the most efficient path from the base station to all user nodes in the network [1, p. 2]. In contrast, 3DMA-DS finds the most efficient path from all user nodes to the base station [1, p. 2]. It determines the path by first creating a line, called the reference line, from the sender to the destination [1, pp. 4–5]. The sending node then draws lines from itself to all of its neighbours and selects the node whose line forms the smallest angle with the reference line [1, pp. 4–5]. That neighbour node is selected as the next IM for the data to be forwarded to, the reference line is redrawn from the IM to the destination, and the process repeats until the destination is reached [1, p. 5]. To solve the VNP, if the current IM has no further nodes to move forward to, it backtracks to the previous node and selects the next node with the smallest angle from the reference line [1, p. 2]. To solve the looping problem, every node traversed is removed from possible nodes for future traversal in order to avoid going back to the same dead-end node [1, p. 2].

Both user nodes and the base station can use the routes generated by either 3DMA protocol [1, p. 2]. However, only user nodes can use the DS version and the base station can only use the CS version [1, p. 2]. [1] compared the performance of 3DMA-CS and 3DMA-DS to the Greedy Routing (GR) and Most Forward Routing (MFR) protocols [1, p. 11]. Both GR and MFR are algorithms that forward packets to nodes that are closest to its destination; MFR differs in that it tries to minimize the number of hops to the destination [3, p. 30].

The metrics tested for were end-to-end throughput, end-to-end delay, average path length, and average energy consumption while varying the transmission ranges and the number of nodes in the network [1, p. 11]. End-to-end throughput is defined in [1] as the measure of the amount of data that is successfully transmitted from a source to a destination in a particular time frame and is measured in Mbps [1, p. 11]. Both 3DMA-CS and 3DMA-DS outperform greedy and MFR protocols in throughput due to following the reference line. They allow for the shortest path and are optimised by the looping and VNP handling between nodes [1, pp. 12–13]. End-to-end delay is defined as the time taken by a packet being sent from a source to a destination and is measured in seconds [1, p. 11]. Similar to throughput, both 3DMA protocols have shorter delays than others due to finding the optimal shortest path between nodes [1, p. 11]. Average path length is defined as the average number of hops taken for every source to reach a single destination [1, p. 14]. Their results found that both the DS and CS versions of 3DMA required slightly more hops on average than the greedy and MFR algorithm, due to the measures taken for loops and void nodes [1, pp. 14–15]. Finally, average energy consumption is defined as the amount of joules used as packets are sent from source to destination [1, p. 13]. On average, both 3DMA protocols used less energy than greedy and MFR protocols [1, pp. 13–14].

2.2 ONAMA

[2] proposes a distributed scheduling protocol for stationary wireless sensor networks that improves a low latency network's concurrency. The scheduling protocol is called the Optimized Node Activation Multiple Access protocol (ONAMA) and is based off of the Node Activation Multiple Access (NAMA) protocol. ONAMA assumes all nodes are synchronized and divided into time slots [2, p. 2].

NAMA is a scheduling protocol that selects nodes for channel access based on a priority assigned to them, where the winning node must have higher priority than its neighbours [2, p. 1]. Priorities are determined by hashing a node's unique ID and the current time slot using a fast digest message generator that results in a random integer; the node's ID is then appended to that integer. [2, p. 1]. While NAMA prevents nodes from transmitting if they were to interfere with each other, it results in the channel being underused and not optimally concurrent [2, p. 1]. NAMA uses a conflict graph in order to ensure there are no collisions in transmission [2, p. 1]. A node in a conflict graph represents the transmission link between two nodes in the original network [4, p. 474]. An adjacent link between two nodes in a conflict graph means that there is a common node, in the original network, between the two transmission links [4, p. 474]. For the rest of this paper, nodes in a conflict graph will be referred to as "conflict nodes".

In ONAMA's base version, known as the Distributed Maximal Independent Set (DMIS) algorithm, all possible Maximal Independent Sets (MIS) in the conflict graph are found in order to maximize the number of concurrent transmissions in the network [2, p. 2]. An independent set is a set of conflict nodes that are non-adjacent to each other, and it is maximal if adding any other conflict node would no longer render it an independent set [2, pp. 1–2]. Conflict nodes are in a state of either Active, Inactive, or Undecided, which is determined based on both their priority and the state of their neighbouring conflict nodes [2, p. 2]. The DMIS algorithm is run by each conflict node in parallel and runs a loop that completes once the node is not Undecided [2, Algorithm 1, 4]. In the loop, if the current conflict node's priority is greater than the priority of all Active and Undecided neighbours, then the current conflict node's state is set to Active [2, Algorithm 1, 4]. If any neighbouring conflict node has a higher priority and is Active, then the current conflict node is set to Inactive [2, Algorithm 1, 4]. Active means it will join the MIS and be able to transmit [2, p. 2]. Inactive means it will need to wait for future time slots in order to send [2, p. 2]. Through these MISs, we can see that all the conflict nodes that belong to those sets would be able to transmit without interfering with its neighbours, allowing for more nodes to communicate at a time. The NAMA protocol does not require data exchange between neighbouring conflict nodes, making it convenient to compute node priority quickly [2, p. 3]. However, one of DMIS's phases requires transmission of data to determine the state of neighbours, so the pipelined precomputation variant of ONAMA is introduced to reduce delay [2, p. 3]. The pipeline precomputation variant computes the MISs for future slots by having the conflict nodes complete their state exchange in advance by using the priorities of the future slots [2, p. 3].

[2] measured results based on mean concurrency and Packet Delivery Ratio (PDR). Mean concurrency is measured with packets per slot and a more concurrent system would have more packets per slot [2, p. 5]. PDR refers to the ratio of packets received successfully over the total number of packets sent [2, p. 5]. They compared ONAMA with NAMA and a publicly used scheduling protocol iOrder [2, p. 5]. Results found that ONAMA outperformed NAMA by a wide margin and obtained comparable results to iOrder [2, pp. 5–7].

3 Problem and Solution

[1]'s 3DMA routing protocol is currently only capable of having one node (amongst a random distribution of nodes in 3D space) transmit messages at a time in order to avoid interference [1, p. 15]. [2] proposed the ONAMA protocol as a scheduling protocol to allow for concurrent communication; we will attempt to implement ONAMA in order to introduce the concurrent aspect into the existing 3DMA algorithm.

[1] conducted their simulation using MATLAB. Our simulation code is adapted from an existing wireless sensor network simulator based on the SimPy Python framework, WsnSimPy [5]. Using an existing wireless sensor network simulator that includes topology visualization will allow one to easily see the paths determined by 3DMA. The network is represented by one base station, and many user nodes; [1] simulated 100-500 nodes over a 50x60x50 km area, where each node was capable of a 2.5km transmission range. Within wsnsimpy, position and size is represented by pixel size. Therefore, our simulation space is 500x600x500 pixels. Nodes have ranges from 20-50 pixels and use the same amount of 100-500 nodes. The nodes run 3DMA-DS to find the best path to the base station. We did not implement 3DMA-CS as we saw no purpose for the base station to find routes to every node in the network. We wish to replicate the results obtained by [1] as closely as possible, so the simulation parameters are largely kept the same, if possible. Our approach to achieving concurrency is to have the nodes calculate their routes upon network initialization. Since the nodes are stationary and always running, the routes only need to be calculated once. Whenever a node wants to send data to the base node, it will enter the conflict graph, determine its priority and state and send its data depending on if it is added to the MIS.

On initialization, nodes determine routing paths to the base station using 3DMA-DS. As the network is started, each node will have a 20% probability of sending a message each second; [1] specifies this in their simulation parameters as a message of 1MB in size. Nodes will have known their neighbours at this point, if they were in the transmission range of each other and the Euclidean distance between them did not exceed that range [1, p. 4]. The network has an ONAMA scheduler which is responsible for creating the MIS and informing the nodes of when they are allowed to send their data to the base node. When a node has data it would like to send, the ONAMA scheduler adds its conflict node to the conflict graph and decides if it is allowed to transmit [2, p. 2]. Before determining the MIS, the ONAMA scheduler waits for the conflict nodes to resolve their states so no conflict node is left as Undecided [2, Algorithm 1, 4]. Upon transmission, a node will follow the 3DMA routing path it previously determined at the start. After all nodes in a MIS complete transmission, their corresponding conflict node is removed from the MIS and the conflict graph. They then repeat the process of attempting to send data every second with a 20% probability of success.

Should there be a void node, where no neighbours are detected and the destination is not reached, the IM will be backtracked and the next possible minimum angle neighbour is selected [1, p. 7]. Concerning looping, if a node was selected for an IM previously and was successful in forwarding data, it is excluded from the routing path (parent nodes to backtrack to if needed) [1, pp. 5–7]. This would allow an optimal amount of multihop wireless nodes to communicate without interfering with each others' transmissions.

The simulation code starts by generating the nodes at possible positions. This is done by adding the base station node at (40, 40, 40) coordinates, then for the specified number of nodes given as a parameter, every subsequent node is generated with respect to the previous node. To do this, the previous node's (in this case, the base node) coordinates are used as a reference to determine the position of the next sensor node. The new position is generated with random x, y, and z coordinates that are within transmission range of the previous node. These values are also checked to see if they are within the borders of the simulation area. Through this method of generation, this guarantees that every node will be within transmission range of at least one other node. Next, all nodes initialize and calculate their 3DMA routing path to the base station node. The simulation is then started, where all sensor nodes will have a 20% chance of wanting to send data. If it is going to send, it will use up some energy and send it along the its 3DMA routing path. Only one node is allowed to send at a time, as constrained by [1] to simulate the avoidance of interference. Any intermediate nodes that forward the message on will also expend energy in doing so.

