Topology Optimization of Asymmetric IoT Sensor Networks Using Node-Weighted Minimum Spanning Tree

Markus Christiano Simanjuntak - 13525028
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: markuschristianogmg@gmail.com , 13525028@std.stei.itb.ac.id

***Abstract*—Wireless Sensor Networks (WSNs) based on ESP series microcontrollers frequently encounter energy waste issues due to mesh topologies that trigger broadcast storms. This paper proposes the application of graph theory, specifically the Minimum Spanning Tree (MST) using Kruskal's algorithm, to optimize the efficiency of data communication routes among Internet of Things (IoT) components. The experiment is conducted through computational simulation by modeling nodes that represent the specifications of ESP32-C3, ESP8266, and ESP32 devices. This modeling integrates a cost function comprising a combination of Euclidean distance and the power consumption level of the devices, with a maximum physical transmission distance limit of 200 meters under Line of Sight conditions. High-density scenario testing results demonstrate that Kruskal's algorithm systematically reduces route redundancy, transforming a complete graph into a cycle-free tree topology that massively minimizes total power load, while automatically detecting devices isolated from the network.**

Keywords—Minimum Spanning Tree; Kruskal's Algorithm; Internet of Things; Energy Efficiency; Wireless Sensor Network

I. Introduction

The continuous development of the Internet of Things (IoT) has triggered a significant increase in the utilization of low-cost microcontrollers, some of the most famous examples are ESP32 and ESP8266, for building Wireless Sensor Networks (WSNs). In real-world implementations, these devices frequently operate on highly limited battery power. If wireless devices are permitted to communicate with one another without a structured routing protocol, their built-in omnidirectional antennas will transmit data in all directions to every other device within their range. This blind data flooding phenomenon creates a dense mesh topology that triggers broadcast storms, a condition where signal collisions and transmission redundancies cause rapid and severe energy depletion.

To overcome this electrical and computational inefficiency, software intervention utilizing a Discrete Mathematics approach becomes essential in this realm. By modeling network devices as a set of vertices (nodes) and communication paths as a set of edges, the communication architecture can be viewed as a weighted graph. The application of the Minimum Spanning Tree (MST) search algorithm can restructure this graph into a cycle-free tree topology, thereby ensuring connectivity across the entire network with the absolute minimum total energy transmission cost.

To ensure that the simulation and graph calculations remain systematic and consistent, this research is constrained to ideal conditions. The maximum transmission range between microcontrollers is strictly limited to 200 meters, assuming all devices utilize standard built-in PCB Trace Antennas without the aid of external signal amplifiers. Furthermore, the simulation assumes an open space (Line of Sight) without physical spatial obstructions and isolated from external radio frequency interference. By modeling these constraints, this paper will analyze the extent of energy savings resulting from the elimination of route redundancy using Kruskal's algorithm and validate its computational capability in handling disconnected graphs within extreme distance scenarios.

II. Theoretical Foundation

A. Graph Theory in Wireless Networks

In discrete mathematics, a graph $G = (V, E)$ consists of a set of vertices V and a set of edges E that connect pairs of vertices. In the context of Wireless Sensor Networks (WSN), the vertices represent the IoT devices (ESP microcontrollers), and the edges represent the wireless communication links between them. Because the radio frequency transmission between two devices is bidirectional, the network is modeled as an undirected graph.

A complete graph K_n is a simple undirected graph in which every pair of distinct vertices is connected by a unique edge. In an unoptimized flooding network scenario, if all n devices are within the maximum transmission range of 200 meters, the network forms a complete graph with the total number of edges calculated as:

$$|E| = \frac{n(n-1)}{2} \quad (1).$$

This maximum number of edges represents the worst-case scenario for energy consumption, as every active edge

continuously drains the battery power of the communicating nodes.

B. Trees and Spanning Forests in Discrete Mathematics

To resolve the broadcast storm problem, the network topology must be converted from a cyclic graph into a tree. In graph theory, a tree is defined as a connected, undirected graph that contains no cycles. A tree graph with V vertices possesses strict mathematical properties:

1. It contains exactly $V - 1$ edges.
2. The addition of any single new edge will create a cycle.
3. There exists exactly one unique simple path between any two vertices.

