ASSIGNMENT - 2

Simulation of GHS Algorithm for finding Minimum Spanning Tree over a Distributed System

Shubhankar Suman Singh (2016CSZ8113)
Venkata Ramana Sreevathsa Meesala (2013CS10262)

OVERVIEW

In this assignment we have simulated Gallager Humblet Spira (GHS) For calculating Minimum Spanning Tree over a network in a distributed fashion. Each Node is simulated a separate thread in Java and nodes communicate with each other using messages and predefined message queue(implemented using Concurrent Lists Java).

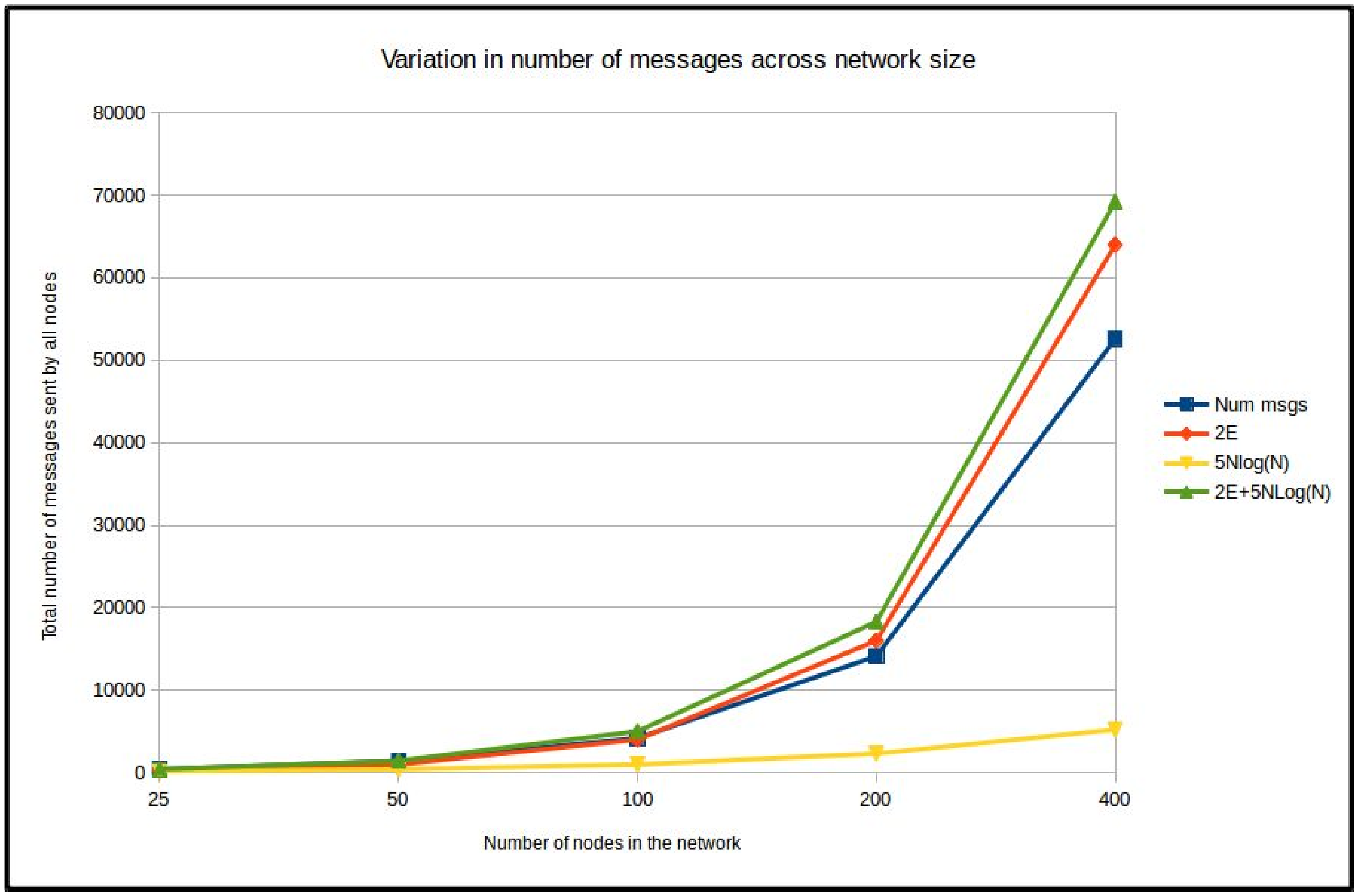
We have written an input_converter.py which converts the edge list into an adjacency matrix which is the desired input to the GHS program. Similarly we have written an output_converter.py which converts the Graph into the desired output format.Error detection was done by keeping separate log files for each node and automated scripts.

RESULTS

We have simulated a network of 25, 50, 100, 200 and 400 nodes. We have recorded the logs and we have calculated the total no of messages taken to create the MST and plotted the graph.

The theoretical result is that the the maximum number of messages is bounded by **5*N*log(N)** + **2*E** which matched with our simulations. (E is the number of edges in the network and N is the number of nodes in the network).