3DMA with ONAMA creates an *onama_scheduler* object which is initialized with information about all the nodes. All nodes are assigned a corresponding *ContentionEntity* object which represent the node's conflict node. A *ContentionEntity* object uses the ID and path of its associated node to represent the node's path in the conflict graph. If a node wishes to send to the base station, its *ContentionEntity* is added to the conflict graph. Each entity runs a variant of the DMIS algorithm described in [2]. The original algorithm runs a while loop that does not complete until every node's state is no longer Undecided. This was done with the assumption that the nodes would be running in parallel and that the states of other nodes would change as another node's loop was running. Since our simulation is single threaded, we removed the loop from the algorithm as it would run into an infinite loop problem. Our alternative algorithm was to have the ONAMA scheduler run a loop that only breaks when all entities are no longer in a state of Undecided. In the loop they would call the DMIS algorithm for each entity, individually, until all of their statuses were assigned. The ONAMA scheduler is responsible for generating the MIS by only adding entities that are in an Active state, and these entities indicate their unique priority. This unique priority was described as a hash of the node's ID concatenated with the time, and that resultant hash is concatenated with the node ID again for uniqueness. Furthermore to ensure randomness, the resulting hash was used as the seed for a random function and the next random number generated was used as the priority. Entities with a higher priority to their neighbours become Active. Otherwise, the entity becomes Inactive, as no entity is left in an undecided state. After the Active entity allows its nodes to send their data, both the conflict graph and MIS are reset, and the priorities are computed once again upon the next time slot of the scheduler. Each entity uses a global time variable that is only updated by the scheduler. This is to replicate a parallel and synchronized environment so all the entities are using the same time. With ONAMA, multiple nodes are able to send simultaneously, without interference, as they are non-adjacent to each other.

Another modification required for our simulation was the removal of time slots when implementing ONAMA. ONAMA as described in [2] is used for slotting a TDMA-based system. Due to the varying lengths of time required for sensor nodes to send to the base station, we were unable to establish a TDMA-based scheduler. Our alternative was having the next set of nodes send as soon as the last transmission from the previous MIS was received by the base station.

We have a list of assumptions we made during the development of the simulation. All nodes are sensor nodes, except for the base node. All nodes have GPS abilities and know their own coordinates, the coordinates of their neighbours, and the coordinates of the base node. Since each node knows the position of every other node, they should all be capable of calculating their paths to the base station offline. All nodes are stationary and will not enter or leave the network during simulation. Every node will be within transmission range of at least one other node, and nodes can only transmit to each other if the Euclidean distance between each other is less than their transmission ranges. Each node has a unique ID for ONAMA. The base node can receive multiple messages at once, but nodes can only send one packet of data at a time. A packet will be able to reach its destination before the next time slot begins. Finally, we are not concerned about the channel bit error rate, so the loss of packets/data and occurrences of errors are not included in our simulations (as we are more concerned with concurrency).

Our simulations were set up to include user input so the user could specify the number of nodes to be used for the simulation (ranging from 100 to 500), as well as their transmission ranges (ranging from 25 to 500). The nodes are randomly distributed in 3D space although wsnsimpy only displays the network in two dimensions.

In attempting to recreate our own simulations, it was apparent that although [1] may have been "peer-reviewed", there is an overall lack of description and details in certain areas that would have benefited from it. First, in the *vnp_handling* algorithm described, the pseudocode given is insufficiently explained and actually contains a branch that is impossible to reach [1, Algorithm 6, 7]. It does not read like a sufficient solution to the void node problem, so several changes were required. A list to track all nodes visited was added to solve the looping problem, the usage of the flag was omitted, and the entire branching statement on Line 22 of Algorithm 6 was not implemented [1, Algorithm 6, 7]. Second, the described method to calculate throughput did not fully explain its reasoning as to why it chose certain values. Values were chosen for our simulations that are believed to replicate [1]'s results to the best of our abilities. Third, the energy usage is given in the network parameters, but it is never explained how any node uses these energy values. This is also hindered by the fact that how energy consumption was measured was not described. Fourth, the propagation time is not specified, so the values cannot be replicated and can only be relatively compared. Fifth, when [1] varied the transmission range in their testing, the number of nodes used for their trials was not clear. Finally, while it was stated that their network simulation was run 100 times, how many messages were sent and how long each of those individual trials took were not mentioned. In general, there is a distinct lack of description as to how most of the metrics were collected, making replication of results difficult.

In order to prepare the simulation but also adhere to the assumption that every node will be within transmission range of at least one other node, it is likely that the method used to generate nodes affected the metrics. In order to ensure that no node was placed without a neighbour, we created each node sequentially where a new node was created within transmission range of the previously created node. This often leads to bottlenecks which can severely reduce concurrency with ONAMA scheduling. These bottlenecks occur when a sequence of nodes are generated following a certain direction which forces the path of every node generated afterwards to go through that sequence of nodes. The most common instance of this was when the base station node would have only one neighbour node. With a single neighbour, only one node in the whole network would be able to send at a time and the whole point of concurrency is useless. A solution to this would have been to devise a node generation algorithm that incorporated a random uniform distribution function that took into account the need of having every node be within range of at least one other node. In attempting to match the data of [1] but with an unknown run time of the network trials, it seemed that with a greater number of nodes (and default node transmission range of 25), there was less data reaching the base node in 20 seconds. Reaching the base node is a required condition to record the total delay accumulated from each time the message was forwarded, so such trials may have skewed data indicating a low end-to-end delay.

When [2] were discussing the hash of a node, they mentioned that they used a “fast message digest generator that returns a random integer”. This is vague and has no indication to exactly which generator they used. Our initial solution to this was using an MD5 hash function as our fast message digest generator. However, this ran into the issue of nodes with larger ID values consistently having higher priority over their neighbours, thus being the only nodes sending any information. Our solution to this was using the key generation function *pbkdf2_hmac* from the hashlib library in Python. This function took in a random salt value to help generate some randomness. The result from the function was then used as the seed for Python’s random function and the next random number generated was used as the node’s hash. This hashing procedure led to priorities that seemed random no matter what ID was used.

Another issue with [2] was there was no explanation on how the nodes knew what time slot they were allowed to send. We still had this problem even though we did not use a TDMA system. Our solution to this problem was by having the *onama_scheduler* object be able to manage all *ContentionEntity* objects instantly with no delay and full privilege.

In terms of collecting results, arbitrary values are assigned for energy levels and delay of hops for energy consumption and end-to-end delay metrtics. Since it was unclear in how energy was used in [1], each time a node sends or forwards data, it will expend 400,000,000 nanojoules ($\frac{50 \text{ nanojoules}}{1 \text{ bit}} * 8,000,000$ bits, or 1 MB of data). The propagation time from the original source repository contained a propagation time value, defined as the distance from the current node to the destination node, divided by 1,000,000 [5]. End-to-end delay here is thus measured as the summation of these propagation times between intermediate nodes as a message travels to the base node. End-to-end throughput was described in a calculation that occurred after a path was generated; a specific code and bit rate are applied depending on the length of hops in the path and the size of the packet over the time it took to be transmitted is calculated [1, Algorithm 1, 5]. Average path length was calculated after 3DMA routing paths were calculated, where the it was the sum of all path lengths divided by the number of paths. Mean concurrency was observed as the number of concurrently transmitted packets per given time frame.

To collect the metrics, the 3DMA and 3DMA with ONAMA simulations were run with variations made to only two parameters per test set, the node transmission range and the network node count parameters. Each test set was comprised of running 100 simulations with 20 seconds per simulation. When testing different values for one parameter, the other parameter was kept constant. In testing varying transmission ranges, the default node count was always 250. Varying node counts had a default transmission range of 25. The first tests varied the transmission ranges, and test sets were completed with values from 20 to 50, inclusively, with 5 increments in between. For varying node counts, test sets were completed with values from 100 to 500 with 50 increments. These test values were chosen to match the ones performed in [1]. For each 3DMA test, end-to-end throughput, end-to-end delay, average path length, and energy consumption were collected into a file. The 3DMA with ONAMA tests collected the same results but also included the mean concurrency. For each set of 100 results, the average values were used in creating the graphs that follow in the *Analysis*.

4 Analysis

Due to using a different framework from both [1] and [2] we were unable to replicate the environment conditions of either paper. Also due to the vague information and lack of detail on how the metrics were collected in [1], much of the comparisons in the collected data are relative to the original. Recall that the only implementation of 3DMA in this project was the distributed version, 3DMA-DS.

