A Monte Carlo analysis of dynamic sensor node routing in various mesh networks

Authors:
Lloyd Gonzales, Vincent Lee

Additional files:
https://github.com/lgactna/cpe-400-final (source code attached with report)

Class:
CPE 400, Spring 2023
Tapadhir Das

Table of Contents

Abstract

Sensor node networks often operate under different constraints and topologies than conventional networks. They often must be fully battery operated due to their placement and environment, requiring that routing algorithms optimize for battery life. Existing pathfinding algorithms, such as Dijkstra's, can be suitable if certain conditions are met.

One possible application of such a network is in sending data to a central command and control center, or C2. It may not be the case that all sensor nodes have the transmission power or range to communicate directly with the C2. Thus, a mesh network must be constructed in which each node must forward data from "upstream" nodes further away from the C2. Such a network may be particularly relevant in the context of extraterrestrial networks, in which there is little preexisting infrastructure.

Here, we use a Monte Carlo approach to determine the viability of various network topologies in a naive dynamic routing algorithm. We present the impact of certain network features on the operational lifetime of the network, defined as the time during which all nodes have nonzero energy, as well as other core metrics that may be more important in specific applications. We show that the quality of "balance," or the equal distribution of work, within upstream nodes is the most critical factor in the operational lifetime of such dynamically-routed sensor node networks.

Introduction

Sensor node mesh networks

A mesh network is a form of network in which each node connects directly and dynamically to other nodes. Typically, these networks are relatively reliable and easy to configure; because each node can forward data to and from any other node, it means that each node is capable of configuring and organizing itself within the network without needing a centralized controller to exist. This contrasts with the traditional star network topology, in which links are hierarchical and certain nodes hold more "responsibility" than others.

Because of the "plug-and-play" nature of mesh networks, they often use a dynamic routing system such as OSPF. OSPF operates using Dijkstra's algorithm, in which the global state of all hosts within an autonomous system is maintained. Changes within the network are quickly conveyed to other nodes, maintaining the balance and reliability of the network. However, maintaining the global state of all hosts within an extensive mesh network is infeasible; indeed, OSPF is only "global" within a single autonomous system. More importantly, it assumes that the cost of maintaining an accurate global state does not outweigh its benefits, which may not be the case in certain niche systems.

One potential application of a mesh network is creating a network of sensor nodes in which a star topology is impossible. For example, sensor nodes are often used to collect data in remote regions, where connecting the sensor directly to a power grid is infeasible. Often, these nodes cannot also depend on other independent sources of electricity, such as solar energy. In these cases, the nodes must be fully battery-operated; since the sensor node is in a remote region, it is difficult to replace the battery.

Sometimes, such networks can be supported with more capable infrastructure. For example, consider the barbell graph in Figure 1.

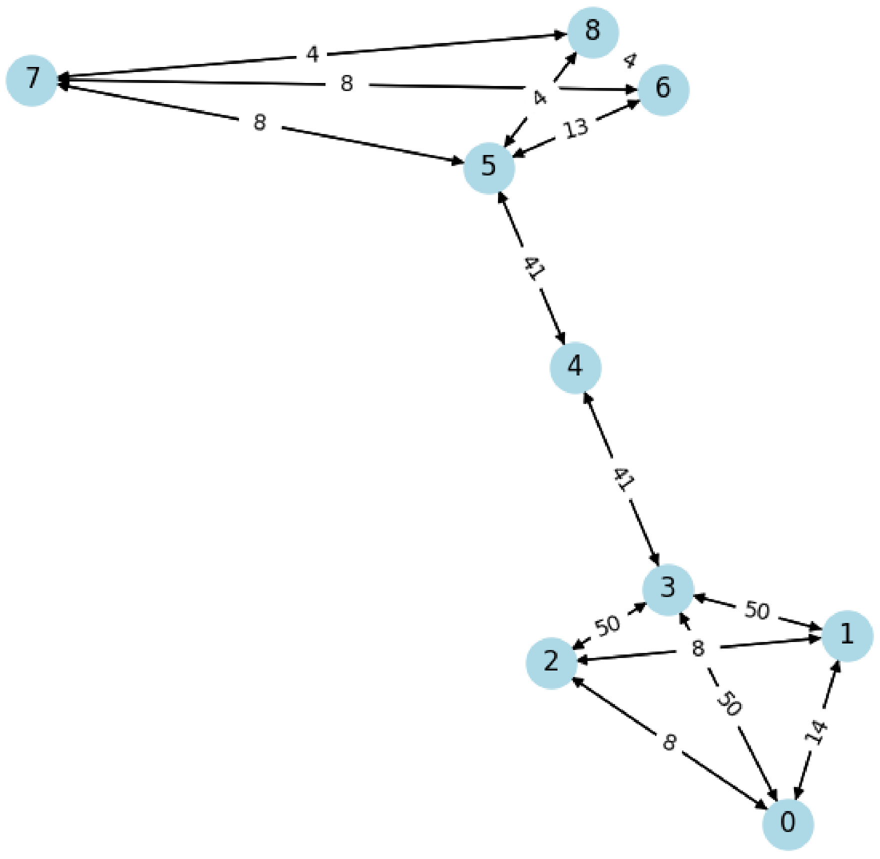


Fig. 1: A barbell graph of four nodes at each connected “bell,” with one node supporting the connection from the two cliques at nodes 3 and 5. The edge labels, while significant, are not relevant to this discussion.

Here, nodes 0 through 3 and nodes 5 through 8 form complete subgraphs. Notice that nodes 3, 4, and 5 can serve as the “backbone” of the mesh network. If these nodes have a significantly greater battery capacity – or even infinite – compared to other nodes, they can easily support the transfer of data from each subgraph to the other. This is largely similar to the routing performed in many protocols, in which a hierarchy of more capable systems supports less capable systems when routing.

However, it may be that more capable infrastructure to support the system does not exist, as is currently the case in extraterrestrial environments. Therefore, these nodes must be able to collect, transmit, and forward data from other equally low-power nodes for as long as possible. With the increasing trend towards future space travel and the need for medium- and long-term infrastructure on other celestial bodies, where satellites and power grids are nonextant, research must be performed into viable network configurations in these conditions. We now present a novel case of a possible sensor node configuration in which reasonable infrastructure limitations and conditions exist on the network.

Scenario

Suppose that in an extraterrestrial environment, a variety of sensor nodes are deployed in a mesh network configuration, which must coordinate with each other to forward data to a command and control center, or C2. Although the C2 has the capability to make long-range broadcasts of data and perform computationally-expensive processing, individual sensor nodes do not. As a result, these sensor nodes primarily exist to gather data and possibly forward upstream data to some central point for further processing.

This is particularly significant in that it means nodes that are further away from the C2 place greater stress on the network than those that are closer to the C2 – the greater the depth of a sensor node relative to the C2, the greater the number of nodes that must expend energy forwarding data from that distant node. However, regardless of the distance of any sensor node from the C2, that sensor node is always within range of any broadcast from the C2. This means that network overhead in this scenario, such as global state, can always be transmitted to the entire network.

In such sensor node mesh networks, the quality of the system can degrade rapidly after any node loses all of its battery, as it increases the probability that only a small number of paths from one group of sensor nodes to another exists. That is, the (permanent) failure of a small set of unlucky nodes can create a chokepoint in which one node is forced to forward all the traffic from upstream nodes, as shown in Figure 2.