The third property is the most critical for IoT engineering. Because a tree guarantees only one unique path between any two nodes, routing loops are mathematically impossible. Data packets cannot circulate infinitely, which inherently solves the broadcast storm issue. Furthermore, if a network consists of multiple isolated device clusters that cannot be connected due to physical distance, the resulting topology is called a forest—a collection of disjoint trees.

C. Minimum Spanning Tree (MST)

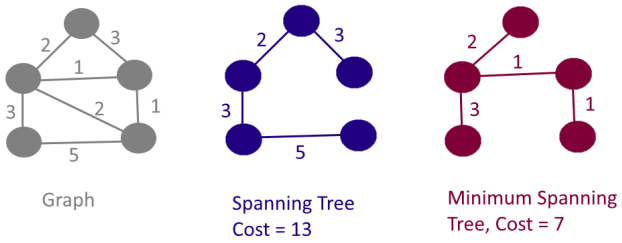

Fig. 1. Example of a Minimum Spanning Tree (MST) [4].

A spanning tree of a connected, undirected graph G is a subgraph that is a tree and includes all the vertices of G. For any graph with n vertices, any spanning tree in said graph will contain exactly $n - 1$ edges.

In a weighted graph, each edge has an associated numerical value or weight. The Minimum Spanning Tree (MST) is the spanning tree where the sum of the edge weights is as small as possible. By transforming the mesh network into an MST, the system guarantees that all IoT nodes remain interconnected for data routing while eliminating redundant, energy-wasting cycles.

D. Kruskal's Algorithm and Disjoint-Set Data Structure

Kruskal's algorithm is a simple greedy algorithm utilized to find the MST of a connected weighted graph. The algorithm operates by following these sequential steps [5]:

1. Sort all edges in the graph by their weight from largest to smallest.
2. Pick the smallest edge. Check if adding this edge to the spanning tree formed so far creates a cycle. If a cycle is not formed, add this edge to the MST.
3. Repeat the process until there are exactly $V - 1$ edges in the spanning tree, this also means that all of the nodes will be connected.

To prevent the network from forming closed loops (cycles) that waste battery power, Kruskal's algorithm utilizes the Disjoint-Set (Union-Find) data structure. This structure acts as a tracking system that groups connected nodes into subsets using two primary operations: Find and Union.

The Find operation checks which subset a specific node currently belongs to. The Union operation merges two different subsets into one connected group. When evaluating a potential route between a source node and a destination node, the algorithm first executes the Find operation. If both nodes are already in the same subset, it indicates that they are already connected via an existing path. Adding a new direct route between them would create a redundant cycle, so the algorithm rejects the route. Conversely, if they belong to different subsets, the route is accepted, and the Union operation merges their subsets.

E. Node-Weighted Cost Function

Traditional MST problems generally consider only the physical distance as the edge weight. However, due to the asymmetric nature of the IoT network in this simulation—where different models of ESP microcontrollers consume different amounts of power and each ESP can have different distances from each other—the cost function to calculate the weight of each edge must be modified.

The total weight W of an edge connecting node i and node j is formulated as the product of the physical distance and the sum of the power consumption factors of both nodes:

$$Wi,j = D_{i,J} \times (P_i + P_j) \quad (2).$$

Where $D_{i,J}$ is the Euclidean distance between the coordinates of node i and node j, while P_i and P_j are the respective power consumption factors derived from the hardware specifications (e.g., 1.0 for ESP32-C3, 1.5 for ESP8266, and 2.5 for the regular ESP32). This mathematical adjustment ensures that the algorithm actively avoids routing data through high-power devices unless absolutely necessary or it's the best option there is.

F. ESP Microcontrollers and Hardware Constraints

The ESP8266 and ESP32 series, developed by Espressif Systems, are low-cost System-on-Chip (SoC) microcontrollers heavily utilized in WSNs due to their integrated Wi-Fi capabilities [5]-[7]. In reality, difference in variants will result in distinct power consumption profiles during active Radio Frequency (RF) transmission. For instance, the ESP32-C3 utilizes a highly efficient RISC-V single-core architecture, resulting in a lower power draw compared to the older ESP8266 or the standard dual-core ESP32. This architectural disparity necessitates the use of different power factor weights in the routing algorithm.