We have created random graphs where the ***probability of an edge existing*** *is* ***0.2*** *(It is a sparse graph).*

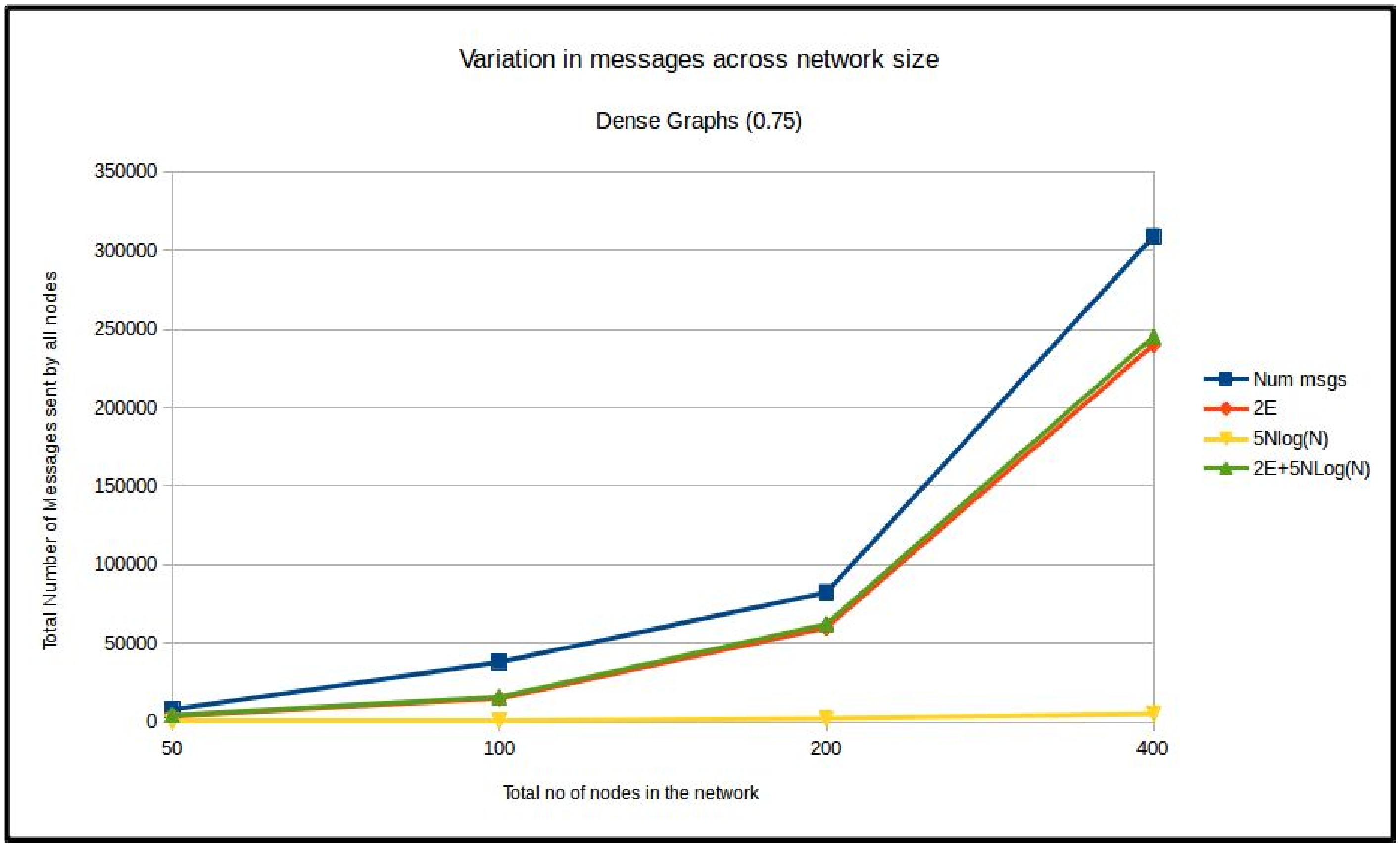


As you can see the actual number of messages less than 2E + 5NlogN.
The total number of messages sent are significantly less than the upper limit here. The algorithm runs faster for sparse graphs compared to dense ones.