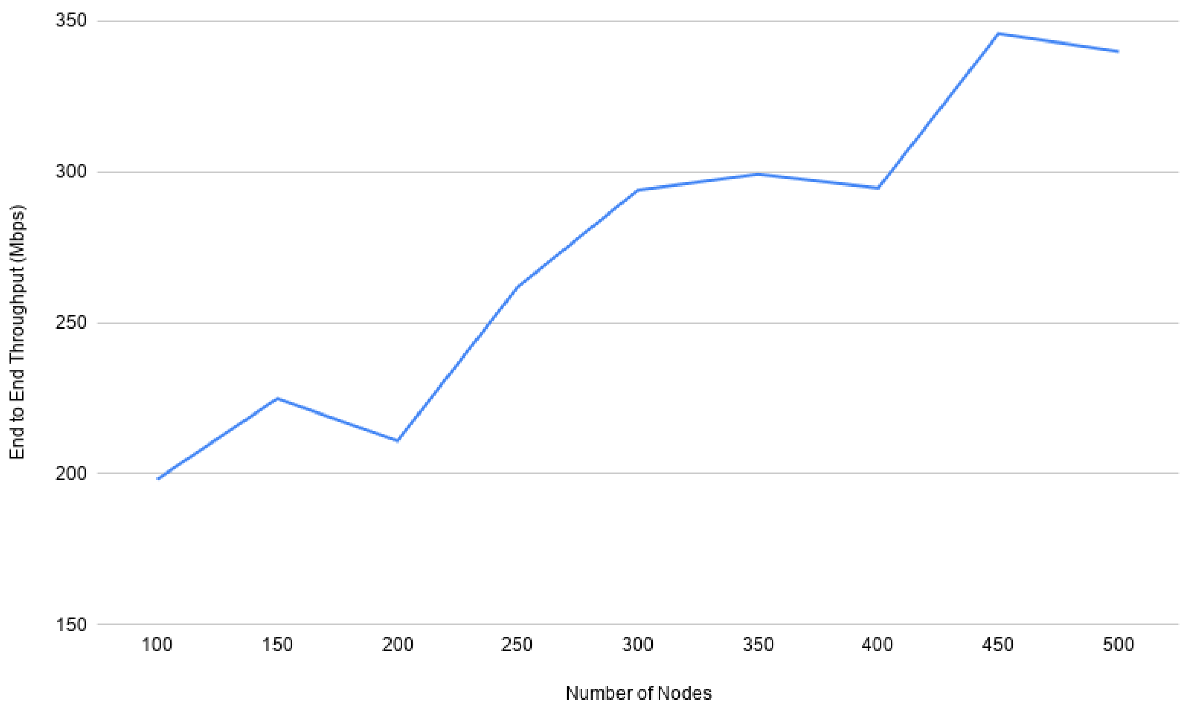

Figure 1: Average end-to-end network throughput of 3DMA implementation, with node transmission range of 25

As the number of nodes increases, so does the end-to-end throughput. There appears to be some sort of discrepancy between how [1]'s throughput was generated compared to ours, but the cause is unknown. Regardless, the data can be relatively compared with the original, to which it somewhat resembles.

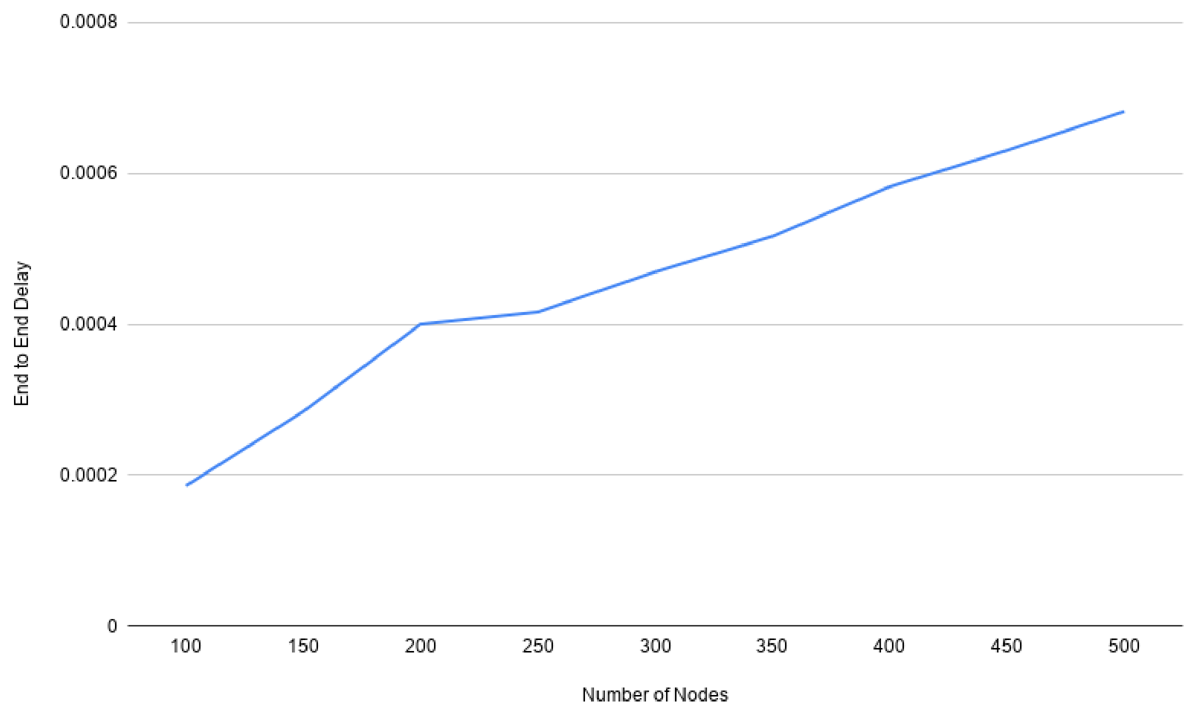

Figure 2: Average end-to-end network delay of 3DMA implementation, with node transmission range of 25

As the number of nodes increase, the end-to-end delay also increases. This does not resemble the original data, and this may be due to the method of collection of end-to-end delay times, as well as our method of generating nodes which probably led to a bias due to bottlenecks.
Note: The data at 500 nodes did not record any delay values for 4 entries (because no message made it to the base node within the 20 second trial length), so the average of values is only for 96 trials.

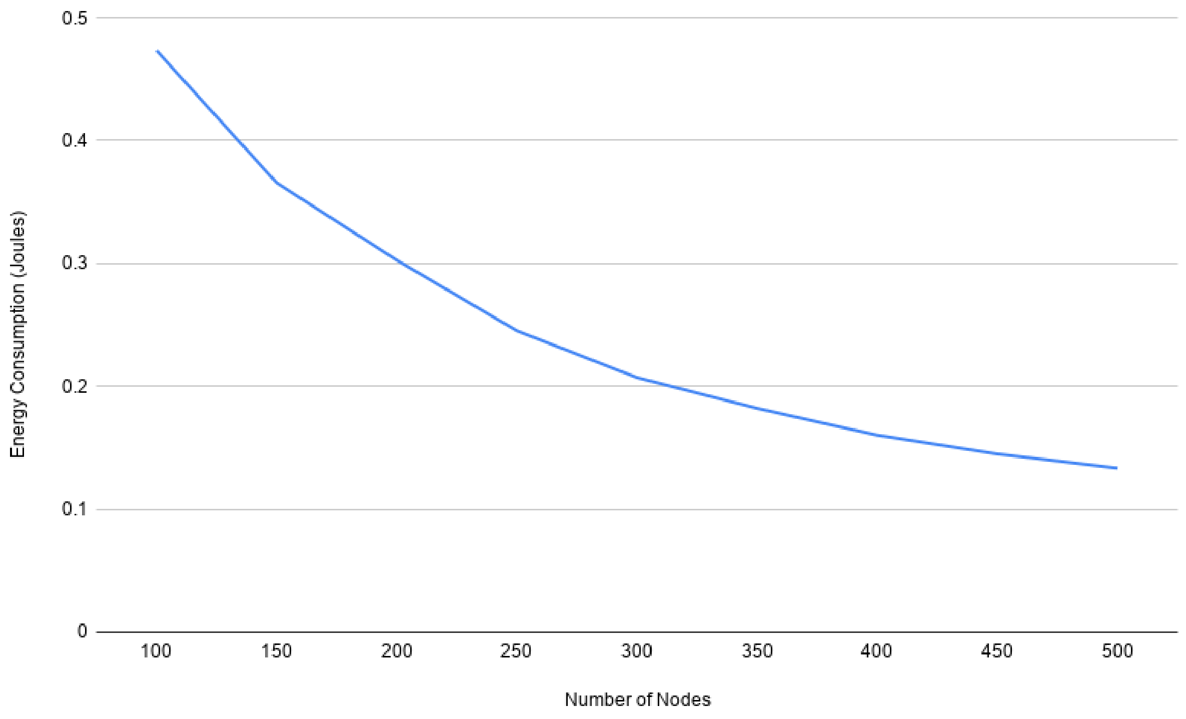

Figure 3: Average network energy consumption of 3DMA implementation, with node transmission range of 25

Due to the lack of detail on how the energy consumption was originally collected, the scale of the data collected does not match. In both cases, the energy consumption decreases as the number of nodes increase. This trend may be explained with the reasons given in [1, p. 13].

