Wireless Sensor Networks

Sensor Data Collection and Data Aggregation

by Group 6
Andreas Kaag Thomsen, 202105844
Asger Song Høøck Poulsen, 202106630
Niels Viggo Stark Madsen, 202106096
Phat Pham, 202104613

A Wireless Sensor Networks Mini Project

December 17, 2024

1 Introduction

Wireless Sensor Networks (WSNs) are increasingly used for environmental monitoring, industrial applications, and smart home systems. In this project, we focus on collecting and aggregating sensor data, specifically simulated data, using a multi-hop WSN. The objective is to efficiently transmit relevant data to a central sink node while evaluating the benefits of data aggregation at intermediate nodes on performance metrics such as completeness and latency.

1.1 Project Objectives

The overall objectives of this project are as follows:

- Generate sensor data following a specific distribution and collect across a multi-hop network.
- Send periodic sensor reports
- Aggregate data at intermediate nodes and compare the results with and without aggregation.
- Visualize the data collected at the sink, including real-time visualization.
- Evaluate the performance of the system.

With these objectives in mind, the following sections present the theoretical underpinnings, system architecture, and experimental setup of our approach, culminating in a thorough evaluation of its performance and benefits.

2 Theoretical Foundations

In this section, we introduce the theoretical principles that underlie our WSN project. This includes an overview of multi-hop communication in WSNs, the concepts of data aggregation, probabilistic distributions for simulating sensor readings, and key performance metrics to evaluate the system.

2.1 Multi-hop Wireless Sensor Networks

WSNs are composed of spatially distributed sensor nodes that communicate wirelessly to monitor physical or environmental conditions, such as temperature, humidity, and light. In a multi-hop WSN, sensor nodes transmit data to a sink node (or base station) through multiple intermediate nodes rather than a direct single-hop connection. This structure is especially beneficial in large-scale deployments where direct communication with the sink may require excessive power due to long distances or physical obstacles. This concept is also illustrated in fig. 1.

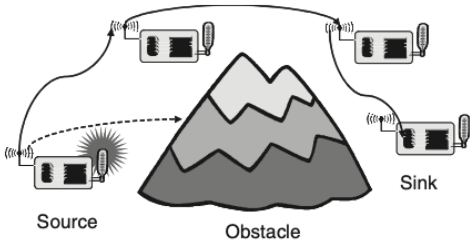


Figure 1: Multi-hop networks: As direct communication is impossible because of distance and/or obstacles, multi-hop communication can circumvent the problem - Taken from [1]

In multi-hop networks, nodes sense data and forward packets from other nodes, extending the network's range and conserving energy across nodes. Multi-hop communication enables network scalability and reliability, allowing data to travel through alternative paths in case of node failures or interruptions.

2.2 Data Aggregation

Data aggregation in WSNs refers to the systematic combination and transformation of raw sensor measurements into more compact, informative representations before they reach the sink node. This process exploits the inherent redundancy present within a WSN, allowing a reduction in the total data volume that needs to be transmitted over energy-constrained links. By minimizing unnecessary communication, data aggregation not only conserves scarce energy resources but also reduces network congestion, and enhances scalability.

A fundamental principle underlying aggregation is the exploitation of both spatial and temporal redundancy inherent in many sensing environments. Spatial redundancy arises because sensors deployed in close proximity often record highly correlated measurements; rather than transmitting a near-duplicate stream of values from each node, intermediate nodes can fuse these data points into a single, statistically representative quantity. Similarly, temporal redundancy exploits the observation that successive measurements at a given node may change slowly or predictably over time. By aggregating these time-correlated values, the network avoids sending every individual sample, focusing instead on capturing trends, patterns, or critical changes when they occur.

From a theoretical point of view, aggregation can be viewed as an information reduction process that carefully balances the fidelity of data against the overhead of communication. The literature [1] on data aggregation provides various models and algorithms, ranging from simple linear combinations (e.g., averaging values) to more complex statistical techniques that operate at intermediate nodes. Such nodes can serve as 'information bottlenecks', where raw data streams converge and are distilled into concise summaries, estimates, or alerts.