Here are the actual Numbers obtained in the simulation

N	25	50	100	200	400
No of messages	434	1384	4160	14113	52583
2E	250	1000	4000	16000	64000
5NlogN	174	424	1000	2301	5204
2E + 5NlogN	424	1424	5000	18301	69204

Results for random graphs where the probability of an edge existing is ***<u>0.75</u>*** *<u>(It is a dense graph).</u>*



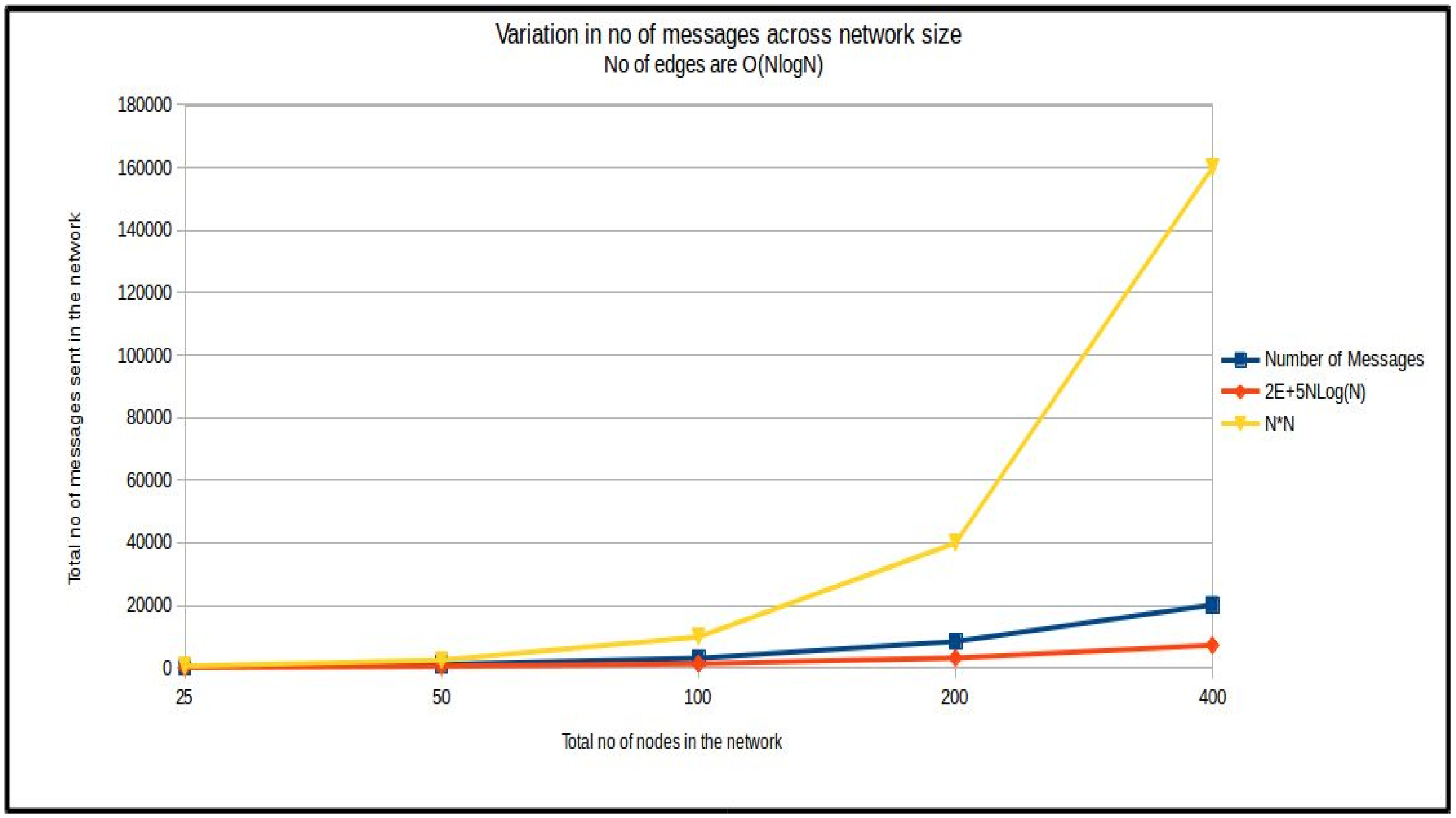
Here the number of messages are higher than the the upper bound. There are one possible reason why this may be happening.Our estimate for the number of edges is not exact. We are approximately taking it to be 0.75*N*N. This may be inaccurate.

Here are the actual numbers obtained in the simulation

N	50	100	200	400
No of messages	7889	38131	82463	308887
2E	3750	15000	60000	240000
5NlogN	424	1000	2301	5204
2E + 5NlogN	4174	16000	62301	245204

In order to confirm that our the number of messages are actually of O(E + 5NlogN) we have created graphs where the number of edges are of order NlogN.

Here are the results.



Here we can clearly see that the no of messages are also in the same order as NlogN. This proves that the messages are not growing in O(N*N) but actually growing of O(E).

Here are the actual numbers from the simulation.

N	25	50	100	200	400
No of messages	482	1229	3111	8542	20194
2E + NlogN	244	594	1400	3221	7285
N*N	625	2500	10000	40000	160000

CONCLUSION

The GHS algorithm is working perfectly for various network sizes and random networks. The number of messages are quite large and as the network size increases the time taken increases by O(E + NlogN) - If the graph is dense then the no of messages increase polynomially E = N0*N (This becomes the dominating factor). If the graph is sparse (E < NlogN or E ~ O(N)) then the number of messages increase logarithmically O(NlogN) (dominating term is the NlogN)

For large size networks like the internet GHS algorithm does not scale.