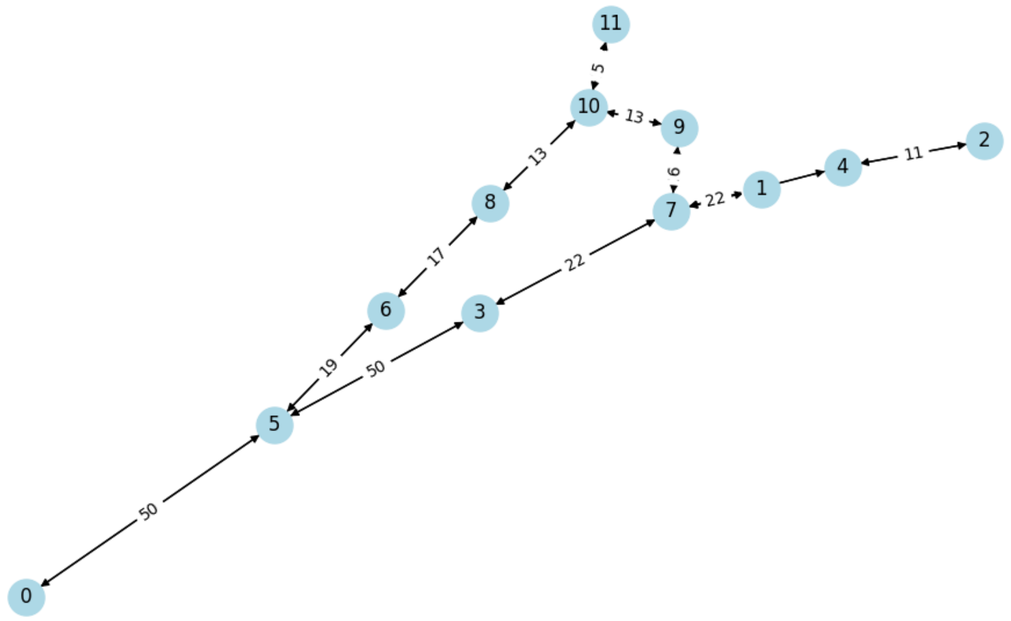


Fig. 2: When node 0 is chosen as the C2, node 5 is forced to forward the traffic of every upstream sensor node in the network, making it likely to run out of battery first.

Additionally, we can reasonably assume that certain sensor nodes are “essential” to the scenario; that is, some of the sensor nodes in our network do not exist for the sole purpose of forwarding data but to collect significant data themselves.

Therefore, we define the operational lifetime of a sensor node network as the time from which data is first sent from any sensor node until the time any sensor node runs out of battery. Since the goal is to optimize the operational lifetime, such nodes must keep track of the lifetimes of their neighbors and use such information to determine which nodes to forward information to. This contrasts considerably with the goals of other algorithms, which often optimize for latency, “cost,” or reliability – even if sending a message through high-battery nodes means that it takes many extra hops, we would rather accept this cost than kill a low-battery node. Simultaneously, we also want to avoid needing an excessive number of sensor nodes just to support forwarding.

As mentioned earlier, many mesh networks use a dynamic routing system based on global state maintained through overhead messages periodically sent throughout the network. However, in the case of the mesh sensor node networks described above, this can be infeasible; a single broadcast message decreases the lifespan of every forwarding node. Therefore, we assume in this scenario that the nodes are not responsible for maintaining global state themselves since the C2 can broadcast information to the sensor nodes at no cost to the sensor nodes themselves.

For this scenario, we present empirical results on the performance of various network topologies in forwarding data from sensor nodes to a single destination, symbolically representing a C2. To consider many possible scenarios and identify common trends between simulations, we use a Monte Carlo approach in which the sequence of transmitting nodes varies from simulation to simulation. Here, we compare the average lifetimes and metrics of various networks against many simulations, such that the nodes forwarding data to the C2 vary from run to run.

Methodology

Key assumptions

In addition to the scenario presented, we assume the following characteristics of the network:

- The cost of collecting data and receiving transmissions is negligible compared to the cost of transmitting data. That is, if a node never needs to forward another node's data or send any data itself, it will never lose battery.
- The nodes and links are infallible; that is, packets sent over links will always reach their destination (unless the sensor node is dead, at which point the network is considered inoperable), and nodes will never become temporarily or permanently inoperable for any reason other than running out of battery. In other words, the probability of a link failing is zero, and the probability of a node failing is zero. The impacts of this assumption will be addressed in a later section.
- The initial battery level of each node is the same and known throughout the network. By extension, this means that the network topology remains constant throughout a single simulation, such that no new nodes are added to or removed from the network.

The assumptions above are critical to ensuring the following fact in our simulations: it is possible to evaluate Dijkstra's algorithm from any node.

That is, the global state is always known. This partly follows from assumptions 2 and 3 above. If every node knows the initial state of the entire network and messages never fail to reach their destinations, then the global state of the network can always be calculated at the C2 based on the complete path taken by each message that arrives.

As mentioned earlier, we also assume in our scenario that the C2 has the ability to make a long-range broadcast to all sensor nodes, such that the global state can be maintained at each sensor node without the sensor nodes having to "advertise" their current battery capacities themselves.

Algorithm and implementation

In our first round of simulations, we assumed that an arbitrary node labeled 0 was the C2 – that is, all transmissions in the network are destined for node 0. Each node was associated with a battery life representing the number of transmissions that can still be made by that node, with each edge in the network weighted by the number of transmissions done across that edge.

We used a directed graph to represent our networks internally – in practice, because only certain edges make meaningful progress towards the C2, these are the edges graphed in our figures (matplotlib only shows the weight of one edge, not both). For a graph with only positive weighted edges, it is never more efficient to traverse nodes in a connected subgraph before going to a “chokepoint” node, as opposed to just going to the chokepoint node directly. A simple example is that a message would never route through a leaf that isn’t the C2 itself.

Implicitly, this means that each outgoing edge in the network going towards the C2 is weighted inversely to the battery life of the source node of each edge. For example, if a source node currently has a battery life of 50 transmissions, and its original capacity was 150 transmissions, then its outgoing edge towards the C2 has a weight of 100. By doing this, the least-cost path found by Dijkstra’s algorithm represents the path to the C2 that uses the nodes with the greatest total remaining battery life.

While possible that a modified pathfinding algorithm may perform better, such as one that considers the battery life of individual nodes rather than complete paths, we use Dijkstra’s here for its simplicity and its use in existing routing protocols.

Our simulation accepted both undirected and directed connected graphs; undirected graphs were simply converted to digraphs, with each undirected edge replaced with two opposing edges, thus ensuring that one “downstream” edge to the C2 exists. In any given simulation of a network, our algorithm does the following:

1. Select a specific node in the network to be the C2.
2. Sample a random node from the network, except for the C2 itself. This is symbolic of the event where a random sensor node needs to transmit data to the C2.
3. Use Dijktra’s algorithm to find the least-cost path from the random node to the C2 based on global edge weights.
4. Select all nodes in the path, excluding the C2, and reduce their battery life by 1.
5. Select each edge of the nodes in the path and increase their outbound weights by 1.
6. Check if any node in the graph has a battery life of 0. If so, terminate the simulation; else, return to step 1.
7. Return the total number of transmissions completed, the average amount of energy used, and the final graph with modified attributes.

After this, the same procedure was done with the C2 as each possible node in the network to fully evaluate the effects of changing the C2 in the same network.

Our simulation was implemented in Python using the NetworkX graphing library, with matplotlib used for visualizing graphs. The source code can be found at https://github.com/lgactna/cpe-400-final.

Results

To gather information on the performance of the algorithm against different network topologies, we generated five classical network topologies and ran 100 simulations on each topology. That is, the sequence of nodes that needed to transmit data to the C2 varied from run to run, but the underlying network topology remained constant. Additionally, we also generated five classes of "random" graphs using built-in NetworkX generators, representing less well-structured networks that may be reflective of scenario-specific requirements.

Although the broad goal is to maximize the total number of transmissions completed in the network, it is also worth considering the ratio of transmissions to total starting battery life across the network. While true that a larger network with more nodes is likely to have a greater number of transmissions before any single node fails, it may not be the case that the nodes are actually used efficiently; perhaps only 10% of the network's total energy is used. Additionally, we also collect routing efficiency, expressed as the number of hops per transmission, to help identify any patterns between other metrics.

Therefore, in each batch of simulations against specified topologies, we collected the following information:

- The number of transmissions successfully performed by the network before the simulation was terminated (that is, the number of transmissions before any node depleted its battery).
- The number of hops required for the transmission to reach the C2. For example, if a message must travel the path 4-3-2-1-0, this transmission requires four individual hops.
- The number of edges in the resulting graph. A higher edge density over the same 12 nodes symbolically indicates that the nodes are physically closer to each other, such that more nodes are able to form more edges. This is likely to contribute to better balancing across the network, improving the number of transmissions made before a sensor node runs out of battery. Of course, this effectively decreases the are of data collection from the sensor nodes themselves.