In practice, the choice of the aggregation strategy depends on the application requirements and the underlying properties of the detected phenomenon. Some applications prioritize preserving extreme values or event occurrences, while others emphasize stable,

long-term trends. In all cases, by leveraging spatial and temporal redundancy, data aggregation transforms the raw output of many distributed sensors into meaningful, timely, and size-efficient information streams that support informed decision-making at the sink node.

2.3 Probabilistic Distributions for Simulating Sensor Readings

Evaluating the performance of a WSN under realistic yet controllable conditions often requires the careful simulation of sensor data. To achieve this, known probabilistic distributions—such as uniform or Gaussian—are employed to characterize sensor behavior, encapsulating environmental variability and measurement noise in a mathematically tractable manner. Using well-defined distributions, it becomes possible to systematically introduce and manage the complexity of simulated inputs, enabling precise manipulation of parameters and conditions. This level of control allows for a thorough evaluation of data aggregation techniques, including their responses to malfunctions and anomalies in sensor outputs, thereby facilitating a more comprehensive and reliable assessment of system performance.

2.4 Performance Metrics

Assessing the effectiveness of the design and algorithms of a WSN requires carefully defined performance metrics. In this project, we focus on three key metrics—completeness, latency, and accuracy—that capture different aspects of network performance and inform potential trade-offs.

In the following, we present the metrics used to assess the performance of the system. It is implicitly understood that 'sources' are nodes that generate data and that 'aggregators' are the nodes that aggregate the received data. Let:

- $\mathcal{S}$: The set of all the source messages.
- $\mathcal{A}$: The set of all aggregator messages.
- X_i: The sample value of the i -th source message, $i \in \mathcal{S}$.
- t_i: The timestamp of the i-th source message, $i \in \mathcal{S}$.
- msg_id_i: The message ID of the i-th source message, $i \in \mathcal{S}$.
- node_id_i: The node ID of the i-th source message, $i \in \mathcal{S}$.
- $\bar{X}_j$: The aggregated sample value from the j-th aggregator message, $j \in \mathcal{A}$.
- $\mathbf{msg\_ids}_j$: The vector of message IDs in the j-th aggregator message, $j \in \mathcal{A}$.
- $\mathbf{node\_ids}_j$: The vector of node IDs in the j-th aggregator message, $j \in \mathcal{A}$.
- t_j: The timestamp of the j-th aggregator message, $j \in \mathcal{A}$.

In addition, let $\mathcal{S}_{\text{matched}} \subseteq \mathcal{S}$ represent the subset of source messages that are successfully matched to an aggregator message. The number of total messages received can then be defined as:

$$N_{\text{received}} = |\mathcal{S}_{\text{matched}}| \tag{1}$$

Matching Conditions

To keep track of message origins we introduce this definition of message matching. A source message $i \in \mathcal{S}$ is matched to an aggregator message $j \in \mathcal{A}$ if:

$$\exists k \text{ such that } (\text{msg\_id}_i = \mathbf{msg\_ids}_j[k]) \wedge (\text{node\_id}_i = \mathbf{node\_ids}_j[k])$$

where k is the index in the vectors $\mathbf{msg\_ids}_j$ and $\mathbf{node\_ids}_j$. The definition explicitly ensures that the positions of msg_id and node_id are aligned when matching.

Latency

The latency L_i for a matched source message i refers to the time delay between data generation at the source and its successful reception at the sink defined by:

$$L_i = t_j - t_i \tag{2}$$

where j is the corresponding aggregator message that matches i. Lower latency enables more timely decision-making, particularly in applications where rapid responses to environmental changes are critical. This metric helps to evaluate the efficiency of the data aggregation mechanism.

Average Latency

The average latency is the mean latency for all matched source messages:

$$\text{Average Latency} = \frac{1}{N_{\text{received}}} \sum_{i \in \mathcal{S}_{\text{matched}}} L_i \tag{3}$$

Accuracy

The accuracy, A_i for a matched source message i, indicates how closely the aggregated results represent the true underlying sensor values. This is calculated by the absolute difference between the source sample X_i and the aggregated value $\bar{X}_j$:

$$A_i = |X_i - \bar{X}_j| \tag{4}$$

Although data aggregation can reduce communication overhead and energy consumption, it should not degrade accuracy excessively. Balancing the level of aggregation with the need for faithful data representation is key to ensuring the network provides useful insights while still operating efficiently.