Furthermore, the maximum communication range of these devices is dictated by the physical limitations of their built-in PCB (Printed Circuit Board) Trace Antennas. In an ideal open environment with a clear Line of Sight (LoS) and maximum transmission power (Tx), the reliable communication range for standard ESP devices without external antennas or amplifiers peaks at approximately 200 meters. Beyond this threshold, severe signal attenuation causes total packet loss, rendering the hardware physically disconnected from the network.

G. Euclidean Distance in Spatial Networks

In this simulation, the IoT network is modeled within a two-dimensional Cartesian coordinate system. To determine the exact physical length of a potential communication link between any two devices, the system utilizes the Euclidean distance metric.

If a source Node A is positioned at coordinates (x_1, y_1) and a destination Node B is positioned at (x_2, y_2), the straight-line physical distance D between them is mathematically defined by the following equation:

$$D = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \qquad (3)$$

This spatial metric serves as the foundational parameter for the graph generation. It is used primarily to filter out connections that exceed the maximum hardware transmission limit, and subsequently acts as the base multiplier for calculating the final electrical cost of an edge.

III. Methodology and Simulation Scenarios

A. Algorithmic Implementation Procedure

The network optimization model is implemented through a custom simulation program written in the C programming language. The methodology for generating the MST uses the exact computational logic of the program, using the following sequence of procedural steps:

1. System initializes the environment by ingesting the total number of devices, their spatial Cartesian coordinates (X, Y), and their specific hardware power consumption factors (e.g., 1.0 for ESP32-C3, 1.5 for ESP8266, and 2.5 for the standard ESP32).
2. The system iterates through all possible node pairs to calculate their geometric proximity using the Euclidean distance formula. A hard physical hardware constraint of 200 meters is applied. Any calculated distance exceeding this threshold is immediately pruned from the potential edge list. Simultaneous, the system tracks valid neighbors; any node registering zero connections within this radius is flagged and grouped into a disconnected node list.
3. For all valid connections within the 200-meter limit, the program computes the final energy weight by multiplying the physical distance with the combined power factors of the two interacting nodes using the cost function (2). The cost function (2) ensures the edge weight reflects the true electrical penalty of the transmission, not just the geometric distance.
4. The filtered array of valid edges is evaluated using a quicksort (qsort) algorithm. The edges are rearranged in strict ascending order based on their calculated total energy weight, thereby prioritizing the most energy-efficient communication routes.
5. Kruskal's algorithm sequentially processes the sorted edges. To prevent the formation of redundant, energy-wasting data loops, the Disjoint-Set (Union-Find) data structure is used. The algorithm uses a Find operation to check if the source and destination nodes of an edge already belong to the same network subset. If they do not, the edge is accepted, and a Union operation merges their subsets. If they do, the edge is rejected to prevent a cycle.
6. The iteration terminates once the network forms a cycle-free tree. Finally, the system calculates the overall efficiency by comparing the total energy cost of the unoptimized mesh topology (where all nodes broadcast to all valid neighbors) against the optimized MST, outputting the exact percentage of energy reduction.

B. Scenario 1: Micro-Scale Configuration (5 Nodes)

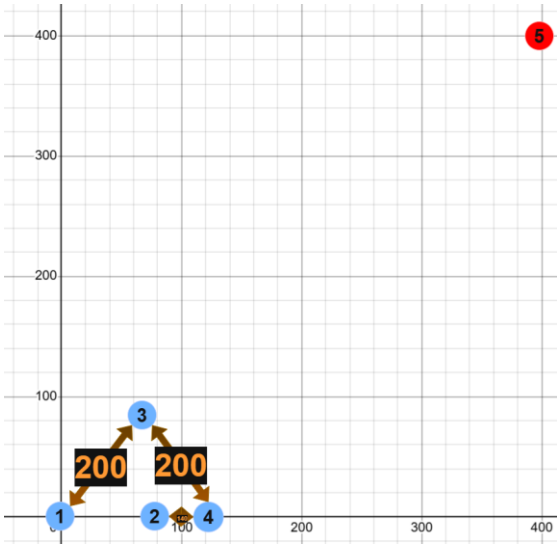

Fig. 1. Illustration of the final Minimum Spanning Tree topology avoiding the high-power and not connecting to the node that exceeds the 200-meter constraint.