A summary of the simulations with node 0 selected as the C2 is described in Table 1. Note that node 0 is arbitrarily chosen for non-classical generated graphs; the actual position of node 0 in all networks can be seen in the Appendix.

Table 1: The results of running 100 simulations against various 12-node network topologies with the C2 at node 0, where all nodes had a battery life of 50 transmissions.

Topology/Network *n = 12*	**Average number of transmissions**	**Average percentage of total energy used**	**Average number of hops per transmission**	**Number of edges**
Mesh [Fig. B1]	431.2	71.87%	1.0	66
Star [B2]	476.11	79.35%	1.0	12
Wheel [B3]	429.31	71.55%	1.0	22
Ring [B4]	95.27	52.36%	3.3	12
Barbell [B5] *$m_1 = 5$* *$m_2 = 2$*	68.72	33.99%	2.97	23
Connected Erdős-Rényi [A1] *$p = 0.01$*	50.0	30.88%	2.84	11
Connected Caveman [A2] *3 cliques of 4 nodes each*	111.0	44.34%	2.86	18
Powerlaw Tree [A3] *$\gamma = 2$*	50.0	31.02%	3.72	11
Watts-Strogatz [A4] *$k = 4$* *$p = 0.25$*	193.35	50.85%	1.58	24
Barabási–Albert [A5] *$m = 3$*	323.9	69.12%	1.28	27

Note that the first five topologies are deterministic, classical graphs; the last five are randomized graphs. All nodes had a battery life of 50 transmissions, meaning that each node could forward 50 packets of data before dying. Note that under the definition of the simulation in the previous section, the transmission – and all of its hops – are counted, even if a node dies in the process of completing a transmission.

The values shown above are helpful in analyzing the impact of the selection of a particular C2 node in each topology and determining what factors contribute to the performance of the algorithm. However, while these values are helpful at a node-level analysis, they can hide the impact of overall network topologies for asymmetric graphs, where an amortized analysis may be preferred. The complete distributions for both the randomized and classical graphs are provided in the Appendix, depicting the performance of each network with the C2 rotated between every possible node in the network in each set of 100 simulations.

Analysis

Classical graphs

We begin by exploring the best-case and average-case scenarios for any ideal topology. An ideal topology is one where all transmissions take exactly one hop, and thus no additional forward is necessary; in turn, the battery life of a sensor node is simply tied to the number of transmissions it must make rather than the number of nodes that depend on it for downstream forwarding to the C2.

The absolute best-case scenario for a network in our simulation is that almost all energy is used across the entire network before a sensor node dies. That is, one of the 11 non-C2 nodes uses up all 50 of its transmissions, and the remaining ten non-C2 nodes use 49 of their 50 transmissions. That is, the upper bound for transmissions per simulation is 540 transmissions, amounting to a total energy usage of about 98.18%. (If the C2 is allowed to transmit to itself at its own expense, the maximum becomes 589 transmissions instead.)

Assuming that all nodes are equally likely to need to transmit data, the average case can be determined with empirical testing. Over 100,000 simulations of such an ideal network, a clear left-tailed distribution emerges, with 480 transmissions being the most common case, as shown in Figure 3. Note that the various discrete "peaks" are likely due to our method of creating the bins.

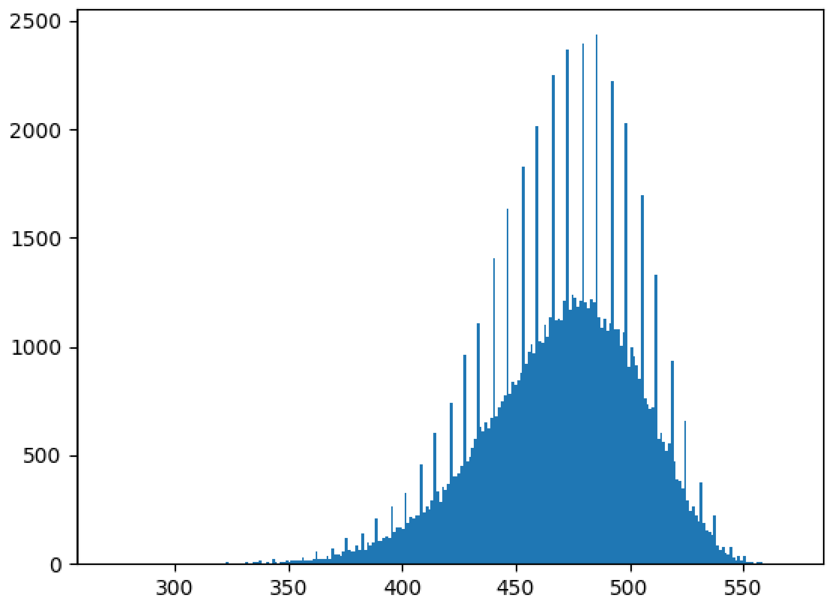


Fig. 3: The distribution of the number of transmissions in an ideal network over 100,000 runs.

From actual testing, the classical networks in which most or all sensor nodes have an edge directly to the C2 perform best, as few (or zero) nodes are responsible for forwarding "upstream" traffic to the C2. While there is a notable decrease in performance in the mesh and wheel topologies compared to the star topology, this is likely a result of the random number generator; indeed, an average of 1 hop per transmission indicates that all transmissions are "ideal" and could not possibly be routed any better. A snapshot of the network before the end of a star network simulation is shown in Figure 4. The edge weights represent the number of transmissions made over that edge, which also indicates the remaining battery life of each node.

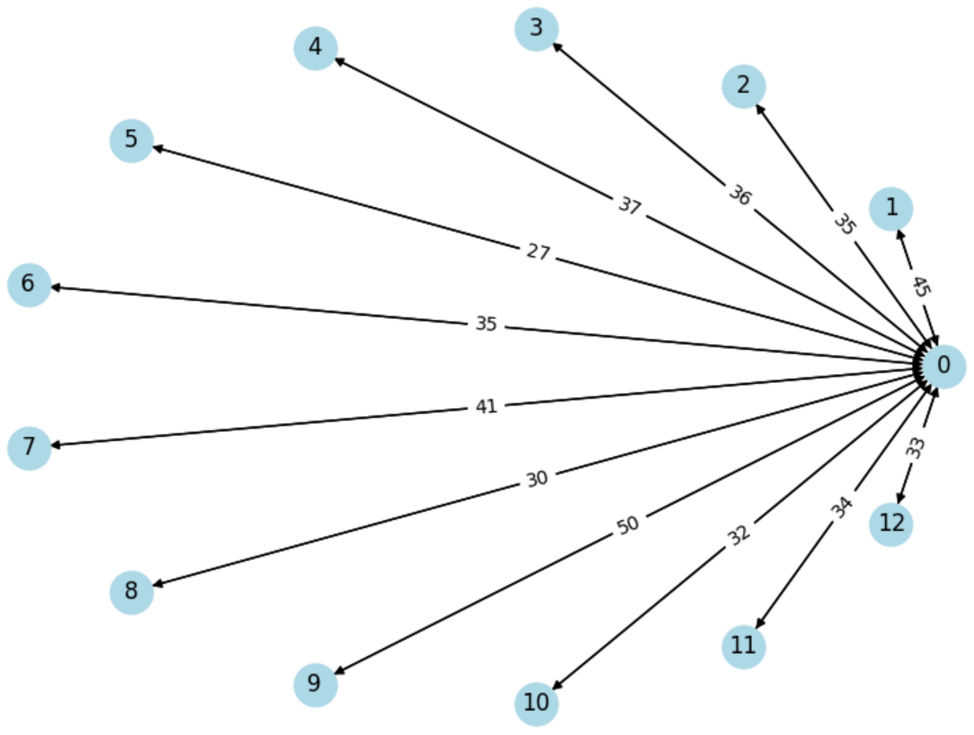


Fig. 4: The resulting edge weights in a sample simulation of a star network with node 0 as the C2. Notice that node 9 has an edge weight of 50 and therefore was the first node to run out of battery.