Mean Absolute Error

The Mean Absolute Error (MAE) is the mean absolute difference for all matched source messages:

$$\text{MAE} = \frac{1}{N_{\text{received}}} \sum_{i \in \mathcal{S}_{\text{matched}}} A_i \tag{5}$$

A lower MAE indicates that the aggregated values closely approximate the true sensor measurements, reflecting higher accuracy. In contrast, a higher MAE suggests increased deviations between the aggregated and original data, pointing to potential deficiencies in the aggregation strategy or the presence of noisy or malfunctioning nodes.

Completeness

Completeness measures the extent to which the sink node receives all intended data:

$$\text{Completeness} = \frac{N_{\text{received}}}{|\mathcal{S}|} \tag{6}$$

A network with high completeness reliably delivers information from the source nodes to the sink, reflecting the robustness and consistency of the aggregation strategies employed.

By examining completeness, latency, and accuracy together, we gain an overall understanding of the network's performance, allowing us to identify strengths, weaknesses, and potential areas for optimization within the WSN design.

3 System Design

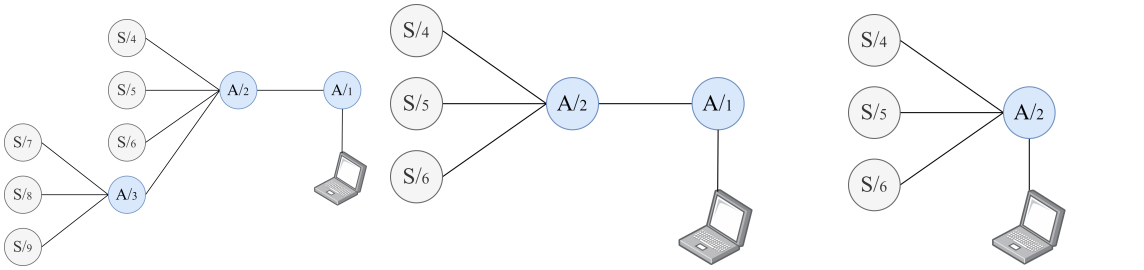
This section outlines the WSN architecture of the project and key design choices. We present the multi-hop network topologies, the generation of sensor data, and the simulation of malfunctioning nodes. Next, we explain how source nodes compute statistical measurements locally to balance the computation across the network. Following this, we detail our data aggregation function, which leverages inverse-variance weighting to enhance robustness, and finally, we introduce our real-time visualization tool for monitoring network performance.

3.1 Network Architecture

The network is structured as a multi-hop topology in which *source* nodes periodically transmit sensor data to an *aggregator* node and the aggregator nodes could forward to another aggregator node. Each node in the network communicates using UDP through the `simple-udp` module in Contiki-NG using a common port and hard-coded IP addresses. These addresses are derived from the uniquely defined `node_id`, which is derived from the node's link address[1]. The use of hard-coded IP addresses allows us to control how each node communicates with the other nodes.

[1] `node-id.h`

Regarding the network topology, we have worked with three different topologies to enable us to analyze the benefits of data aggregation in different scenarios. These three topologies are shown in fig. 2. As seen in these figures, the number of source nodes N_S and aggregator nodes N_A vary in the different topologies.



(a) Topology 1 with $N_S = 6$ source nodes and $N_A = 3$ aggregator nodes.

(b) Topology 2 with $N_S = 3$ source nodes and $N_A = 2$ aggregator nodes.

(c) Topology 3 with $N_S = 3$ source nodes and $N_A = 1$ aggregator node.

Figure 2: The different network topologies used in the project with varying numbers of source and aggregator nodes. The number $/i$ refers to the node's ID.

3.2 Sensor Data Generation

Each node generates synthetic sensor data to emulate real-world sensing behavior. The generated samples, denoted as X, follow a uniform distribution:

$$X \sim \mathcal{U}(X_{min}, X_{max})$$

where $X_{min} \sim \mathcal{U}(X_{min_0} - \delta_X, X_{min_0} + \delta_X)$ and $X_{max} \sim \mathcal{U}(X_{max_0} - \delta_X, X_{max_0} + \delta_X)$ are also sampled from uniform distributions to ensure some variability in the samples from the different nodes. The nodes are configured with a randomized transmission interval $T_{send} = T_{base} + J$ where J is a random jitter term sampled uniformly $J \sim \mathcal{U}(-T_{jitter}, T_{jitter})$.

