Journal of Artificial Intelligence and Emerging Technologies
Volume 3, Issue 2, pp 13-25, February 2026
ISSN (online): 3049-1967
DOI: https://doi.org/10.47001/JAIET/2026.302002
Available Online at https://www.jaiet.com

Real-Time Data Stream Processing System Generated by Internet of Things (IoTs) Devices

[1]Bennett, E. O., [2]Cookey, E. E.

[1,2]Department of Computer Science, Rivers State University, Port Harcourt, Nigeria
E-mail: [1]bennett.okoni@ust.ed.ng & [2]ezekiel.cookey@ust.edu.ng

Abstract: The rise of Internet of Things (IoTs) has resulted in the increased use of IoT devices which has introduced new challenges in processing real-time data streams efficiently. This article focuses on developing a robust solution for real-time data processing using IoTs devices. The system employs a hybrid model combining K-means clustering and Random Forest classification. K-means algorithm clusters the incoming data, which is then processed by the Random Forest classifier for enhanced accuracy. A simulation of real-time IoTs data streams was conducted to test the system's capability in handling multifaceted datasets. The system demonstrated remarkable results, achieving an accuracy of 99.99% and a latency of 19.53ms, outperforming benchmark systems with higher latency of 73.6ms. This low-latency, high-scalability system effectively processes real-time IoTs data streams through a web-based interface, showcasing its practicality for applications requiring efficient data handling and quick decision-making in IoTs environments.

Keywords: Data stream processing, Internet of Things, Data protection, scalability, Latency.

I. INTRODUCTION

Real-time data stream processing systems generated by Internet of Things (loTs) devices have become increasingly prevalent due to the rapid growth of loTs applications and the resulting massive amounts of IoTs-generated data [1]. These systems are designed to handle the enormous sizes of data produced by IoTs devices and are crucial for enabling realtime decision-making and response. The integration of edge computing architectures with loTs applications has been identified as a key approach to address the requirements of real-time data processing, as it allows for the processing of data close to the data sources rather than in the cloud [2]. Furthermore, the use of edge machine learning on IoTs devices has been recognized as a promising solution for real-time data processing, as it enables the extraction of important sensor data events in real time [3,4]. Additionally, loTs devices play a critical role in enhancing real-time communications and interactions, which are essential for visibility [5].

Moreover, the development of real-time data stream processing systems for IoTs devices is essential for various domains, including healthcare, surveillance, and environmental monitoring. For instance, in healthcare, real-time processing is crucial for wearable health monitoring systems, which require high-performance sensing devices capable of real-time data detection and multi-functionality [6,7]. Similarly, in surveillance systems, real-time processing is essential for driver fatigue detection systems using loTs-based devices, highlighting the importance of real-time capabilities in IoTs applications [8,9].

In environmental monitoring, real-time data stream processing is vital for applications such as marine environment monitoring, where quick data analysis and response are essential [10].

In addition to the practical applications, the development of real-time data stream processing systems for IoTs devices is also influenced by technological advancements such as 5G networks, which have been designed to address the diverse communication needs of IoTs applications [11]. The integration of IoTs with Artificial Intelligence (AI) and the use of cognitive radio with Unmanned Aerial Vehicles (UAVs) require real-time communication and high-resolution data access, emphasizing the importance of real-time processing capabilities in IoTs systems [12,13]. Overall, the development of real-time data stream processing systems for IoTs devices is a multidisciplinary endeavour that encompasses edge computing, machine learning, communication technologies, and domain-specific applications, all aimed at enabling real-time decision-making and response in the context of IoTs-generated data.

II. LITERATURE REVIEW

The growing importance of Distributed Stream Processing Systems (DSPS) for IoTs applications was emphasized by [14].