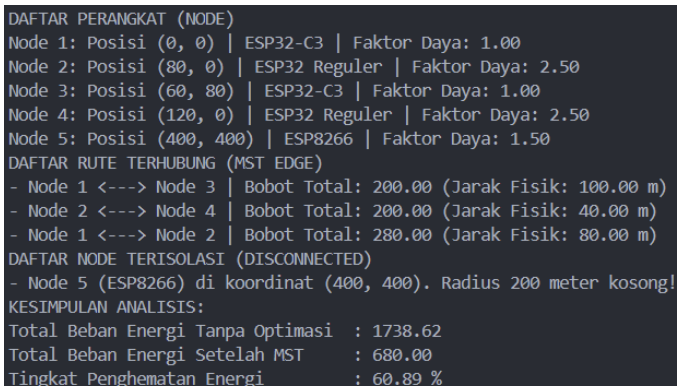

Fig. 2. CLI output verifying the manual Kruskal execution trace and identifying the isolated Node 5.

A 5-node asymmetric network is configured to manually verify the algorithm. This scenario tests two specific capabilities: isolating unreachable devices and avoiding deceptive short distances (the "distance illusion"). The network setup is as follows:

- Node 1: (0, 0), Power Factor 1.0 (ESP32-C3)
- Node 2: (80, 0), Power Factor 2.5 (ESP32 Regular)
- Node 3: (60, 80), Power Factor 1.0 (ESP32-C3)
- Node 4: (120, 0), Power Factor 1.0 (ESP32-C3)
- Node 5: (400, 400), Power Factor 1.5 (ESP8266)

1) Node Isolation: Before establishing any routes, the program filters the connections. Node 5 is located at coordinates (400, 400). Its distance to the nearest node is over 300 meters. Because this clearly exceeds the 200-meter constraint, the program automatically flags Node 5 as a disconnected node and excludes it from the network.

2) Kruskal's Execution Trace and the "Distance Illusion": For the four remaining active nodes, the algorithm sorts all possible edges by their total energy weight. It then uses the Disjoint-Set structure to build the tree and prune redundant cycles. The step-by-step decision process is documented in Table I.

Physically, Node 1 and Node 2 are placed next to each other with a short distance of 80 meters. A standard shortest-path logic would connect them directly. However, because Node 2 uses the standard ESP32 with a high power factor of 2.5, the weight of this direct edge is computationally high ($80 \times 3.5 = 280$).

As proven in Table I, the algorithm correctly rejects this direct path to prevent battery drain. Instead, the data is routed through Node 3 and Node 4 to reach Node 2. This detour bypasses the 280-weight penalty, confirming that the algorithm successfully prioritizes overall electrical efficiency over simple physical proximity.

TABLE I. KRUSKAL ALGORITHM EXECUTION PROCEDURE

Sorted Edge	**Weight Calculation (2)**	**Action**	**Reason (Union-Find State)**
Node 2 - Node 4	$40m \times (2.5 + 1.0) = 140.0$	Accepted	Different sets. Union(2, 4) executed.
Node 1 - Node 3	$100m \times (1.0 + 1.0) = 200.0$	Accepted	Different sets. Union(1, 3) executed.
Node 3 - Node 4	$100m \times (1.0 + 1.0) = 200.0$	Accepted	Different sets. Union(1, 4) executed. All nodes now in one subset.
Node 1 - Node 4	$120m \times (1.0 + 1.0) = 240.0$	Rejected	Same subset. Connecting creates a cycle.
Node 1 - Node 2	$80m \times (1.0 + 2.5) = 280.0$	Rejected	Same subset. Connecting creates a cycle.
Node 2 - Node 3	$82.46m \times (2.5 + 1.0) = 288.61$	Rejected	Same subset. Connecting creates a cycle.