In such near-ideal cases, additional edges are unnecessary; if there is additional overhead in maintaining the unused edges to other sensor nodes (to which data may never be forwarded), the star topology is most performant in both edge efficiency and operational lifetime. Indeed, the average number of transmissions over 100 simulations for the star topology shown in Table 1 is the most out of the other topologies.

However, the position of the C2 dramatically influences the performance of these networks. While true that the star topology removes any excess edges while still being the most performant, as shown in Figure B2 in the Appendix, it is also most sensitive to a change in C2. For example, if the C2 is moved to one of the peripheral nodes of the star network, the performance drastically decreases. In this case, one node is responsible for forwarding the packets of all the nodes to the C2, capping the operational lifetime of the entire network to the battery life of a single node.

In contrast, the completely connected (mesh) graph shown in Figure B1 is completely insensitive to the location of the C2; indeed, because every single node is connected to each other, it is always the case that the C2 can be reached in exactly one hop. This is indicated in the completely flat distributions for the mesh graph in Figure B1. Similarly, the wheel graph depicted in Figure B3 still performs best when the "hub" of the wheel is the C2; however, because of the additional edges compared to the star network, it is possible for the network to live for longer if a peripheral node is the C2 instead. Since there are always three neighbors for any peripheral node, there are more routing options for forwarding messages before any node loses power.

Even if the "ideal" case of always having a direct connection to the C2 is eliminated, there are still benefits in having a symmetric graph. This can best be seen in Figure B4, in which the ring graph depicted has a flat distribution across its metrics. Regardless of the node chosen to be the C2, the relative paths available to the sensor nodes remain the same and do not introduce a single chokepoint in the network. In Figure 5, with node 0 as the C2, the C2's immediate neighbors have nearly identical edge weights.

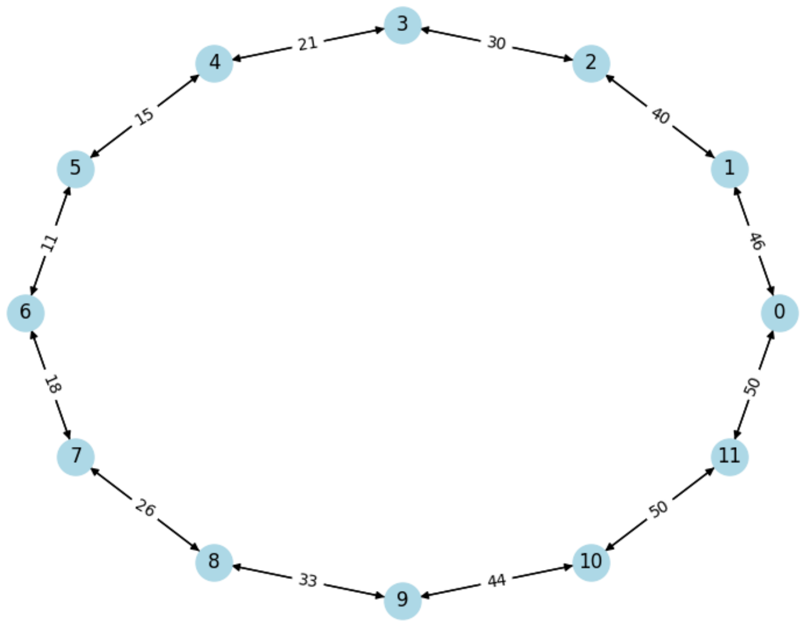


Fig. 5: A sample simulation of a ring topology with node 0 as the C2.

An interesting classical case is the barbell graph, which consists of two completely connected subgraphs (cliques) linked by a series of nodes. As shown in Figure B5, the performance of the graph is improved when the C2 is one of the nodes on the path connecting the two cliques and is maximized when the C2 is at the center of the path.

Indeed, if the C2 is in one of the cliques, then it becomes necessary to traverse the full "connecting" path for transmissions to go from one clique to another.

Randomized graphs

While these relatively uniform graphs can reveal certain patterns that increase performance, it is also important to consider the nonuniform cases. Not all network topologies can be symmetric, and certainly not all network topologies can have their sensor nodes within the direct transmission range of the C2. To further improve our analysis, we looked at various random graphs available through NetworkX and ran simulations similar to the classical cases.

We found that the performance of the network is maximized when the C2 chosen has the greatest number of nodes within a limited depth relative to other possible nodes. Such a C2 is less likely to have a chokepoint nearby that affects many upstream nodes; it is also simultaneously less likely to have an extremely distant node that places a high burden on the network. The randomized powerlaw tree depicted in Figure A3 has several features in its distributions that demonstrate this:

- Simulations in which the leaf node was the C2 always have an operational lifetime of 50, as its sole neighbor ends up being a chokepoint through which all forwarded information must pass.
- In contrast, simulations with nodes 4 and 6 as the C2, though extremely distant from other nodes, have a slightly higher average operational lifetime than 50 because they have at least one neighbor that does not need to pass through an upstream chokepoint.
- Simulations with nodes 9, 10, and 11 as the C2 tend to need only two hops for any given transmission to reach the C2. In other words, the vast majority of nodes are no greater than depth 2 relative to the C2 if a breadth-first search were to be performed.

Indeed, the average number of hops per transmission reflects the relative depth of sensor nodes to the C2 in any given graph. This trend is consistent across all graphs; indeed, the Watts-Strogatz graph in Figure A4 depicts a similar trend, in which the centrally-located node 3 has a higher performance than node 4 as a C2. When node 3 is the C2, the burden of forwarding is pushed up to nodes 2 and 4, which are responsible for three and four nodes, respectively. In contrast, when node 4 is the C2, the burden of forwarding is largely pushed onto node 3, which must handle six nodes (some of which are distant).

From these graphs, it becomes evident that low-depth nodes relative to the C2 should be responsible for forwarding a balanced amount of traffic for the network to be most effective. That is, if there are four nodes at depth 1 relative to the C2, each of those nodes should be responsible for handling the forwarding of one-fourth of all nodes in the network for battery life to be optimized.

When we look at randomized graphs where the density of edges is less sparse, this trend continues but is less evident. In the connected caveman graph depicted in Figure A2, nodes

1, 5, and 9 have the worst performance as C2s because their cliques are not completely connected (that is, one edge has been removed); although nodes 1 and 3 are members of the same clique, node 3 is connected to every member of the clique, while upstream traffic from node 0 must pass through an additional member of the clique for the traffic to reach a C2 at node 1. In turn, the distribution of traffic between node 1's immediate neighbors, nodes 2 and 3, is slightly more uneven than for other members of the clique as C2s.

An interesting conclusion from these randomized graphs is that the average depth of a transmitting node – that is, the average number of hops in a transmission – does not necessarily correlate with the operational lifetime of the network. This is best seen in the Barabasi-Albert distributions in Figure A5, where node 2 has the worst operational lifetime despite not having the highest average hops per transmission (which belongs to node 3 as the C2 instead). Indeed, the "balance" of immediate neighbors is much more important than the number of nodes within a certain depth of the C2.

Finally, the same Barabasi-Albert graph allows us to make one final statement about the beneficial quality of a sensor node network. When upstream subgraphs have a generally higher edge density, it is more likely that the work distribution is evenly distributed. For example, consider node 9 as the C2. Although it has the second-highest average number of hops per transmission, indicating that it is relatively distant from many nodes, it still has a relatively high operational lifetime compared to other nodes, none of which are leaves. Node 9 has four neighbors, some of which are connected to the same neighbors and some of which share edges with the same depth 2 nodes relative to node 9. As a result, it is more likely that when traffic comes from distant nodes such as nodes 2, 3, and 4, their traffic will be evenly distributed across the depth 1 nodes through the "balancing layer" formed by the depth 2 nodes.

All of the statements above are consistent with the final graph we discuss here, the connected Erdős-Rényi graph shown in Figure A1:

- The use of a leaf node as the C2, particularly node 1, caps the operational lifetime of the network at 50 transmissions because node 0 must forward 100% of the entire network's traffic.
- Nodes 0, 3, and 4 perform better than other nodes because they are most "centrally" located within the graph, indicated by the lower number of hops per transmission.
- Although node 5 has a relatively low number of hops per transmission, indicating that the average relative depth of the network's nodes is low, its neighbor nodes are imbalanced in terms of responsibility. Nodes 10 and 8 are unlikely to take more traffic than nodes 3 and 4, effectively forcing nodes 3 and 4 to take nearly all of the network's traffic with no buffer of highly connected "balancing nodes" in between. Thus, node 5's performance as a C2 is worse than that of nodes 6 and 7, despite having similar hops per transmission.