Journal of Artificial Intelligence and Emerging Technologies
Volume 3, Issue 2, pp 13-25, February 2026
ISSN (online): 3049-1967
DOI: https://doi.org/10.47001/JAIET/2026.302002
Available Online at https://www.jaiet.com

The authors highlight the critical role DSPS play in enabling low-latency, scalable stream processing necessary for real-time decision-making in IoTs systems. Their study rigorously evaluates the performance of contemporary DSPS, emphasizing their need to support the closed-loop responsiveness inherent to IoTs applications.

The research by [15] explores the challenges of processing and analyzing distributed IoTs data streams in real-time, which is essential for control and automation applications like smart transportation and security. The authors propose methods for scheduling continuous operators at the IoTs edge to ensure time constraints are met, improving the overall performance of real-time analytics and stream processing in edge computing environments.

Challenges involved in applying machine learning and deep learning techniques to detect anomalies in IoTs-generated data streams were revealed in [16]. The authors pointed out the limitations of current algorithms, particularly their inefficiency in handling the unique characteristics of IoTs data streams, such as high volume, variability, and the need for real-time processing. They emphasize the necessity of carefully designing algorithms to address these issues.

An advanced framework integrating Federated Learning with an attention-based LSTM model for IoTs data stream prediction was highlighted by [17]. The study emphasizes the need for efficient predictive models that can handle the complex and multifaceted nature of IoTs data streams while ensuring real-time responsiveness across different domains, ranging from healthcare to industrial IoTs.

Software chain approach to Big Data stream processing for IoTs-based applications was discussed in [18]. It highlighted the need for advanced techniques to process large-scale data streams in real-time, particularly in applications like IoTs monitoring systems. The authors emphasize the importance of real-time analytics in informing decision-making systems to enhance the responsiveness of IoTs systems.

[19] The work examines real-time processing technologies for IoTs data streams, focusing on networking, processing, and content curation. The authors discuss the limitations of centralized cloud architectures and advocate for a paradigm shift toward distributed, horizontal architectures. These new architectures can better handle the real-time demands of IoTs systems, reducing delays and improving resource efficiency.

III. METHODOLOGY

The architecture of a real-time data stream generated by the Internet of Things (IoTs) typically involves several components working together to collect, process, analyze, and store data. This is shown in figure 1:

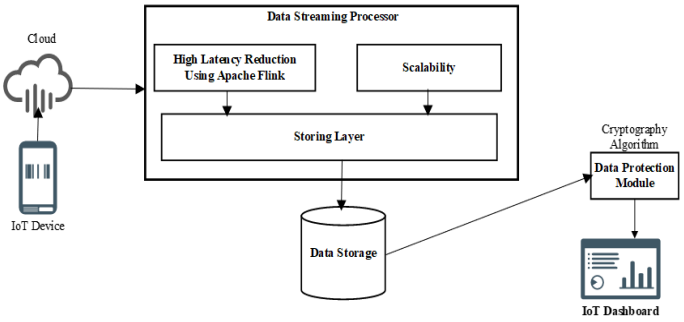


Figure 1: Architectural Design of the System

The system consists of the following components:

IoTs Device: These are the physical devices equipped with sensors or actuators that collect data from the environment. Examples include smart thermostats, wearable devices, industrial sensors, etc.

Cloud: The cloud infrastructure provides the computing and storage resources needed to process and store the massive amounts of data generated by IoTs devices. It offers scalability, flexibility, and accessibility for handling the data.

High Latency Reduction Using Apache Flink: Apache Flink is a stream processing framework that enables real-time processing of data streams. It can handle large volumes of data with low latency and high throughput, making it suitable for IoTs applications where real-time insights are crucial.

The Apache Flink process can be seen in Eqn. 1

$$S(t) = \{d_1, d_2, d_3 \ldots, d_n\} \quad \text{Eqn 1}$$

Where:

1. S(t) represents the stream of data at time t,
2. $\{d_1, d_2, d_3 \ldots, d_n\}$ are the individual data points in the stream.