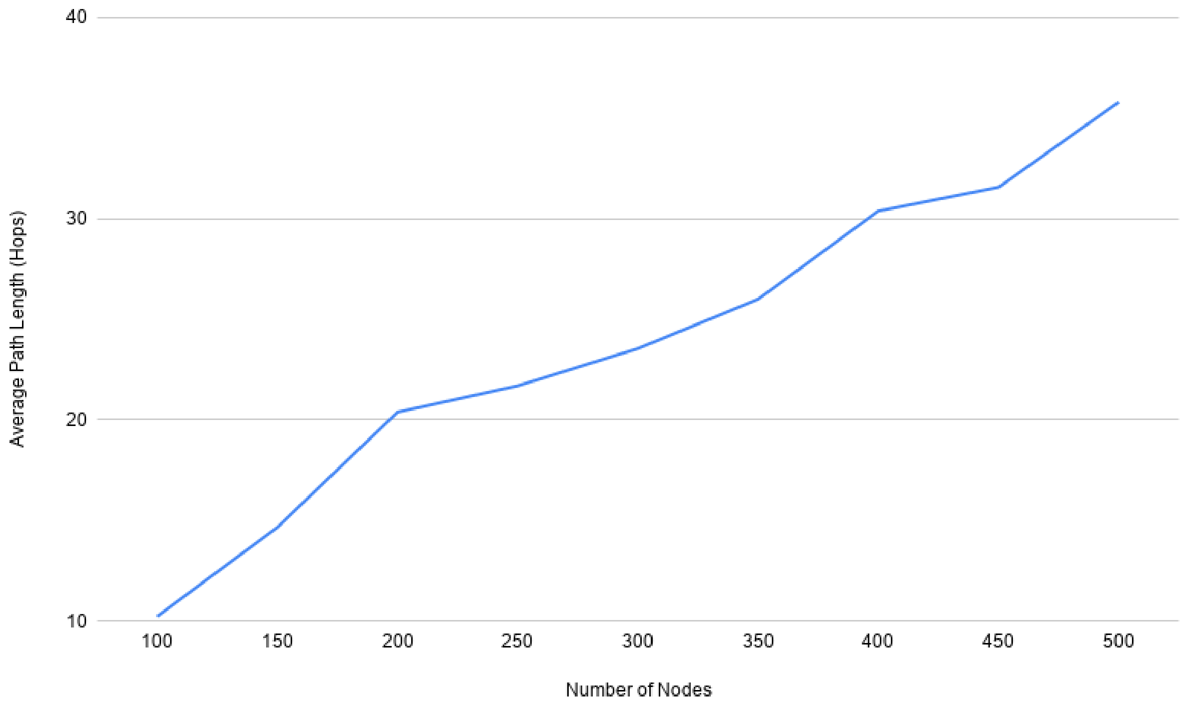

Figure 4: Average path length of 3DMA implementation, with node transmission range of 25

Here, the trend of our simulation data does not match the original trend. Our observed average path length tended to increase as the number of nodes increased, and the most likely cause for this may be due to our method of node generation which was not uniformly distributed and led to bottlenecks.

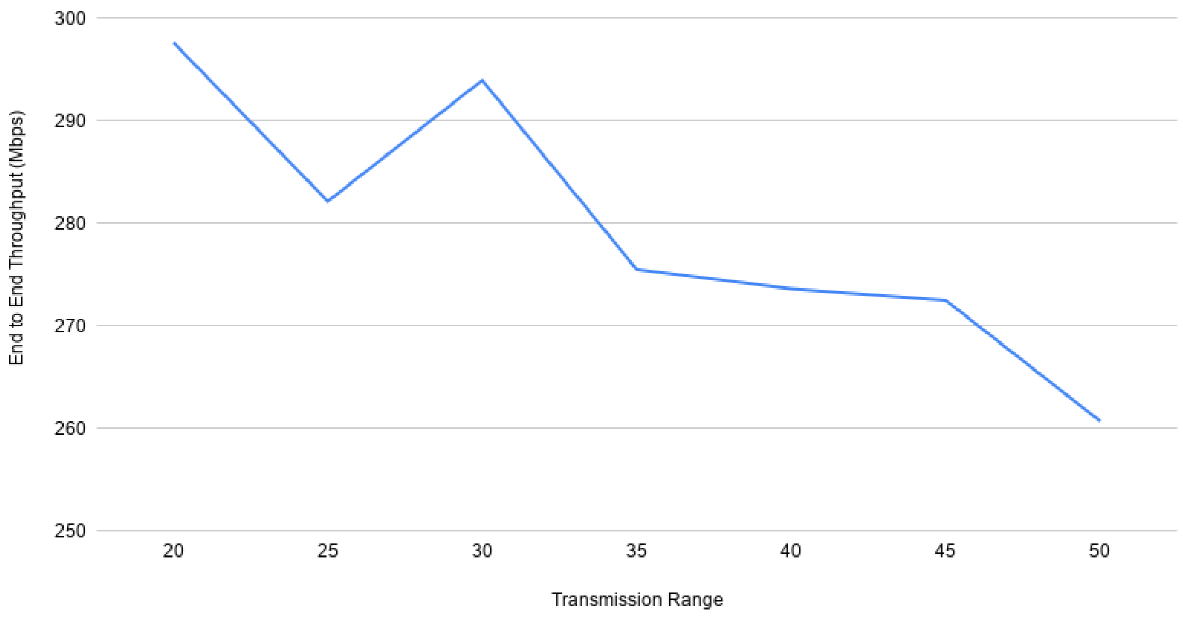

Figure 5: Average end-to-end network throughput of 3DMA implementation, with node count of 250

The data collected for throughput with varying transmission ranges actually relatively resembles the original data. This is due to less hops required for larger transmission ranges which results in a lower network throughput.

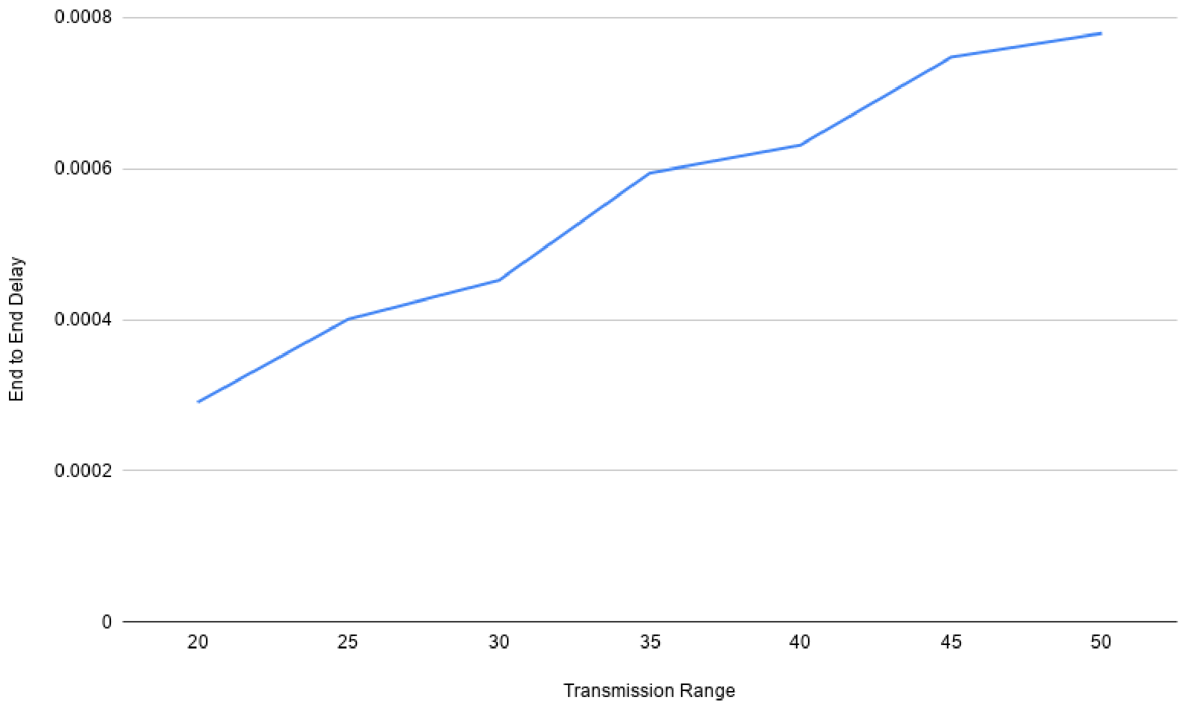

Figure 6: Average end-to-end network delay of 3DMA implementation, with node count of 250

Both our results and the results of [1] increased in delay as transmission range increased.