Conclusion

We have shown potentially novel contributions for a specialized sensor node network in which certain environmental and technological assumptions can be made. That is, such sensor node networks should optimize for node reachability and "balance", which can be quantified in various ways. These must be weighed against the overhead cost of maintaining "balancing layers" of nodes through a high density of edges. Other beneficial graph qualities, such as the avoidance of chokepoints near C2s and the reduction of relative node depths to the C2, help improve the operational lifetime of these networks to a smaller degree.

Many of these findings are analogous to the design of modern networks. Indeed, most popular websites today place their services behind load balancers, in which traffic from many different sources is forwarded to a set of nodes responsible for distributing the workload evenly to the actual servers. Similarly, our findings of strongly connected subgraphs improving the efficiency of our networks are consistent with the hierarchical nature of routing across autonomous systems.

There is still considerable future work to be done in exploring the idea of a mesh sensor node network, especially in unfamiliar extraterrestrial conditions. As mentioned in our assumptions, we assume that the mesh network still has a "central" node capable of supporting some of the network's overhead, namely maintaining global state. This makes Dijkstra's algorithm valid in this specific scenario, but what if the mesh network must be completely decentralized? This raises several questions about how much information a dynamic routing system in a battery-constrained mesh network needs and how such a network should be structured.

In such a case, certain dynamic routing systems would not be suitable as-is. These algorithms require knowledge of global state, which is expensive to maintain in a constrained network where each node can only make a finite number of transmissions before dying. Does this mean that partial or imperfect global state is acceptable – that is, can we approximate the battery life of other nodes and still get comparable results to our ideal global state algorithm? What if we decrease the scope of the accurate state we maintain such that we only need to forward battery information to nearby neighbors?

Addressing our other assumptions in a more decentralized system:

- What happens if the links are fallible? Then, it becomes even more expensive to maintain an accurate global state, even if incomplete. In faulty systems, is it worth it to send acknowledgments of battery life information? In more reliable systems, is it sufficient to simply assume that calculated downstream battery lives are accurate?
- What if the initial global state is not known by the entire system, and what if extra nodes are added throughout the lifetime of the system? Is it efficient to do global floods of the state of new nodes, or is it acceptable to only maintain partial knowledge of the network at each node?

As with many other analyses in computer science, the best topology and protocol for maintaining a robust network will depend on the actual needs of the network. This can entail creating new algorithms, establishing dynamic parameters, and making other design decisions that best fit a specific scenario.

Appendix

Randomized graphs

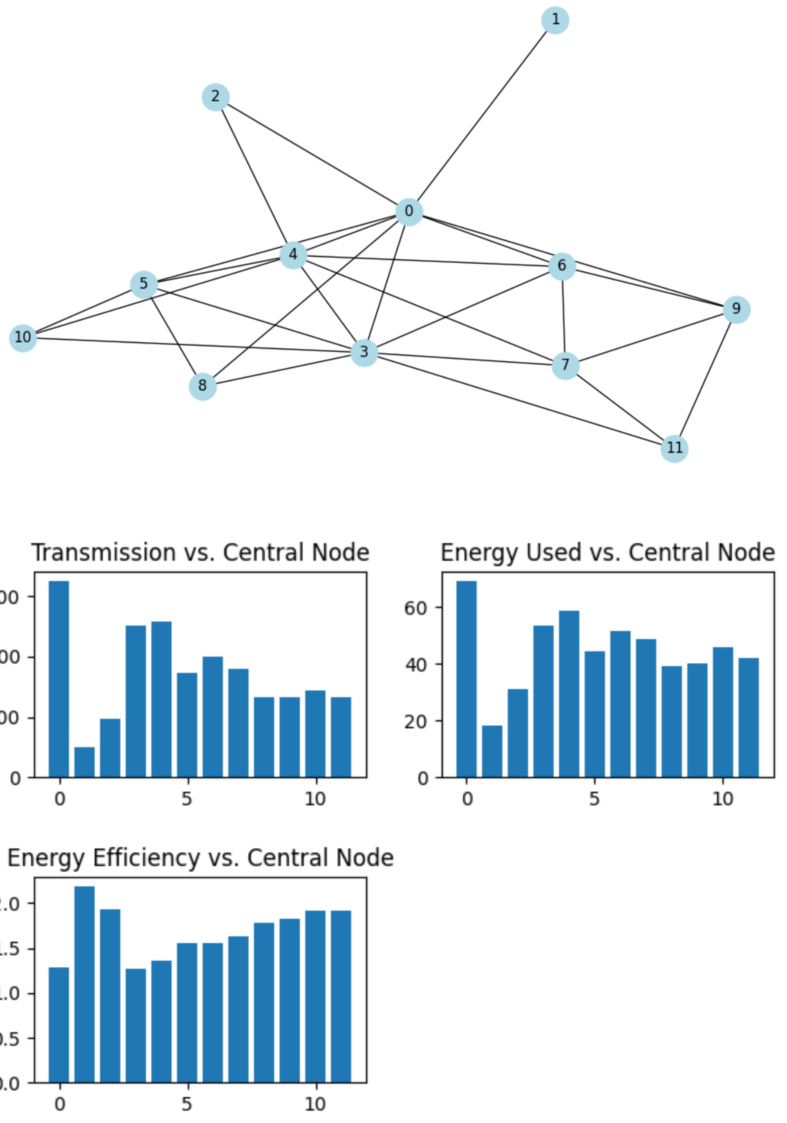


Fig. A1: The connected Erdős-Rényi graph used for simulations (top); the distributions for this graph of various metrics in Table 1 (bottom).

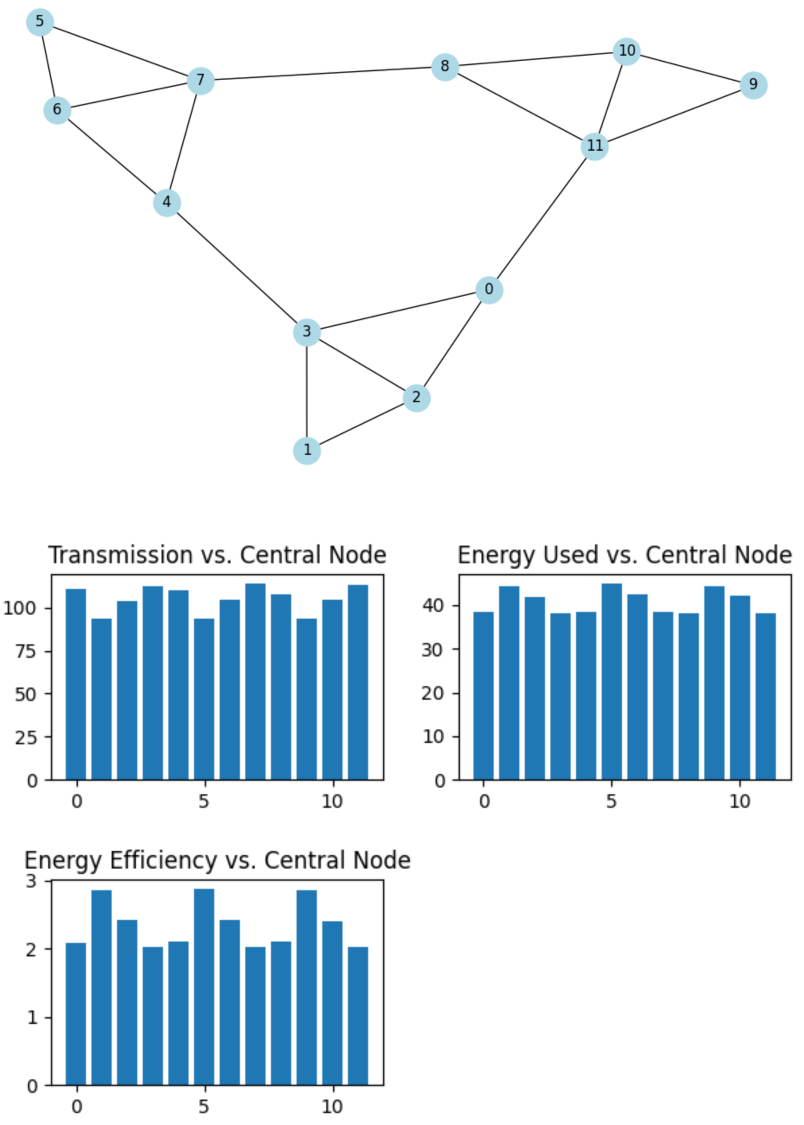


Fig. A2: The connected caveman graph used for simulations (top); the distributions for this graph of various metrics in Table 1 (bottom).