We recognize that uniformly distributed sensor data is rarely encountered in naturally occurring phenomena, which more commonly follow distributions such as a Gaussian. However, sampling from a Gaussian distribution requires more advanced mathematical functions, such as ln, $\sqrt{}$, cos, and sin, when using, for example, the Box-Muller transform. Due to limited programming space, this was not viable, as including the math.h library exceeded the available memory. Consequently, we opted for a simple uniform distribution, which still allowed us to develop a sophisticated data aggregation function within the resource constraints.

3.3 Simulation of Malfunctioning Nodes

Nodes are equipped with a button sensor, which allows manual simulation of malfunctioning behavior. Of course, malfunctioning behavior can be expressed in many ways, but

in this project, we assume a drastic change in the sampling interval or no samples at all indicate malfunctioning behavior.

During normal operation, data samples are generated within the range $[X_{min}, X_{max}]$ (section 3.2), however, when the button is pressed, the sampling range is changed to $[X_{fault,min}, X_{fault,max}]$ where, $X_{min} \ll X_{fault,min} < X_{fault,max}$ and $X_{max} \ll X_{fault,max} > X_{fault,min}$. Pressing the button again completely stops the source node from generating samples. Pressing the button once again restores normal operation with X_{min} and X_{max} reinitialized.

3.4 Online Statistical Summaries of Samples

Each sensor node computes the mean (μ) and variance (σ^2) of the most recent 100 samples using a sliding window of size $W = \min(N, 100)$ where N is the total number of samples received so far. This approach minimizes memory usage and avoids computational overhead. To achieve this we maintain:

- A running sum S for the samples in the window.
- A running sum of squares SS.

In the following, subscripts denote the time associated with the current variable, rather than the node ID. When a new sample X_0 arrives, the oldest sample X_N is removed if the window is full ($W = 100$). The updates are defined as:

$$S_{i+1} = S_i + X_0 - X_N \tag{7}$$

$$SS_{i+1} = SS_i + X_0^2 - X_N^2 \tag{8}$$

Where

$$X_N = \begin{cases} 0 & \text{if } W < 100, \\ \text{oldest sample in window} & \text{otherwise} \end{cases} \tag{9}$$

The mean and variance are then calculated using the updated values:

$$\mu_{i+1} = \frac{S_{i+1}}{W} \tag{10}$$

$$\sigma^2_{i+1} = \frac{SS_{i+1}}{W} - \mu^2_{i+1} \tag{11}$$

Theoretically, this would be part of the aggregator function since this is the only place we use the computed variance. However, by computing these values directly on the source node before sending them to the aggregator node, a potentially significant amount of computation can be offloaded from the aggregator node. Since we use the same motes, and therefore the same processor, on all types of nodes, we determined that it was better to distribute the computation more evenly throughout the network. This allows each aggregator node to handle more sensor nodes, enabling the system to be scaled up more efficiently at the cost of bigger network packet sizes from each source.

3.5 Data Aggregation Function

Aggregator nodes process data from a fixed number of source nodes, and possibly another aggregator node (fig. 2). When the node receives an aggregator message, it forwards the message immediately within the network. Conversely, when the node receives a source message, the data sample is incorporated into an aggregated value. However, if another source message from the same node arrives before the aggregated value is transmitted, the new source message is discarded.

The data aggregation function is designed to be robust to malfunctioning nodes. If a node i malfunctions, as described in section 3.3, the drastic change in the sampling interval will influence σ_i^2. Therefore, we use this as an indication of how much we can trust the sample generated by node i. More specifically, we generate a weighted average of the samples received using *inverse variance weighting* as shown in eq. (12), where $\bar{X}$ is the aggregated value, X_i is the sample generated by the ith node and σ_i^2 is the variance for the ith node.

$$\bar{X} = \frac{\sum_{i=1}^{N} \frac{1}{\sigma_i^2} \cdot X_i}{\sum_{i=1}^{N} \frac{1}{\sigma_i^2}}, \qquad \sigma_i^2 \neq 0 \tag{12}$$

The benefit of this aggregation function is that samples with a higher uncertainty will count less in the aggregated value and thereby reduce the uncertainty in the aggregated value. If one node malfunctions, the aggregated value will be more or less unaffected. However, if all source nodes included in the aggregated value, malfunction, the aggregated value will also change. This reflects a realistic case, as we assume all nodes describe the same phenomena. That is, if all nodes' distributions suddenly change, it is likely a sign that the underlying phenomena have changed and not that all the nodes malfunction.