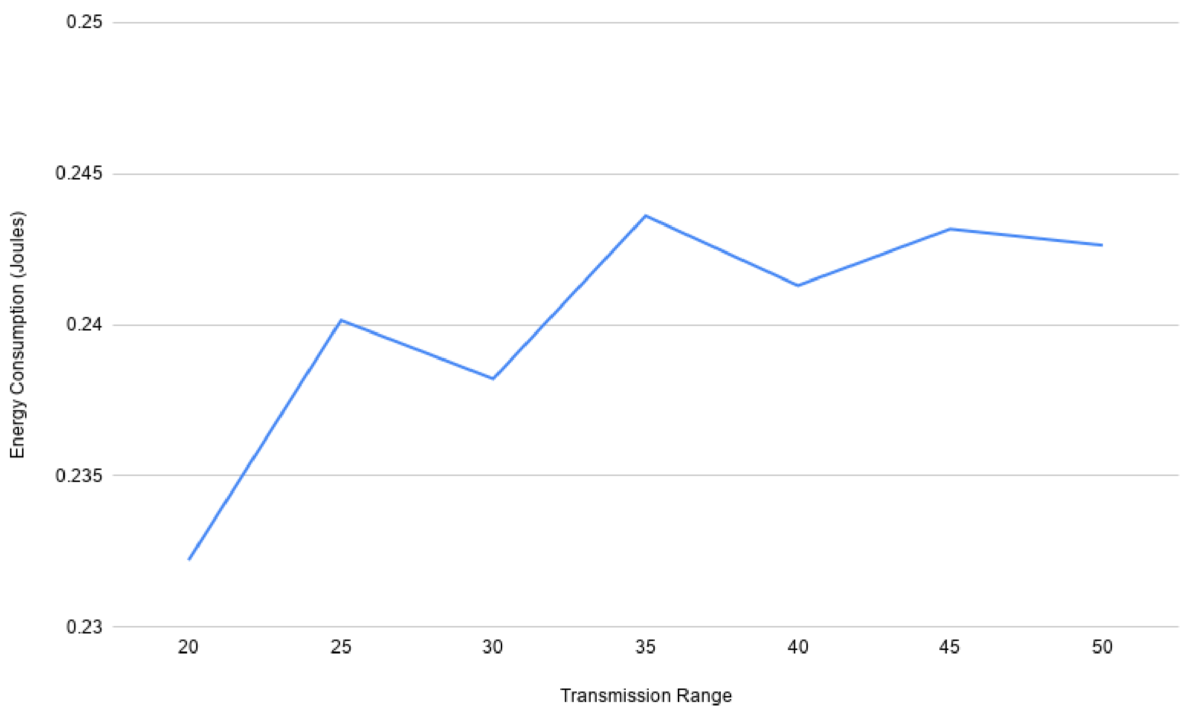

Figure 7: Average network energy consumption of 3DMA implementation, with node count of 250

Both our results and the original results increased in energy consumption as transmission range increased. This is likely because as nodes have a further reach, they are more likely to be able to be within range of more neighbour nodes, increasing potential to be an IM.

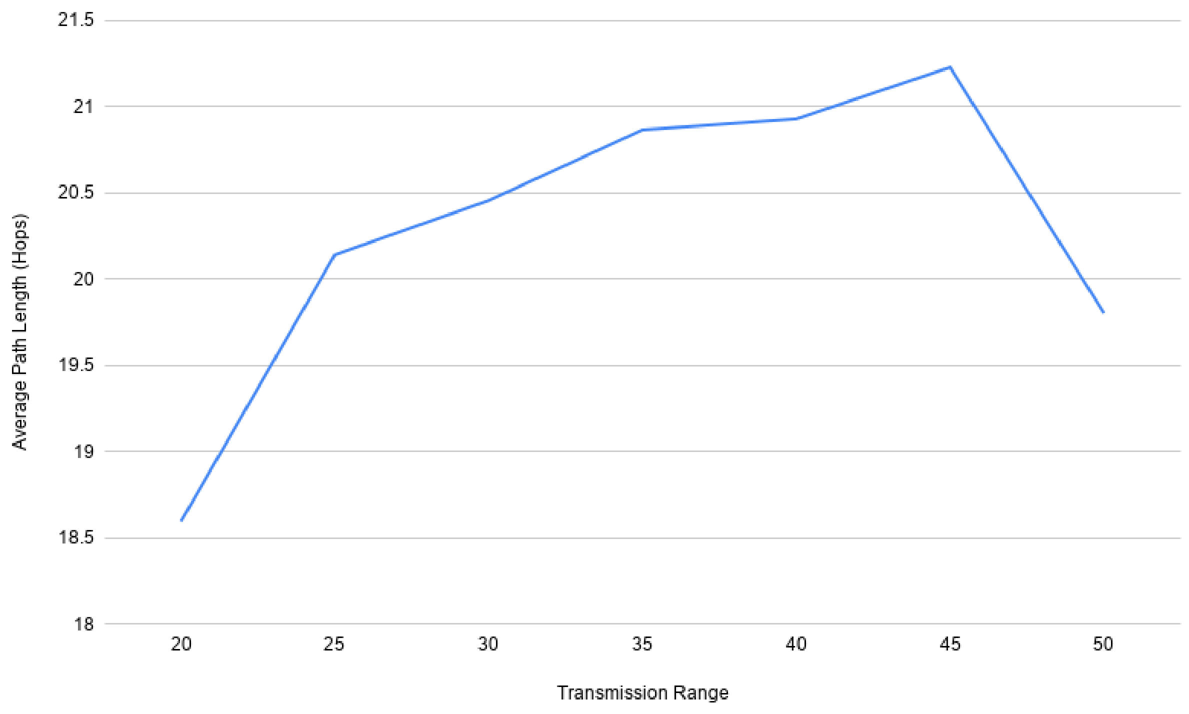

Figure 8: Average path length of 3DMA implementation, with node count of 250

Our data of average path length does not relatively match the original data at all. The original observed an overall downward trend, while we observed an increase of average path length overall (with a sudden drop at the end).

There is an aspect of our implementation of 3DMA that does not match up with [1]'s implementation. We believe this is due to the uncertainties of their method of data collection. We are unsure that the methods we used to collect data were the same in [1].

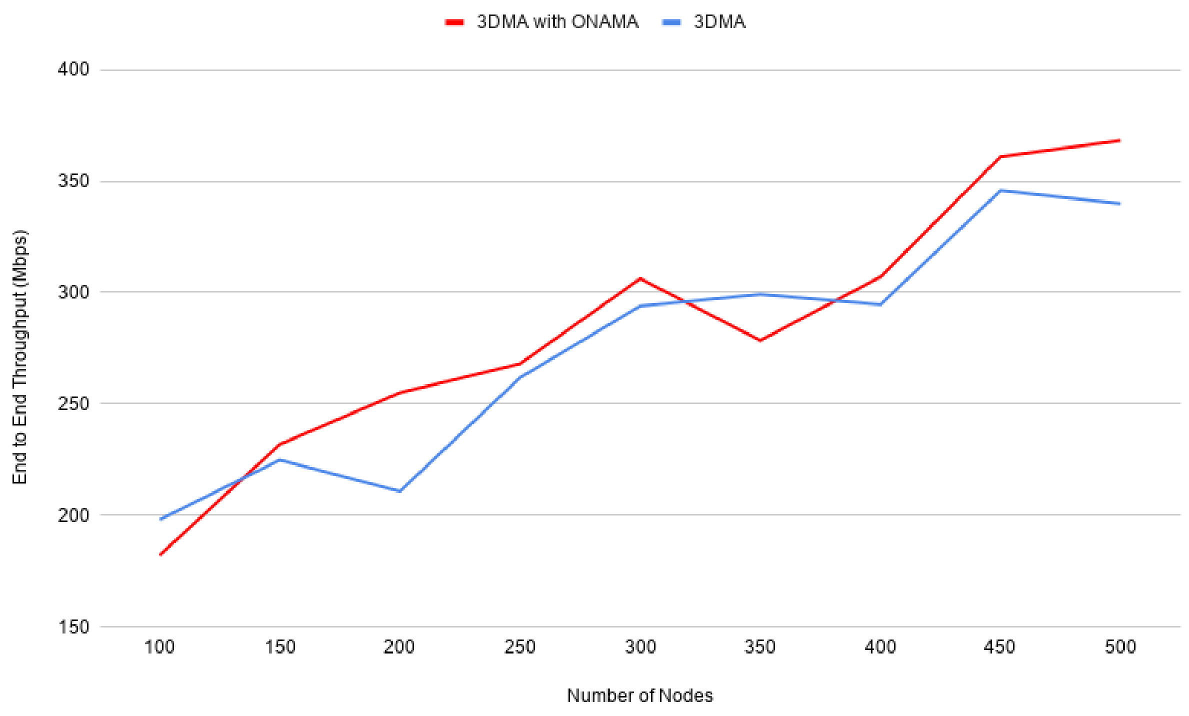

Figure 9: Average end-to-end network throughput of 3DMA with ONAMA implementation, with node transmission range of 25

Both 3DMA and 3DMA with ONAMA generated nodes in a similar manner. Therefore, variability in the end-to-end throughput is a result of the randomness. The two data sets are fairly consistent.