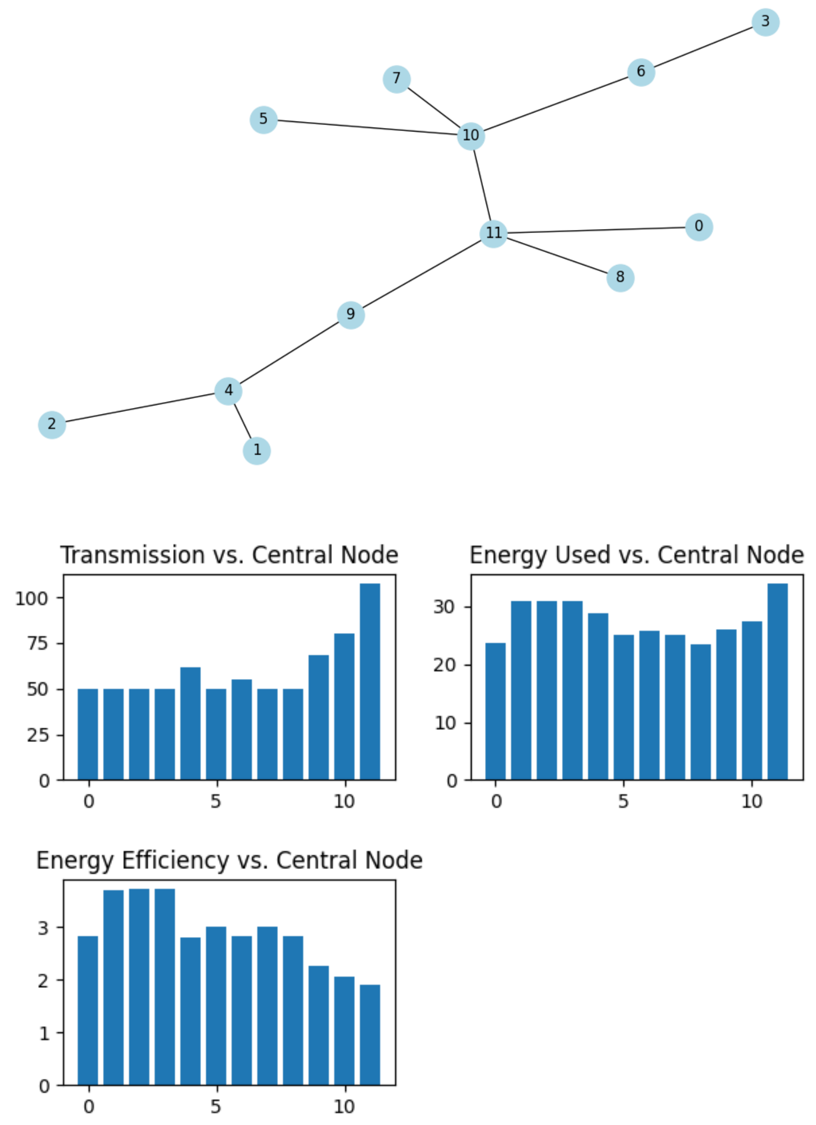
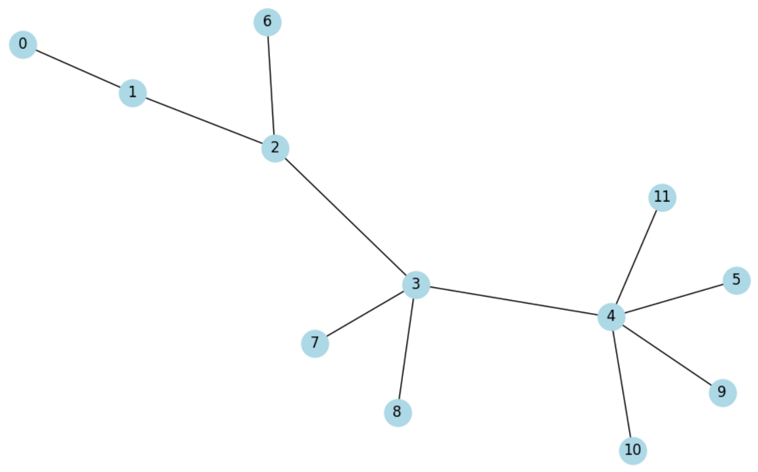


Fig. A3: The randomized powerlaw tree used for simulations (top); the distributions for this graph of various metrics in Table 1 (bottom).



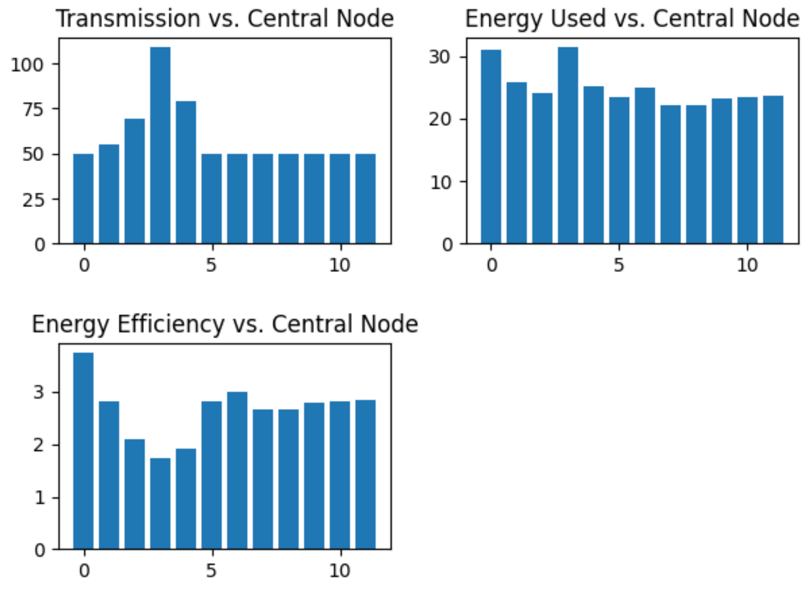
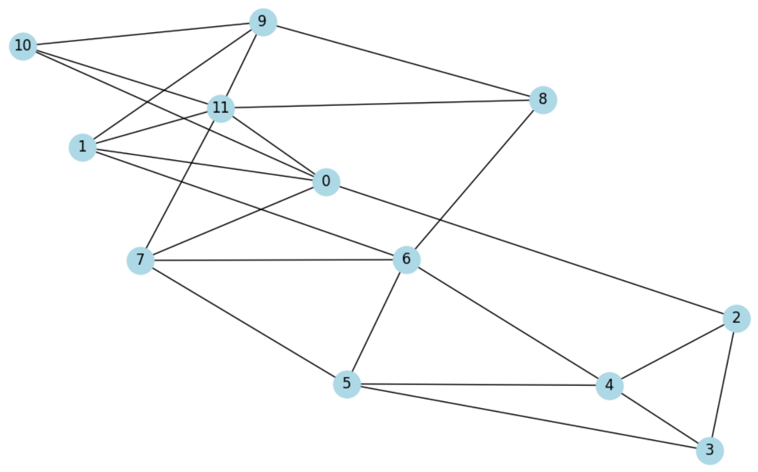


Fig. A4: The Watts-Strogatz graph used for simulations (top); the distributions for this graph of various metrics in Table 1 (bottom).