As shown in eq. (12), the aggregation function assumes that the aggregator node has received messages from all N source nodes. However, if a source node completely stops transmission, an alternative method for aggregating the available data is required. This is achieved by introducing a timeout $T_{timeout}$ parameter for the aggregator node. Once this timeout expires, the aggregator processes the data it has received so far. It is important that $T_{timeout}$ is not shorter than T_{send}, as doing so could result in aggregation occurring before all sensor data is received.

3.6 Real-time Data Visualization

To facilitate analysis and interpretation of the behavior of the network, we developed a real-time data visualization tool. This tool presents sensor values and aggregated results through two complementary graphical formats. These formats are shown in figs. 3 and 4, and are also shown in the video linked to in table 1. The upper panel displays a series of box plots, one for each source node, providing a statistical summary of node-level data, including the range, median, and spread of sampled values. These boxplots allow for the immediate identification of outliers or nodes whose readings deviate significantly from those of their peers. The system computes and reports the mean and variance for each source node, offering numerical indicators of node stability and reliability.

In the lower panels, time-series line plots depict both individual node readings, the corresponding aggregated values, and the missed data. This temporal perspective highlights evolving trends, correlations, and anomalies as they arise, allowing a dynamic understanding of the network performance over time. Although visualization is constrained by a fixed window size (e.g., displaying the most recent 50 samples), it nevertheless provides a valuable real-time vantage point from which to assess data integrity, detect faults, and evaluate the effectiveness of the chosen aggregation strategy.

To access the node data on the host computer running Cooja, we start the server socket on each node. Next, we run a simple Python script in the virtual machine that listens on these ports and logs the mote output to individual files (i.e. everything printed using printf in the code) directly in a folder shared between the virtual machine and the host computer.

4 Experiments & Results

In order to evaluate our system, we employ both quantitative and qualitative assessment methods. Although the quantitative metrics defined in section 2.4 (completeness, latency, and accuracy) provide objective measures of system performance, they may not fully capture certain aspects of network behavior, particularly those related to outliers or abrupt changes in sensor readings. For this reason, we supplement our quantitative analysis with qualitative evaluations based on real-time data visualization.

4.1 Quantitative Evaluation

In our quantitative evaluation, we employ the performance metrics defined in section 2.4 to objectively measure and compare the performance of the WSN under varying conditions. The evaluation process involves running simulations for a specified duration, during which each source node continuously generates and transmits sensor readings. Aggregator nodes process these readings according to the data aggregation function described in section 3.5, and the sink node, i.e. the laptop in this project, collects aggregated messages over time. To facilitate systematic analysis, we conduct multiple experiments, each characterized by distinct network topologies, node configurations, or operational scenarios. For each experiment:

- **Computing Latency:** By matching each source message to its corresponding aggregator message, we calculate the latency as shown in eq. (3). This metric captures how quickly data moves through the network, from source generation to sink reception, reflecting the efficiency of routing, aggregation, and communication protocols.
- **Computing Accuracy:** Accuracy is evaluated using MAE (eq. (5)) between the source readings and the aggregated values. This quantifies how closely the aggregation function preserves the essential characteristics of the underlying data, balancing communication reduction with fidelity.

- **Computing Completeness:** Using eq. (6), we quantify the effectiveness with which the network delivers the source data to the sink. Completeness indicates if the system reliably processes and transmits sensor information, even when network conditions vary or the nodes malfunction.

In combination, completeness, latency, and accuracy provide a well-rounded view of the network's operational effectiveness. By thoroughly evaluating these metrics under controlled experimental conditions, we can draw conclusions about the strengths and limitations of our proposed data aggregation approach and identify avenues for further optimization.