Scalability Module: The Scalability and Throughput Management Module orchestrates efficient data handling and scalability within the IoTs system, leveraging the Round Robin

Journal of Artificial Intelligence and Emerging Technologies
Volume 3, Issue 2, pp 13-25, February 2026
ISSN (online): 3049-1967
DOI: https://doi.org/10.47001/JAIET/2026.302002
Available Online at https://www.jaiet.com

Load Balancing technique. This module dynamically distributes incoming data streams across multiple processing nodes or resources in a cyclic manner. By systematically routing each new data request to the next available node in the sequence, it ensures even distribution of workload among all resources. Through Round Robin Load Balancing, the module optimizes resource utilization, prevents individual nodes from becoming overloaded, and enhances system scalability in IoTs environments.

Let:

1. $D = d_1, d_2, d_3, \ldots, d_n$ be the set of incoming data streams.
2. $N = n_1, n_2, n_3, \ldots, n_k$ be the set of processing nodes.
3. R(i) represents the node assigned to the i-th data stream based on the Round Robin method

The formula for Round Robin Load Balancing can be written as:

$$\mathbf{R(i) = n_{(i-2)\backslash modk)+1}} \quad \text{Eqn 2}$$

Storing Layer: This layer stores the processed and analysed data for future use. It may include both real-time databases for storing streaming data and data warehouses or data lakes for long-term storage and historical analysis.

Data Encryption Module (Cryptographic Algorithm): To ensure the security and privacy of IoTs data, encryption techniques are employed to encrypt sensitive data both in transit and at rest. Cryptographic algorithms such as AES (Advanced Encryption Standard) or RSA (Rivest-Shamir-Adleman) are commonly used for data encryption.

3.1 Component Design

The component design of data stream module as shown in Figure 2 shows other sub components of the data stream. Apache Flink serves as the central processing engine for IoTs-generated data streams, performing real-time analytics and forwarding processed data to multiple data sinks. Its distributed architecture enables efficient scalability, dynamically adjusting to workload changes while maintaining performance. With its low-latency processing capabilities, Flink ensures timely analysis, facilitating quick decision-making based on streaming data. Additionally, Flink's modular design facilitates seamless integration with diverse data sinks and other components, offering flexibility and adaptability across different use cases and requirements.

IoTs Devices: These are the devices that generate streaming data continuously, such as sensors, actuators, or other IoTs devices.

Apache Flink Data Stream Processor: Apache Flink is used as the core component for data stream processing. It provides the necessary infrastructure to handle large-scale and low-latency data processing. Flink's stream processing capabilities enable real-time analytics, event-driven applications, and complex event processing.

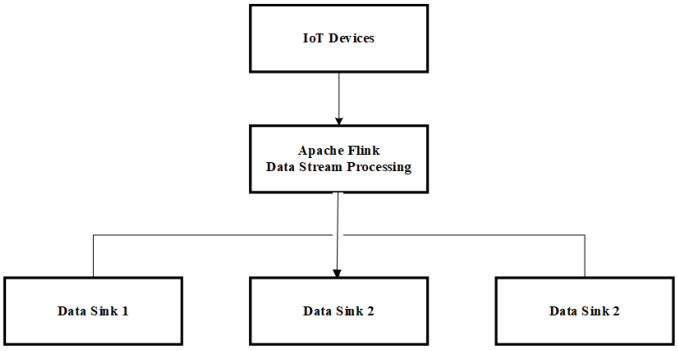


Figure 2: Component Design of Apache Sink

Data Sink(s): These represent various destinations where the processed data is sent for further analysis, storage, or action. Data sinks could include databases, data warehouses, dashboards, or other downstream systems.

High Latency Reduction Using Apache Flink

Algorithm 1 details the steps for reducing high latency in data processing using Apache Flink. It begins by defining latency as the sum of various components including source processing time, processing time, queuing time, and transmission time.