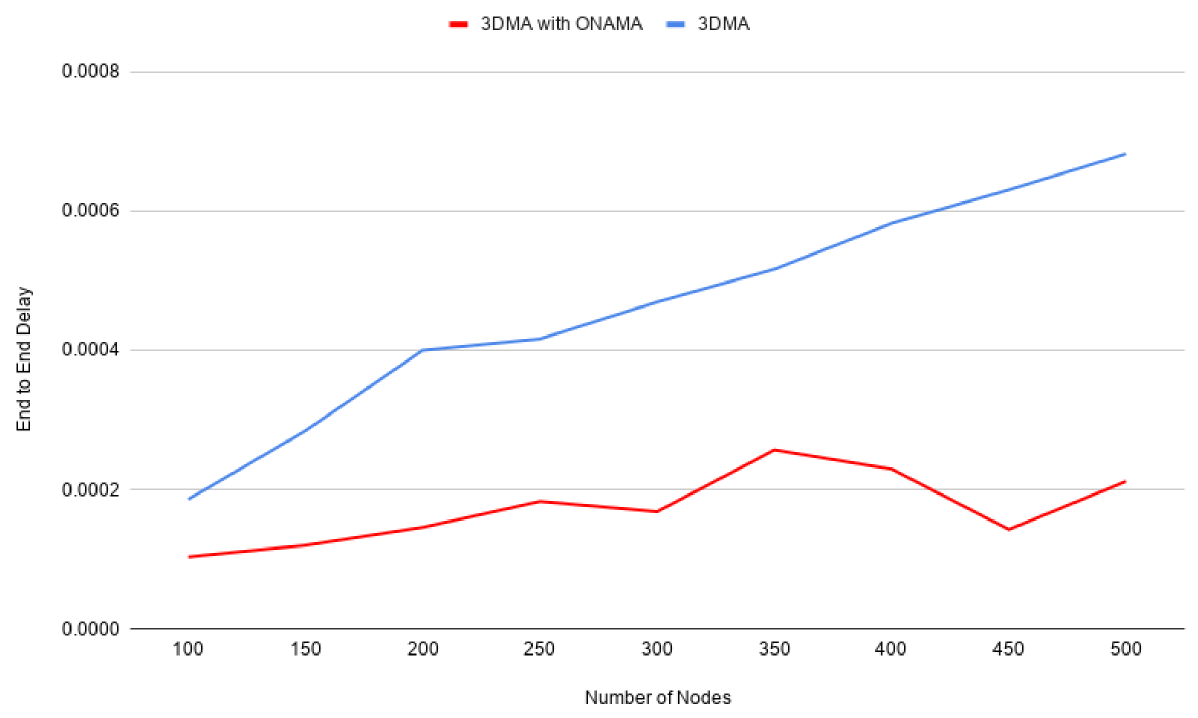

Figure 10: Average end-to-end network delay of 3DMA with ONAMA implementation, with node transmission range of 25

Here, 3DMA with ONAMA is observed to have a lower end-to-end delay compared with 3DMA. This is because nodes adjacent to the base station sent data more often than other nodes due to not sharing a path with other nodes. This lowered the overall end-to-end delay.

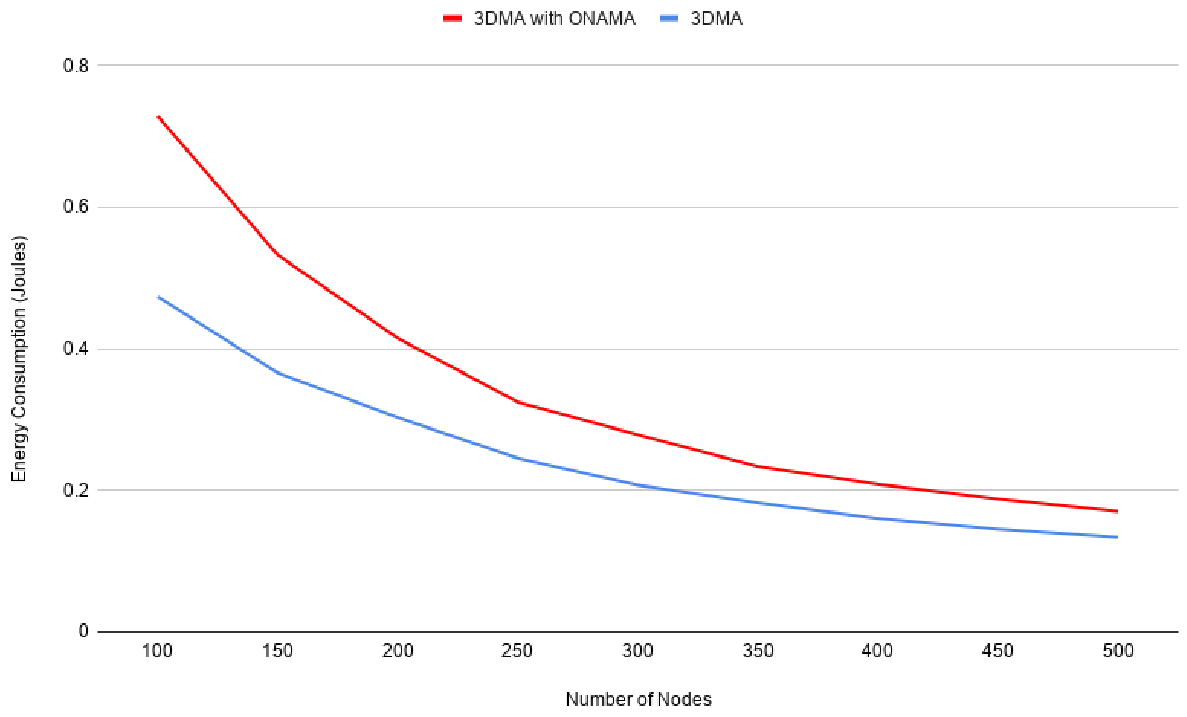

Figure 11: Average network energy consumption of 3DMA with ONAMA implementation, with node transmission range of 25

3DMA with ONAMA is observed to use more energy than 3DMA, and this result is expected because more nodes are sending or forwarding data on average, per time period.

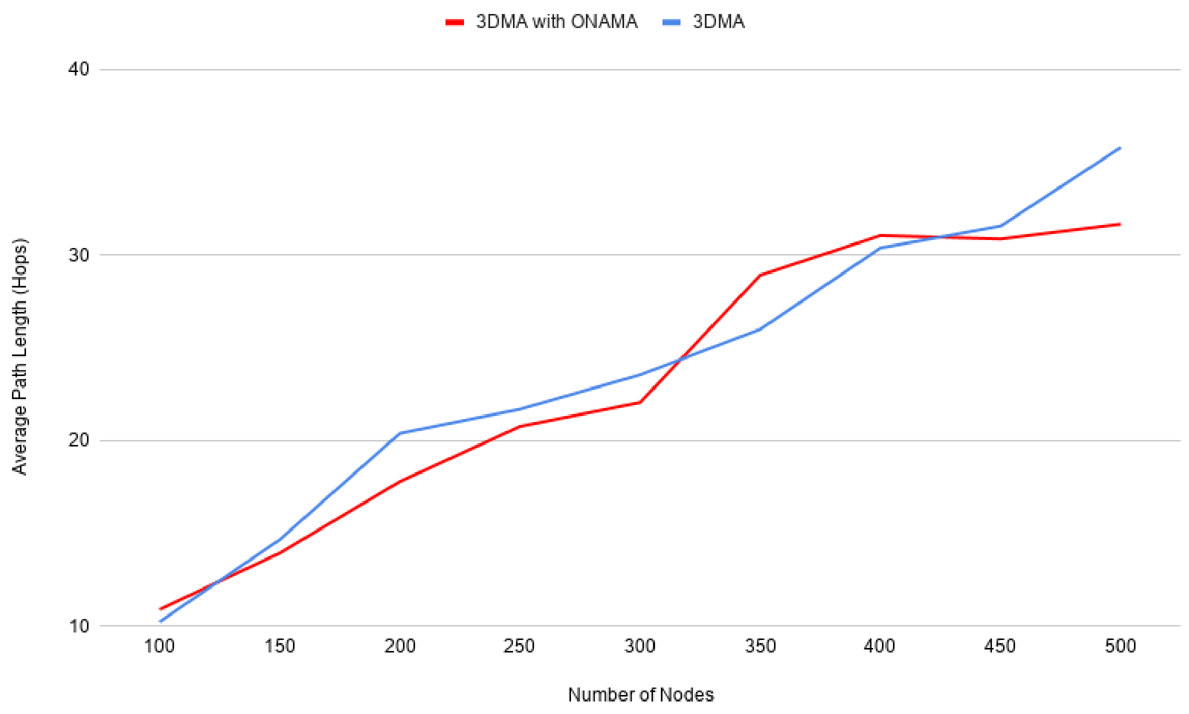

Figure 12: Average path length of 3DMA with ONAMA implementation, with node transmission range of 25

Like end-to-end throughput, both algorithms generate nodes similarly. Therefore, we have similar path lengths.

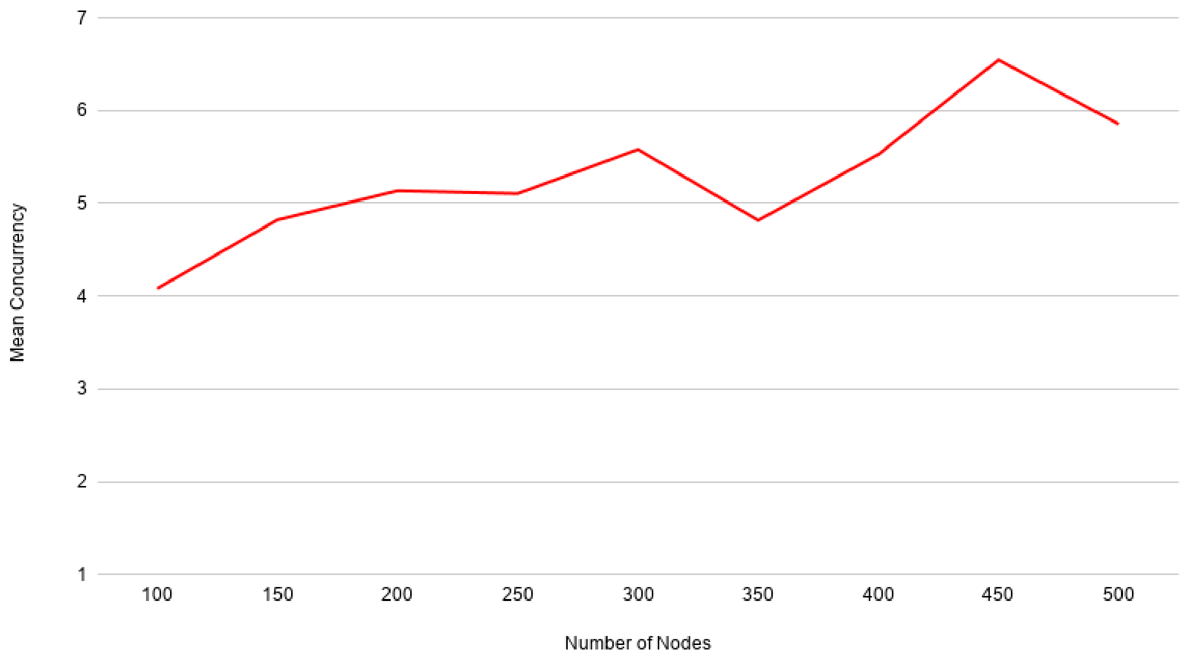

Figure 13: Mean concurrency of 3DMA with ONAMA implementation, with node transmission range of 25

Mean concurrency was certainly observed to be greater than one node per time period. Here, concurrency goes up as it is more likely that with more nodes, larger MISs will be formed as a result.