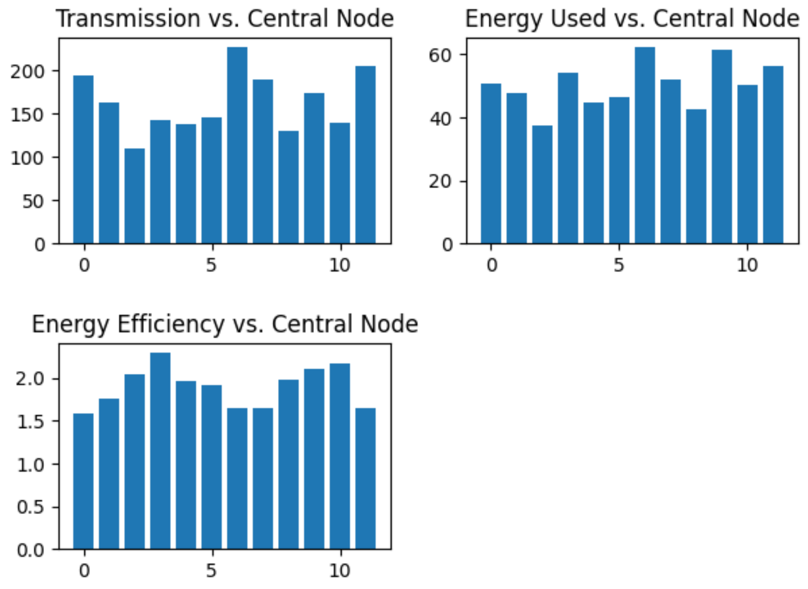
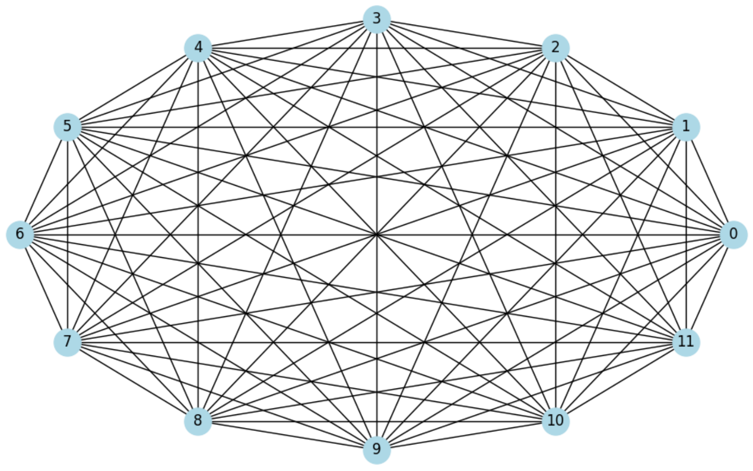


Fig. A5: The Barabasi-Albert used for simulations (top); the distributions for this graph of various metrics in Table 1 (bottom).

Classical graphs



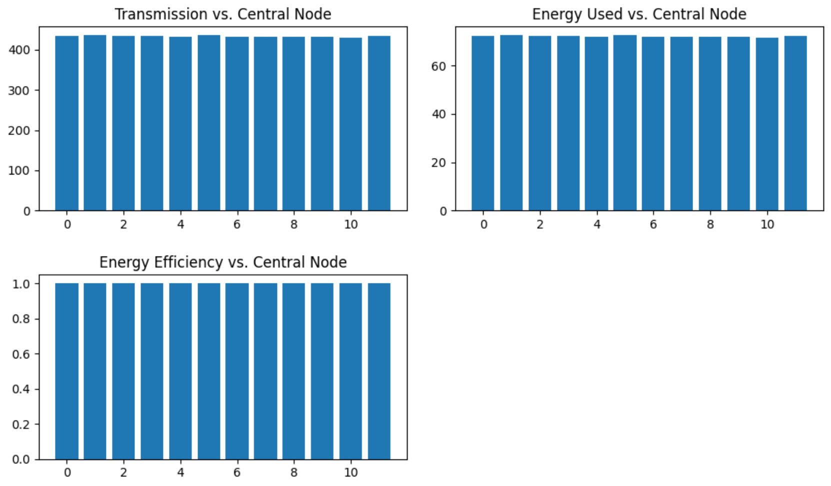
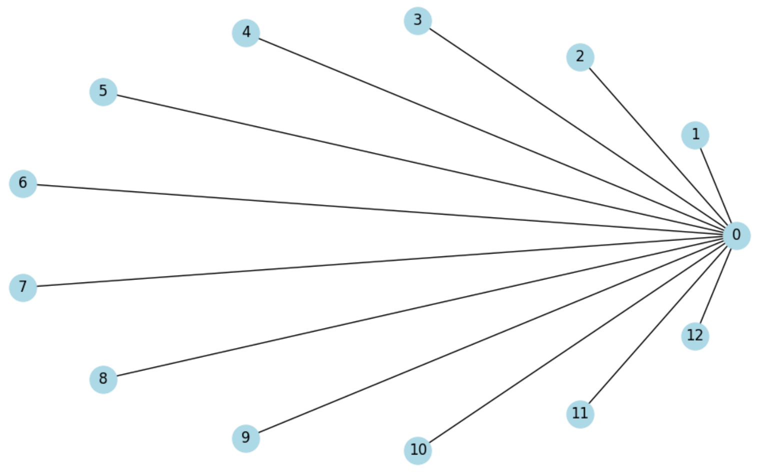


Fig. B1: The mesh graph used for simulations (top); the distributions for this graph of various metrics in Table 1 (bottom).



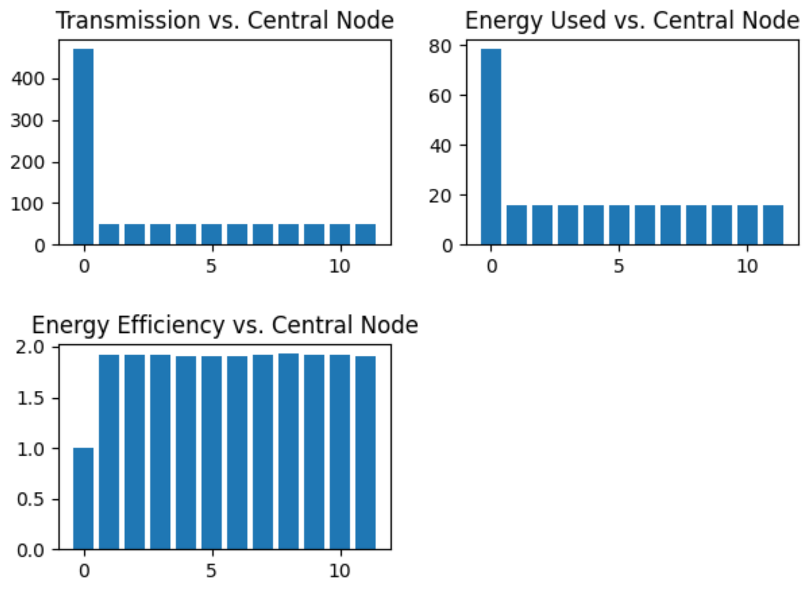


Fig. B2: The star graph used for simulations (top); the distributions for this graph of various metrics in Table 1 (bottom).