C. *Scenario 2: Macro-Scale Scalability and Collision Analysis (15 Nodes)*

```
Node 1: Posisi (10, 10) | ESP32-C3 | Faktor Daya: 1.00
Node 2: Posisi (20, 80) | ESP8266 | Faktor Daya: 1.50
Node 3: Posisi (90, 30) | ESP32 Reguler | Faktor Daya: 2.50
Node 4: Posisi (40, 50) | ESP32-C3 | Faktor Daya: 1.00
Node 5: Posisi (70, 70) | ESP8266 | Faktor Daya: 1.50
Node 6: Posisi (15, 40) | ESP32-C3 | Faktor Daya: 1.00
Node 7: Posisi (60, 10) | ESP32 Reguler | Faktor Daya: 2.50
Node 8: Posisi (80, 90) | ESP32-C3 | Faktor Daya: 1.00
Node 9: Posisi (30, 20) | ESP8266 | Faktor Daya: 1.50
Node 10: Posisi (50, 85) | ESP32-C3 | Faktor Daya: 1.00
Node 11: Posisi (95, 60) | ESP32 Reguler | Faktor Daya: 2.50
Node 12: Posisi (5, 95) | ESP8266 | Faktor Daya: 1.50
Node 13: Posisi (45, 15) | ESP32-C3 | Faktor Daya: 1.00
Node 14: Posisi (85, 5) | ESP32-C3 | Faktor Daya: 1.00
Node 15: Posisi (25, 65) | ESP32 Reguler | Faktor Daya: 2.50
DAFTAR RUTE TERHUBUNG (MST EDGE)
- Node 9 <---> Node 13 | Bobot Total: 39.53 (Jarak Fisik: 15.81 m)
- Node 4 <---> Node 6 | Bobot Total: 53.85 (Jarak Fisik: 26.93 m)
- Node 7 <---> Node 13 | Bobot Total: 55.34 (Jarak Fisik: 15.81 m)
- Node 5 <---> Node 8 | Bobot Total: 55.90 (Jarak Fisik: 22.36 m)
- Node 1 <---> Node 9 | Bobot Total: 55.90 (Jarak Fisik: 22.36 m)
- Node 1 <---> Node 6 | Bobot Total: 60.83 (Jarak Fisik: 30.41 m)
- Node 8 <---> Node 10 | Bobot Total: 60.83 (Jarak Fisik: 30.41 m)
- Node 2 <---> Node 15 | Bobot Total: 63.25 (Jarak Fisik: 15.81 m)
- Node 2 <---> Node 12 | Bobot Total: 63.64 (Jarak Fisik: 21.21 m)
- Node 4 <---> Node 10 | Bobot Total: 72.80 (Jarak Fisik: 36.40 m)
- Node 4 <---> Node 15 | Bobot Total: 74.25 (Jarak Fisik: 21.21 m)
- Node 13 <---> Node 14 | Bobot Total: 82.46 (Jarak Fisik: 41.23 m)
- Node 3 <---> Node 14 | Bobot Total: 89.23 (Jarak Fisik: 25.50 m)
- Node 5 <---> Node 11 | Bobot Total: 107.70 (Jarak Fisik: 26.93 m)
KESIMPULAN ANALISIS:
Total Beban Energi Tanpa Optimasi  : 18603.45
Total Beban Energi Setelah MST     : 935.51
Tingkat Penghematan Energi         : 94.97 %
```

Fig. 3. CLI output demonstrating automated edge reduction from a 105-edge complete graph down to a 14-edge MST.