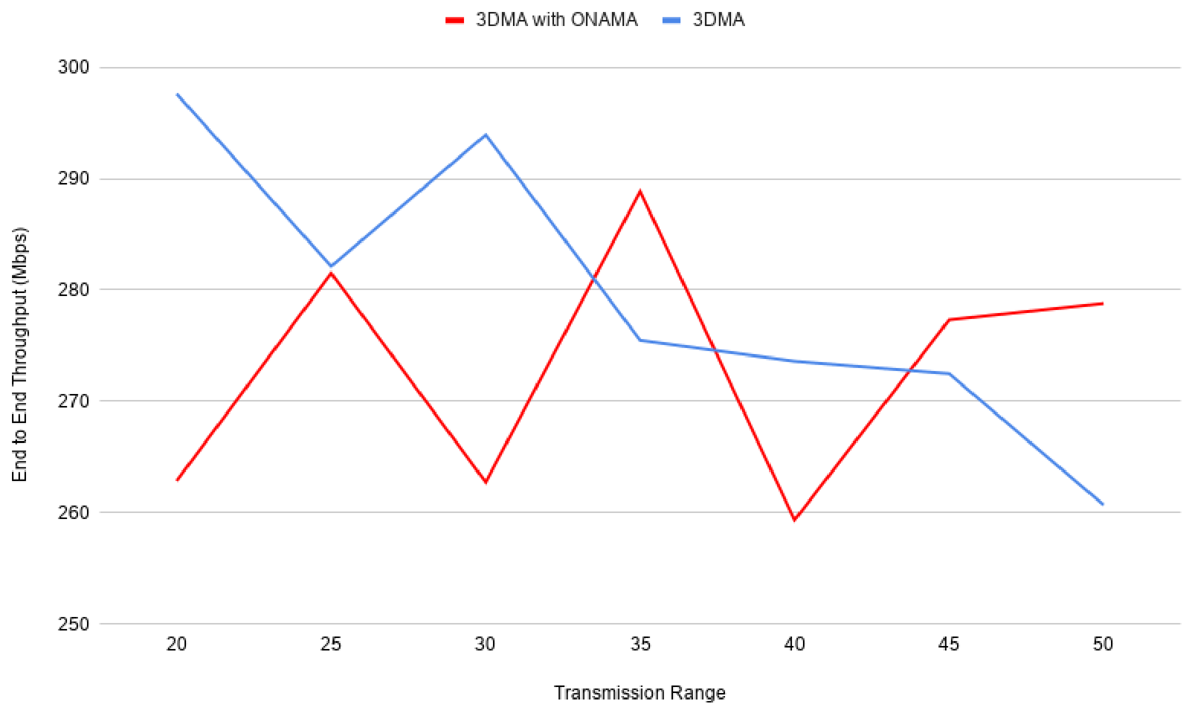

Figure 14: Average end-to-end network throughput of 3DMA with ONAMA implementation, with node count of 250

The variability of this graph is caused by the randomness due to node generation. The implementation of ONAMA should have no effect on the results of this graph.

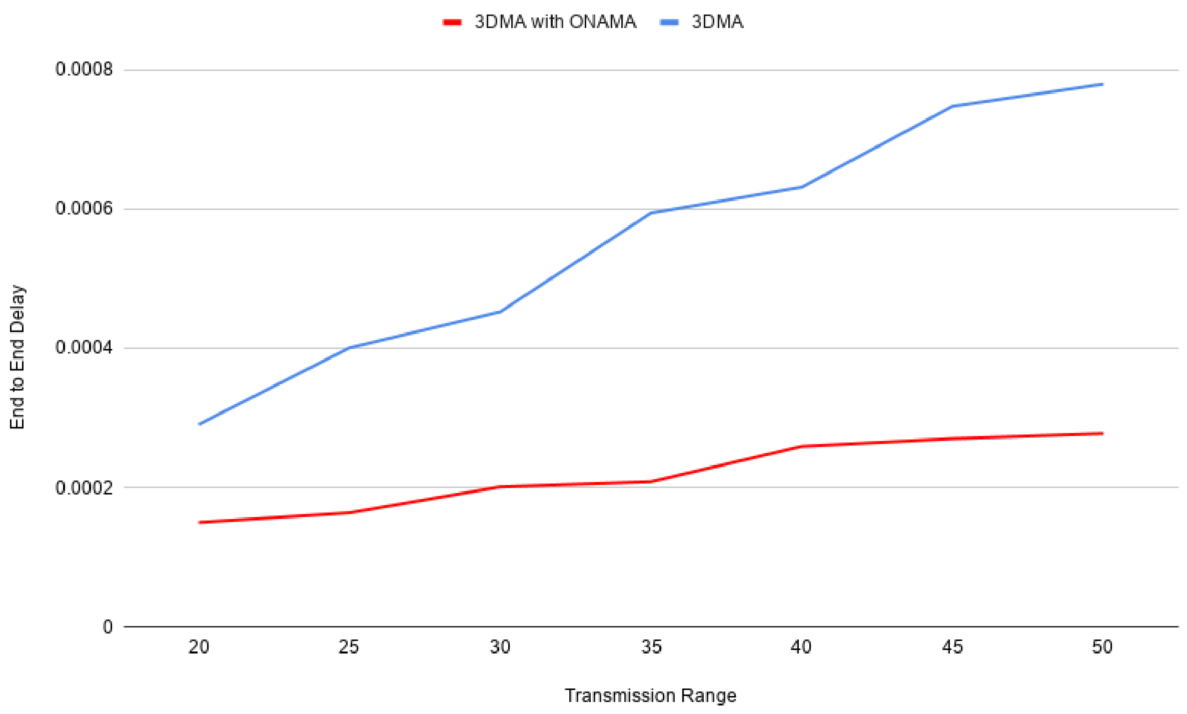

Figure 15: Average end-to-end network delay of 3DMA with ONAMA implementation, with node count of 250

Similar to Figure 10, ONAMA has nodes adjacent to the base station sending more frequently which lowers the overall delay.

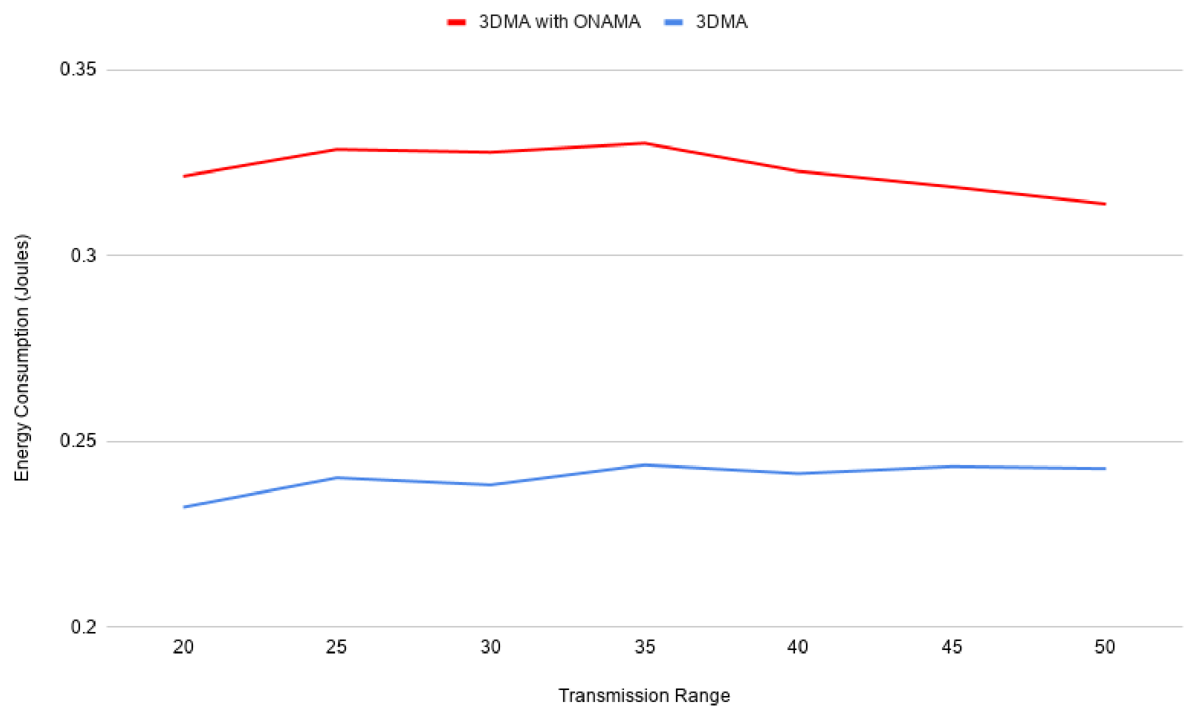

Figure 16: Average network energy consumption of 3DMA with ONAMA implementation, with node count of 250

Due to ONAMA allowing for more than one node to send at a time, the overall energy consumed was higher.