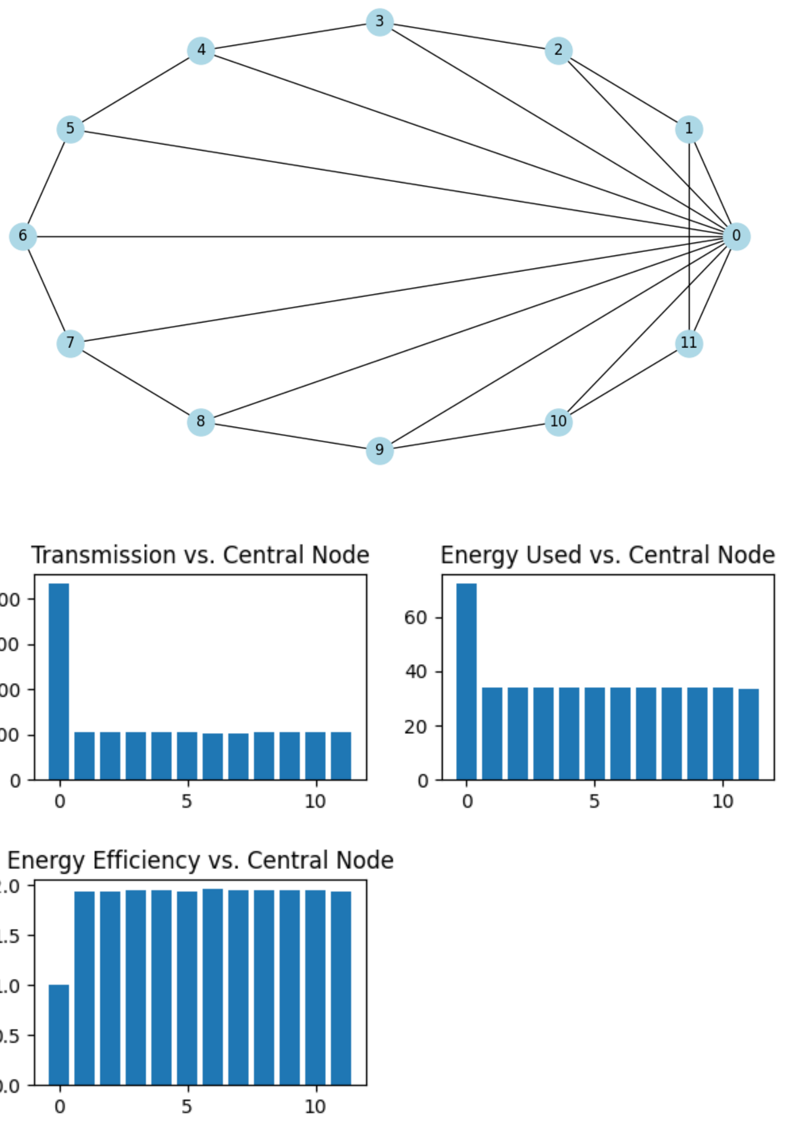
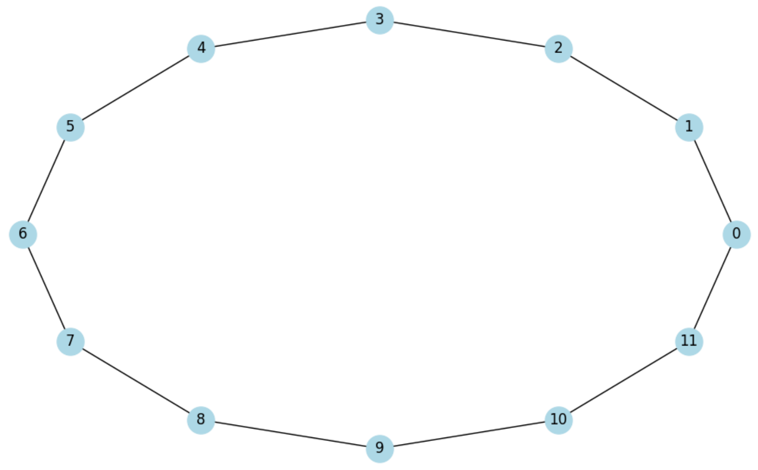


Fig. B3: The wheel graph used for simulations (top); the distributions for this graph of various metrics in Table 1 (bottom).



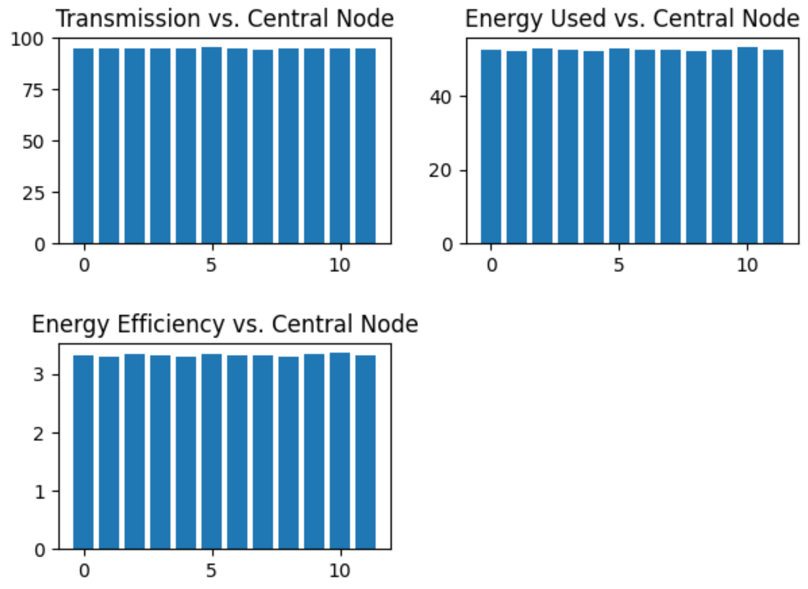


Fig. B4: The ring graph used for simulations (top); the distributions for this graph of various metrics in Table 1 (bottom).

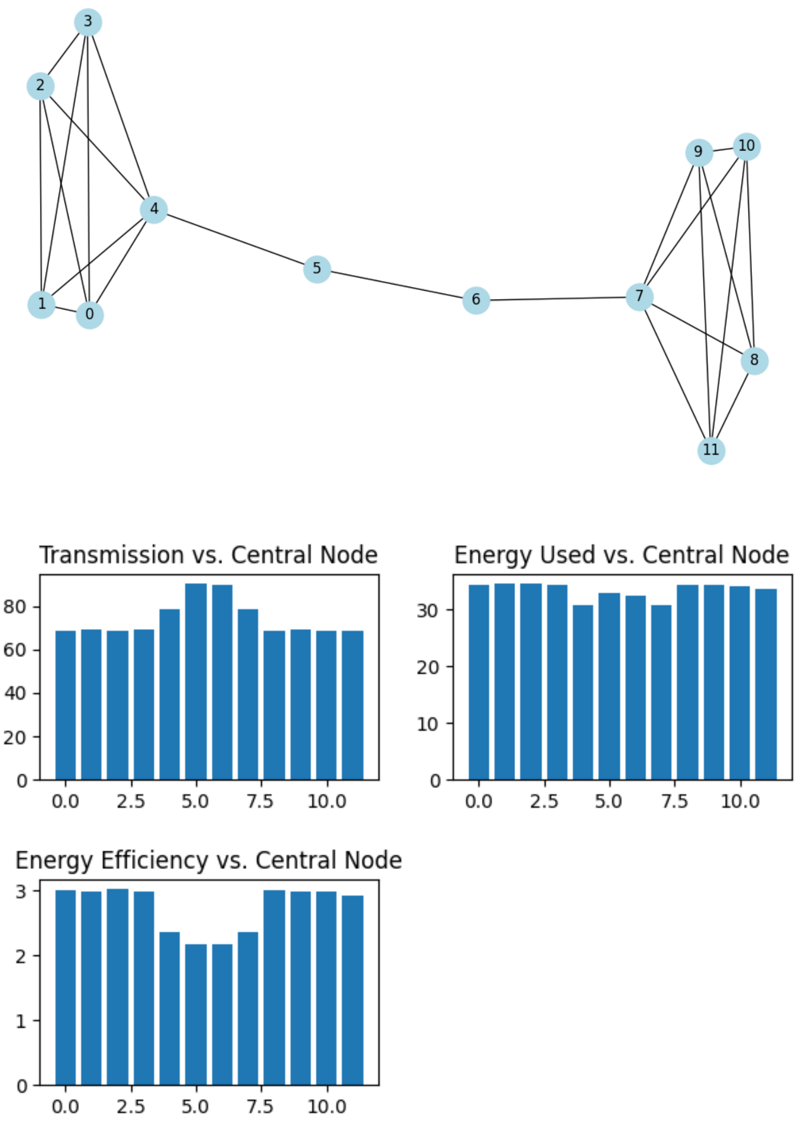


Fig. B5: The barbell graph used for simulations (top); the distributions for this graph of various metrics in Table 1 (bottom).