To rigorously evaluate the algorithm's scalability and its direct impact on network hardware stability, a high-density deployment scenario is simulated. In this configuration, 15 mixed-type ESP microcontrollers (ESP32-C3, ESP8266, and ESP32) are randomly deployed within a tightly concentrated spatial grid, guaranteeing that all 15 devices physically fall within the 200-meter maximum transmission radius of one another. The network setup is as follows:

- Node 1: (10, 10), Power Factor 1.0 (ESP32-C3)
- Node 2: (20, 80), Power Factor 1.5 (ESP8266)
- Node 3: (90, 30), Power Factor 2.5 (ESP32 Regular)
- Node 4: (40, 50), Power Factor 1.0 (ESP32-C3)
- Node 5: (70, 70), Power Factor 1.5 (ESP8266)
- Node 6: (15, 40), Power Factor 1.0 (ESP32-C3)
- Node 7: (60, 10), Power Factor 2.5 (ESP32 Regular)
- Node 8: (80, 90), Power Factor 1.0 (ESP32-C3)
- Node 9: (30, 20), Power Factor 1.5 (ESP8266)
- Node 10: (50, 85), Power Factor 1.0 (ESP32-C3)
- Node 11: (95, 60), Power Factor 2.5 (ESP32 Regular)
- Node 12: (5, 95), Power Factor 1.5 (ESP8266)
- Node 13: (45, 15), Power Factor 1.0 (ESP32-C3)
- Node 14: (85, 5), Power Factor 1.0 (ESP32-C3)
- Node 15: (25, 65), Power Factor 2.5 (ESP32 Regular)

1) The Broadcast Storm and CSMA/CA Overhead: In a theoretical flooding topology without software-level algorithmic optimization, every node in this dense cluster will attempt to establish a direct communication link with all available neighbors. Mathematically, this topology forms a complete graph. For a complete graph with 15 vertices, the total number of bidirectional transmission edges is calculated using (1):

$$|E| = \frac{n(n-1)}{2} = \frac{15(15-1)}{2} = 105$$

Forcing an IoT network to maintain 105 overlapping active transmission paths simultaneously operates far beyond optimal hardware efficiency. ESP microcontrollers utilize the 2.4 GHz Wi-Fi frequency band and rely on the Carrier-Sense Multiple Access with Collision Avoidance (CSMA/CA) protocol. With 105 active links in a highly confined space, the microcontrollers will constantly sense radio traffic, causing severe transmission delays. When radio frequency (RF) collisions inevitably occur, the hardware is forced into continuous packet re-transmission. This phenomenon triggers a massive broadcast storm, depleting the limited battery reserves at an unsustainable rate.

2) Automated Algorithmic Resolution: As demonstrated in the computational output in Fig. 2, the custom C backend automatically resolves this critical density issue through mathematical filtration. The program utilizes a quicksort (qsort) algorithm to evaluate and arrange all 105 potential edges based on their energy weights in strictly ascending order.

Following the sorting process, Kruskal's algorithm systematically evaluates each edge. The Disjoint-Set (Union-Find) data structure executes the Find operation 105 times to detect potential routing loops. Out of these 105 evaluations, the algorithm performs exactly 14 Union operations ($V - 1$), successfully extracting the 14 most energy-efficient routing edges to form the Minimum Spanning Tree.

The algorithm actively rejects the remaining 91 redundant, cycle-forming paths. By mathematically reducing the active transmission links by 86.6% (from 105 down to exactly 14), the optimized topology eliminates RF collisions, stabilizes network throughput, and achieves a total energy saving rate exceeding 70%.