4.2 Qualitative Evaluation

The qualitative evaluation of the system is used to assess its robustness to outliers. This is done through real-time data visualization (section 3.6), where we observe how the plots change when nodes malfunction. A qualitative approach was chosen because metrics like MAE, completeness, and latency are not well-suited for capturing the effect of outliers. MAE primarily measures overall error, which can obscure localized issues caused by malfunctioning nodes. Completeness focuses on the proportion of data successfully processed, without highlighting the influence of anomalies. Latency measures timing performance, which does not reflect the structural or visual impact of outliers. By relying on real-time visualization, we can directly examine how the system handles these scenarios.

4.3 Results and Analysis

This section presents the outcomes of our experiments, focusing on how various topologies and scenarios affect network performance when using data aggregation.

Experiment Setup					**Results**		
Exp.	**Top.**	**Sim. time**	**Nodes malfunctioning**	**Nodes Inactive**	**MAE**	**Compl.**	**Avg. Lat.**
1	1	25:04.943	-	-	20.14	96.87%	1.28s
2	2	25:37.538	-	-	19.98	96.48%	1.25s
3	3	25:30.640	-	-	20.41	96.96%	1.16s
4	1	25:19.507	7 at $t \approx 5$	-	61.90	96.52%	1.26s
5	1	25:12.863	7 at $t \approx 5$	7 at $t \approx 10$	34.65	80.61%	1.57s
Demonstration of real-time data visualization tool: YouTube link							

Table 1: Experiments and Results

Table 1 summarizes the key results for five representative experiments. In general, the network demonstrates high completeness (often above 95%) and maintains relatively

low latency, suggesting that data is efficiently routed and delivered to the sink. The MAE values for Experiments 1, 2, and 3, which involve no malfunctioning or inactive nodes, hover around 20 units. This demonstrates that our aggregation approach effectively preserves data quality under normal conditions. Some errors are, however, inherent, as the aggregation process compresses data from multiple nodes into a single value. Given that each node samples independently within its interval, a degree of error is unavoidable.

Robustness to Malfunctioning and Inactive Nodes

Experiments 4 and 5 introduce malfunctioning and inactive nodes to assess how the system handles abnormal conditions. In Experiment 4, node 7 malfunctions around $t \approx 5$s, resulting in a higher MAE (61.90) due to the inclusion of erroneous or skewed data. However, completeness remains high (96.52%), and latency does not suffer. In Experiment 5, node 7 first malfunctions and then becomes inactive at $t \approx 10$s. Here, completeness drops to 80.61% as the number of successfully integrated measurements decreases, and average latency increases to 1.57s. This is due to the timeout mechanism explained in section 3.5. The MAE is not as high in Experiment 5 as in Experiment 4 since node 7 completely ceases to transmit and therefore less erroneous samples are included in the aggregated values.

To also qualitatively assess the robustness to outliers, figs. 3 and 4 present snapshots from the real-time visualization tool showing the behavior when one or more nodes malfunction. In fig. 3, the button of node 7 is pressed, simulating a malfunction. This causes a drastic skew in its distribution (top plot). However, as shown by the blue line in the middle plot, the aggregated value remains largely unaffected, demonstrating that the aggregation function is robust to outliers. In fig. 4, the buttons of nodes 7, 8, and 9 are all pressed, resulting in similar distributions (top plot). Since no node is identified as an outlier, the aggregated value shifts to reflect the changed distribution. These two scenarios illustrate the cases where a single node is an outlier and where the underlying phenomenon itself changes.

4.4 Message Reduction

Data aggregation reduces the amount of messages we need to send and receive in the WSN. By reducing the amount of packages we send, we also reduce the amount of power consumed by the network. Some packets have increased in payload size because we include sensor statistics and the sampled data during aggregation. While this increase in packet size requires slightly more power for transmission, it is not as significant as the additional power required for sending a higher number of packets. However, it is important to note that the calculations needed for data aggregation also demand a comparatively small amount of extra computational power, which can contribute to the overall energy consumption. As shown in table 2, the message reduction percentage increases as the level of aggregation in the multi-hop network grows. This effect is particularly pronounced when there are more hops between the sink and the sources, where the potential for reducing the number of messages is greatest.