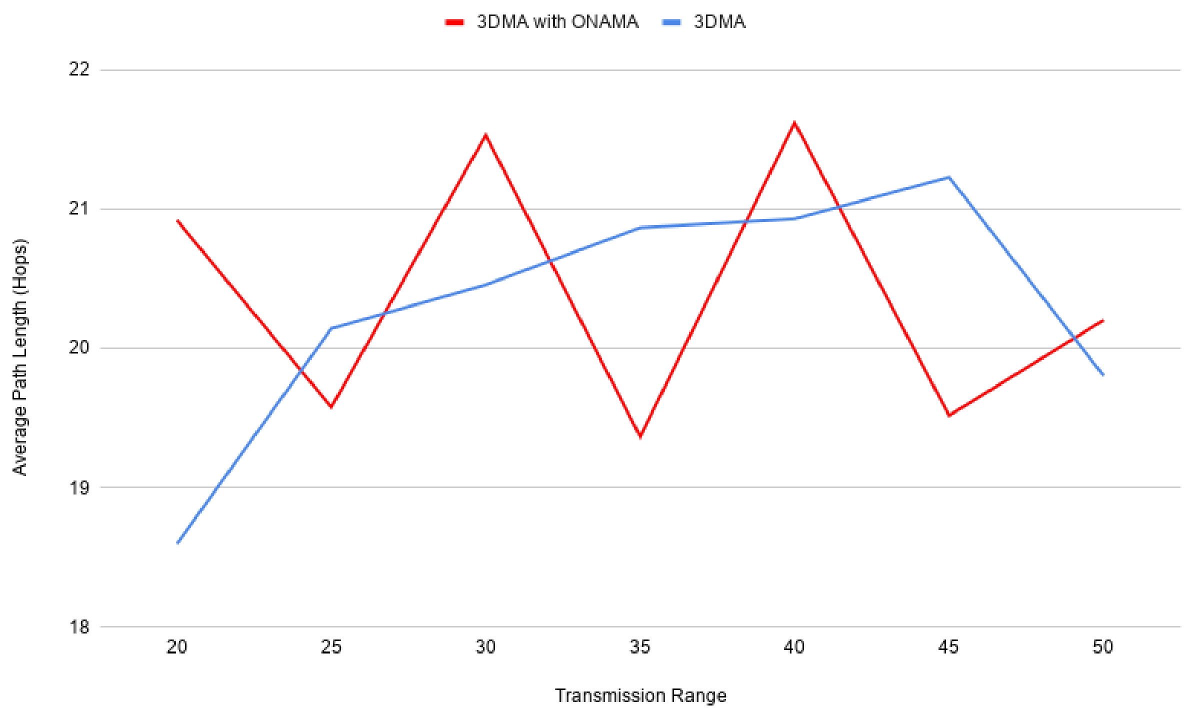

Figure 17: Average path length of 3DMA with ONAMA implementation, with node count of 250

Due to the randomness of node generation, paths are generated with a difference of 1 or 2 nodes. Overall, the values do not vary significantly from each other.

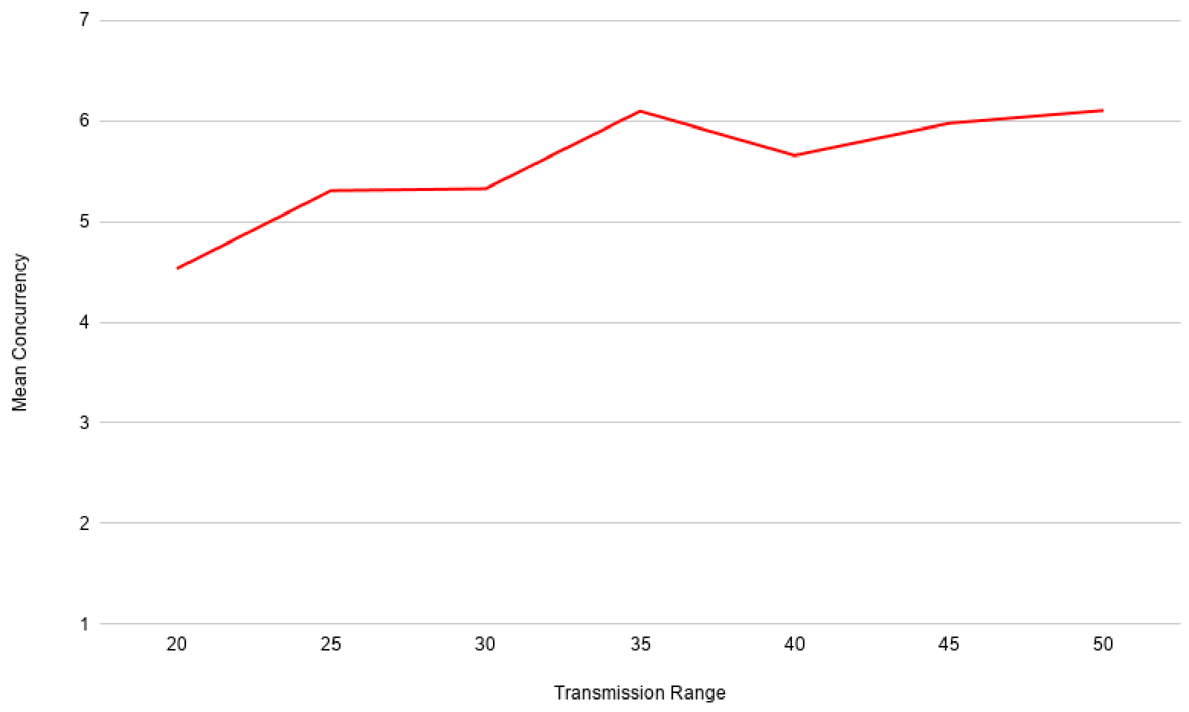

Figure 18: Mean concurrency of 3DMA with ONAMA implementation, with node count of 250

Mean concurrency goes up in the 3DMA with ONAMA implementation because on average, more than one node is sending per time period.

The results obtained from the implementation of 3DMA and ONAMA suggest that concurrency in the routing protocol, 3DMA, is very much possible. However, having concurrency will realistically cost more energy, due to the state exchange communication required to form MISs. Energy usage for state exchange was not incorporated into our simulations; this likely led to lower energy consumption for ONAMA than what it should have been. 3DMA with ONAMA may be suitable for multi-hop wireless sensor networks that are constrained by time, where the window to collect information is limited.

5 Conclusion

The three-dimensional wireless multi-hop routing protocol, 3DMA is not capable of concurrent communications due to interference between neighbouring nodes transmitting at the same time. A scheduling protocol for wireless networks, ONAMA, was proposed to be integrated with 3DMA in order to introduce concurrent transmissions without interference. We found that mean concurrency increased amongst sensor nodes, at the cost of more energy expended. Such results were obtained through implementation of the 3DMA algorithm, and then integrating ONAMA into it. Trials were conducted with varying transmission ranges and number of nodes in order to collect end-to-end throughput, end-to-end delay, average path length, energy consumption, and mean concurrency. We have shown the possibility of a more concurrent multi-hop wireless protocol in combining 3DMA with ONAMA that is applicable for time-constrained sensor networks.

6 List of Acronyms

IM	Intermediate Node
3DMA-CS	3D Minimum Angle-Centralized
3DMA-DS	3D Minimum Angle-Distributed
VNP	Void Node Problem
NAMA	Node Activation Multiple Access
ONAMA	Optimized Node Activation Multiple Access
MIS	Maximal Independent Set
DMIS	Distributed Maximal Independent Set
PDR	Packet Delivery Ratio

7 References

[1] A. Begum and M. Hussain, “A simple void node and loop-free three-dimensional routing protocol for multi-hop wireless networks,” *International Journal of Communication Systems*, vol. 33, p. e4185, aug 2020.

[2] L. Xiaohui, Y. Chen, and H. Zhang, “A maximal concurrency and low latency distributed scheduling protocol for wireless sensor networks,” *International Journal of Distributed Sensor Networks*, vol. 11, pp. 1–8, aug 2015.

[3] K. AboodOmer, “Analytical study of MFR routing algorithm for mobile ad hoc networks,” *Journal of King Saud University - Computer and Information Sciences*, vol. 22, pp. 29–35, may 2009.

[4] K. Jain, J. Padhye, V. Padmanabhan, *et al.*, “Impact of interference on multi-hop wireless network performance,” *Wireless Networks*, vol. 11, pp. 471–487, jul 2005.

[5] C. Jaikaeo, “Simpy-based WSN simulator.” https://gitlab.com/cjaikaeo/wsnsimpy/.