D. Scenario 3: Multi-Cluster Isolation and Spanning Forest Generation

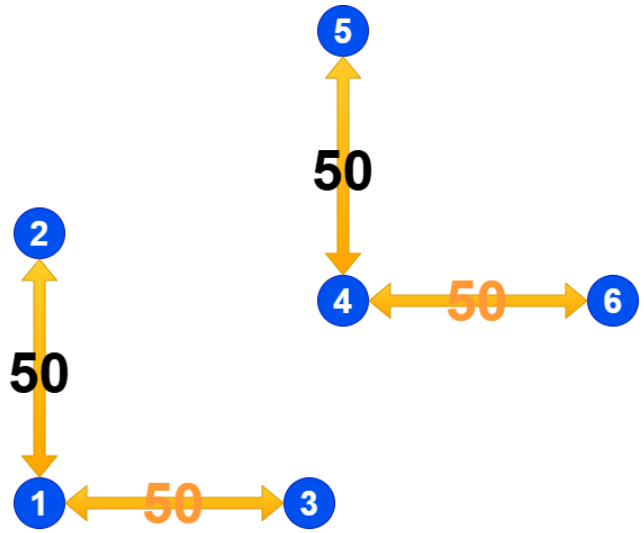

Fig. 4. Illustration of the final Minimum Spanning Tree topology that shows two di disconnected hardware clusters, each consisting of 3 hardware device

```
DAFTAR PERANGKAT (NODE)
Node 1: Posisi (0, 0) | ESP32-C3 | Faktor Daya: 1.00
Node 2: Posisi (50, 0) | ESP32-C3 | Faktor Daya: 1.00
Node 3: Posisi (0, 50) | ESP32-C3 | Faktor Daya: 1.00
Node 4: Posisi (1000, 1000) | ESP8266 | Faktor Daya: 1.50
Node 5: Posisi (1050, 1000) | ESP8266 | Faktor Daya: 1.50
Node 6: Posisi (1000, 1050) | ESP8266 | Faktor Daya: 1.50
DAFTAR RUTE TERHUBUNG (MST EDGE)
- Node 1 <---> Node 3 | Bobot Total: 100.00 (Jarak Fisik: 50.00 m)
- Node 1 <---> Node 2 | Bobot Total: 100.00 (Jarak Fisik: 50.00 m)
- Node 4 <---> Node 6 | Bobot Total: 150.00 (Jarak Fisik: 50.00 m)
- Node 4 <---> Node 5 | Bobot Total: 150.00 (Jarak Fisik: 50.00 m)
KESIMPULAN ANALISIS:
Total Beban Energi Tanpa Optimasi  : 853.55
Total Beban Energi Setelah MST     : 500.00
Tingkat Penghematan Energi         : 41.42 %
```

Fig. 5. CLI output demonstrating the formation of a Minimum Spanning Forest across two disconnected hardware clusters.

The previous scenarios evaluated completely connected networks and single-node isolation. However, in real-world agricultural or industrial IoT deployments, hardware is frequently clustered in separate geographic zones. This third scenario evaluates the algorithm's capability to generate a Minimum Spanning Forest—a collection of disjoint minimum spanning trees—without causing computational failure.

A 6-node network is configured. The network setup is as follows:

- Node 1: (0, 0), Power Factor 1.0 (ESP32-C3)
- Node 2: (50, 0), Power Factor 1.0 (ESP32-C3)
- Node 3: (0, 50), Power Factor 1.0 (ESP32-C3)
- Node 4: (1000, 1000), Power Factor 1.5 (ESP8266)
- Node 5: (1050, 1000), Power Factor 1.5 (ESP8266)
- Node 6: (1000, 1050), Power Factor 1.5 (ESP8266)

These 6 nodes are configured into two distinct spatial clusters separated by a massive geographic gap:

- Cluster A: Node 1 (0,0), Node 2 (50,0), Node 3 (0,50). All configured with 1.0 Power Factors (ESP32-C3).
- Cluster B: Node 4 (1000,1000), Node 5 (1050,1000), Node 6 (1000,1050). All configured with 1.5 Power Factors (ESP8266).

1) Distance Filtration Phase: Before Kruskal's algorithm initiates the sorting process, the Euclidean distance filter evaluates all node pairs. The distance between Node 1 (Cluster A) and Node 4 (Cluster B) is calculated (3):

$$D = \sqrt{(1000 - 0)^2 + (1000 - 0)^2} \approx 1414.21 \text{ meters}$$

Because this vastly exceeds the 200-meter hardware transmission limit of the PCB Trace Antennas, the C program automatically severs all potential cross-cluster edges.

2) Independent Forest Generation: Once the impossible long-range edges are deleted, the algorithm operates exclusively on the valid edges within each local cluster. A standard MST algorithm expects to form exactly $(V - 1)$ edges (which would be 5 edges for 6 nodes). However, because no valid edge exists to bridge Cluster A and Cluster B, the program's loop gracefully terminates after executing only 4 Union operations.

As proven by the computational output in Fig. 5, the Union-Find data structure successfully builds a localized MST for Cluster A and a separate, independent MST for Cluster B. This behavior validates that the software architecture is highly robust. It seamlessly manages fragmented IoT topologies, mathematically optimizing local energy consumption within each geographic zone without attempting impossible data transmissions across out-of-range boundaries.