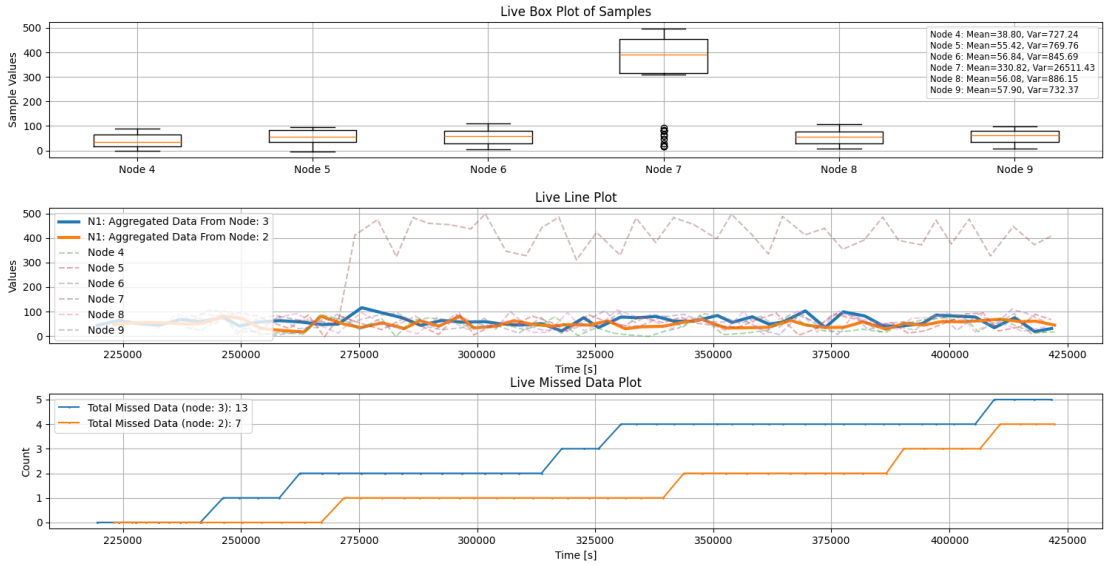


Figure 3: Showcasing of robustness to outliers. The distribution of node 7 is very different from that of the other nodes (top plot), but the aggregated value has not been affected (blue line, middle plot).

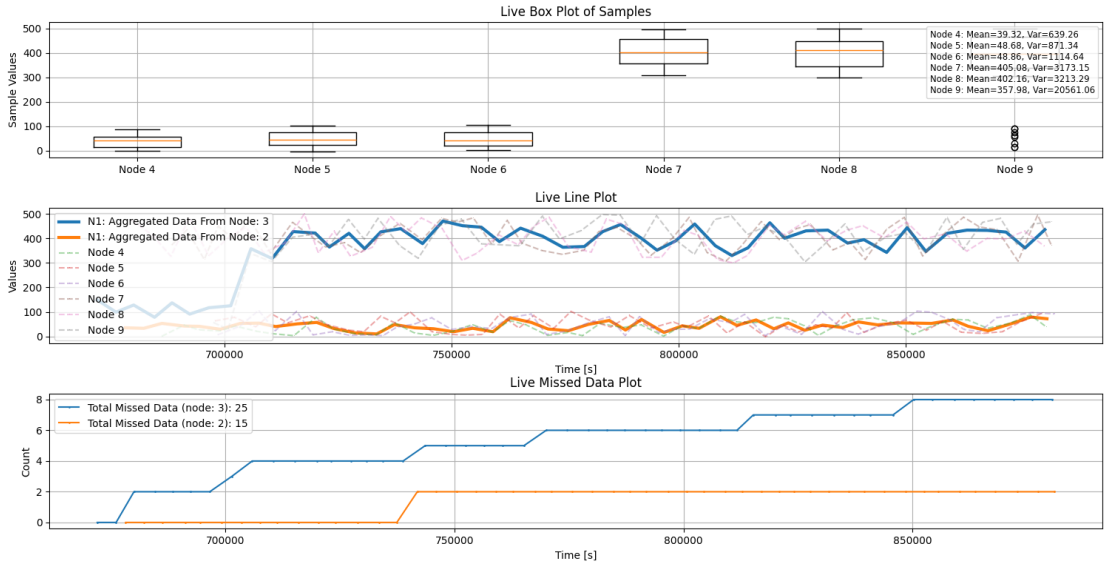


Figure 4: Showcasing of phenomena shift. The distribution of both nodes 7, 8 and 9 have changed (top plot) and therefore the aggregated value also reflects this change in behavior (blue line, middle plot).