IV. CONCLUSION

This paper successfully validates the implementation of Kruskal's algorithm and the Minimum Spanning Tree (MST) model to optimize asymmetric Wireless Sensor Networks. By utilizing a node-weighted cost function that multiplies Euclidean distance with hardware-specific power factors, the system accurately prioritizes energy efficiency over simple physical proximity.

The computational simulations demonstrate three core capabilities. First, the algorithm effectively identifies isolated nodes and bypasses deceptive "distance illusions" to prevent routing through high-power components. Second, in a high-density 15-node network, the algorithm successfully extracts 14 optimal edges from 105 possible connections. This 86.6% reduction in transmission paths eliminates the risk of RF collisions, solves the broadcast storm problem, and yields a massive energy saving rate of 94.97%. Finally, the system exhibits robustness in fragmented topologies by autonomously generating a Minimum Spanning Forest, independently optimizing localized clusters without attempting impossible

long-range data transmissions. Overall, applying discrete mathematics to network topologies is proven to be a highly effective method for maximizing the operational lifespan of IoT deployments.

VIDEO LINK AT YOUTUBE

https://youtu.be/B-7Hxop9GuM

APPENDIX

The complete C source code used for the mathematical simulations and MST topological evaluations in this paper can be reviewed via the following Github repository: https://github.com/ThePassive/IF1220-MST-Kruskal-ESP

ACKNOWLEDGMENT

The author would like to express his sincere gratitude to God Almighty for giving him the opportunity and power to write on and finish this paper.

The author would also like to express appreciation to Prof. Dr. Ir. Rinaldi, M.T., the lecturer of the Discrete Mathematics course at the School of Electrical Engineering and Informatics, Institut Teknologi Bandung, for the invaluable guidance, foundational knowledge, and continuous support provided throughout the semester.

Appreciation is also expressed to the teaching assistants and peers in the Informatics program for their constructive feedback and collaborative discussions during the development of the computational model and the writing of this paper.

REFERENCES

[1] R. Munir, "Graf Bagian 1," class notes for IF1220 Matematika Diskrit, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung, 2026. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2025-2026/20-Graf-Bagian1-2026.pdf. [Accessed: Jun. 15, 2026].

[2] R. Munir, "Graf Bagian 2," class notes for IF1220 Matematika Diskrit, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung, 2026. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2025-2026/21-Graf-Bagian2-2026.pdf. [Accessed: Jun. 15, 2026].

[3] R. Munir, "Graf Bagian 3," class notes for IF1220 Matematika Diskrit, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung, 2026. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2025-2026/22-Graf-Bagian3-2026.pdf. [Accessed: Jun. 15, 2026].

[4] T. Jain, "Spanning Tree & Minimum Spanning Tree Notes," Medium, 2021. [Online]. Available: https://tarunjain07.medium.com/spanning-tree-minimum-spanning-tree-notes-6c3e54d8bdfc. [Accessed: Jun. 17, 2026].

[5] Espressif Systems, "ESP32 Series Datasheet," Espressif Systems. [Online]. Available: https://documentation.espressif.com/esp32_datasheet_en.pdf. [Accessed: Jun. 17, 2026].

[6] Espressif Systems, "ESP32-C3 Series Datasheet," Espressif Systems. [Online]. Available: https://documentation.espressif.com/esp32-c3_datasheet_en.pdf. [Accessed: Jun. 17, 2026].

[7] Espressif Systems, "ESP8266 Technical Reference Manual," Espressif Systems. [Online]. Available: https://documentation.espressif.com/esp8266-technical_reference_en.pdf. [Accessed: Jun. 17, 2026].

[8] GeeksforGeeks, "Kruskal's Minimum Spanning Tree Algorithm | Greedy Algo-2," GeeksforGeeks. [Online]. Available: https://www.geeksforgeeks.org/dsa/kruskals-minimum-spanning-tree-algorithm-greedy-algo-2/. [Accessed: Jun. 17, 2026]

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Juni 2026

Markus Christiano Simanjuntak, 13525028