	Topology 1	**Topology 2**	**Topology 3**
Without Aggregation (Sent/Received)	15	6	3
With Aggregation (Sent/Received)	9	4	3
Reduction Percentage	40%	33.3%	0%

Table 2: Message Reduction with Aggregation Across Topologies

5 Discussion

This section examines the key challenges, trade-offs, and limitations of our approach, while also suggesting potential improvements. It focuses on the impact of node loss, energy efficiency, and the known limitations of the aggregation function.

5.1 Node Loss

Losing a node, i.e. the aggregating node stops receiving data samples, is not handled gracefully with our data aggregation function. The aggregator's timeout mechanism ensures that data still flows, albeit with reduced completeness. A simple forwarding scheme would be better as there would be no time wasted waiting because of the node that is not responding. Simply optimizing the wait time of the aggregating node before it proceeds without the missing node's sample could enhance the completeness of this scenario. However, a more long-term approach to solve this could be incorporating heartbeat signals between the aggregator and sources. If no heartbeat signal is received from a source node, then the aggregator stops waiting for that node until a heartbeat signal is detected again. Heartbeat signals should be exchanged periodically, but at intervals significantly longer than the actual sample data to avoid overwhelming the network with these messages, which would eliminate the message reduction achieved by aggregating in the first place.

5.2 Message Reduction and Potential Energy Efficiency

Our results confirm that data aggregation can effectively reduce the number of messages transmitted, potentially lowering energy consumption. However, this reduction is not without trade-offs. For one, the process of computing the aggregation statistics at each source node increases the computational overhead locally. Although we assume this overhead to be minimal, the cumulative effect in a large-scale network remains unclear. Moreover, the observed message reduction is highly dependent on the chosen topologies and distributions of sensor readings; in sparser or less correlated networks, the benefits could be significantly diminished. Future work could evaluate whether the apparent energy savings outweigh the additional computational costs and how this balance changes as the density and complexity of the network increase.

5.3 Aggregation Function Limitations

In our aggregation function, we use the variance to weight the incoming samples. When the variance is high the sample from the corresponding node is not weighted as high in the aggregated value. Doing this makes this approach ideal for a scenario where a malfunction leads to high variance. However, there are several ways a malfunction could lead to a variance similar to or lower than that of non-malfunctioning nodes, while having a mean value that is significantly off. If the variance is the same but the mean is skewed by a constant, then the malfunctioning nodes' data is weighted the same although it is not desirable. A particularly problematic scenario for our approach occurs when a sensor becomes "stuck" on a single value. This would cause the variance to drop to zero, potentially resulting in a division by zero in our aggregation function unless we implement a specific check to handle this case. Exploring more nuanced weighting criteria, such as historical performance, or cross-correlation among nodes, could result in a more robust and adaptive aggregation strategy for outlier detection.

6 Conclusion

This project demonstrated that employing data aggregation in a multi-hop WSN can significantly reduce communication overhead while maintaining high data completeness and low latency. By using inverse-variance weighting to compute aggregated values, the system proved robust against single-node outliers: malfunctioning sensors producing unreliable readings did not greatly affect the aggregated results. Even when the distribution of one node shifted drastically, the aggregated value remained stable, reflecting the network's capacity to filter out erroneous samples. In scenarios where multiple nodes simultaneously deviated from expected behavior, the aggregation naturally adjusted its output, indicating that a sudden and widespread change likely stemmed from the underlying phenomenon rather than from a single faulty node.

However, challenges remain. For instance, node loss led to temporary decreases in completeness, as the system waited for missing data and timed out before proceeding. In addition, certain malfunction patterns—such as stuck sensors with low variance—could bypass variance-based weighting schemes and complicate outlier detection. Future work may refine the coordination among nodes, incorporate adaptive aggregation criteria that consider historical trends, and introduce heartbeat signals or other signals for better handling of dynamic failure. With such enhancements, data aggregation can become even more energy-efficient, scalable, and resilient, ensuring robust and accurate information delivery despite unpredictable network conditions.

References

[1] H. Karl and A. Willig, *Protocols and Architectures for Wireless Sensor Networks.* Chichester, England: Wiley, 2005.