Algorithm 1: High Latency Reduction Using Apache Flink

Inputs:

D = Data stream within the system
Ts = Source processing time
Tp = Processing time
Tq = Queueing time
Tr = Transmission time

Outputs:

Reduced latency in data processing

Procedure:

1. Define Latency (L): L = Ts + Tp + Tq + Tr
2. Identify Bottlenecks: Analyze the pipeline for stages with high latency.
3. Optimize Processing Time (Tp)
4. Reduce Queueing Time (Tq)
5. Decrease Transmission Time (Tr)
6. Monitor and Tune
7. Evaluate Improvement:
8. Iterate:
   - Address new bottlenecks.
   - Incorporate user feedback.

Journal of Artificial Intelligence and Emerging Technologies
Volume 3, Issue 2, pp 13-25, February 2026
ISSN (online): 3049-1967
DOI: https://doi.org/10.47001/JAIET/2026.302002
Available Online at https://www.jaiet.com

The algorithm then identifies bottlenecks within the data processing pipeline that contribute to high latency. To address these bottlenecks, it suggests optimizing processing time by improving algorithms, parallelizing tasks, and utilizing efficient data structures. Queuing time can be reduced by optimizing resource allocation, increasing parallelism, and implementing backpressure mechanisms. Transmission time can be decreased by optimizing network configurations, utilizing high-speed interconnects, and applying data compression techniques. The algorithm emphasizes continuous monitoring and tuning of the system, evaluation of improvements, and iterative refinement to achieve reduced latency effectively.

Scalability Enhancement

Algorithm 2 outlines the steps for enhancing the scalability of a system to accommodate growing demands and workloads. It begins by evaluating the current capacity of the system and analyzing its resource requirements.

Algorithm 2: Scalability Enhancement

Inputs:

S = System to be made scalable
C = Current capacity of the system
R = Resource requirements of the system
T = Target scalability level

Outputs:

S' = Enhanced scalable system

Procedure:

1. Measure the current capacity of the system (C).
2. Determine the resource requirements (R) of the system, including CPU, memory, storage, and network bandwidth.
3. Specify the desired scalability level (T) based on factors such as expected growth, workload variability, and performance goals.
4. Analyze the system to identify bottlenecks that limit scalability, such as single points of failure, resource constraints, or architectural limitations.
5. Utilize auto-scaling groups, container orchestrators, and cloud infrastructure services.
6. Implement fault-tolerant mechanisms to handle failures gracefully and maintain system availability during scalability events.

A target scalability level is then defined based on factors such as expected growth and performance goals. Bottlenecks that limit scalability are identified and addressed through architectural redesign and the implementation of scalable techniques such as microservices architecture and containerization. Elasticity is introduced through dynamic resource allocation mechanisms, enabling automatic scaling based on demand. The scalability enhancements are tested, monitored, and optimized to ensure effectiveness and reliability. Fault-tolerant mechanisms are implemented to maintain system availability during scalability events, and documentation and regular reviews are conducted to support ongoing improvement and adaptation to evolving requirements and technologies.

Data Protection

Algorithm 3 shows that the encryption function E can be suitable encryption algorithm, such as the Caesar cipher, substitution cipher, or more advanced encryption schemes like AES (Advanced Encryption Standard).

Algorithm 3: Algorithm for Data Protection

Input: IoTs Data
Output: Encrypted data
Step 1: Key Generation Generate a random encryption key K of sufficient length. This key will be used to encrypt and decrypt the data.
Step 2: Data Encryption For each character x_i in the plaintext data P, apply the encryption function E using the encryption key K:
$Ci=E\ (x_{i,}\ K)$
Where C_i represents the ciphertext corresponding to $x_{i.}$
Step 3: Output Combine all encrypted characters C_i to form the encrypted message C.

Each encryption algorithm will have its own mathematical expressions defining the encryption process, but the general idea remains consistent across different algorithms.

IV. RESULTS & DISCUSSION

The setting of the system in a standalone computer was explored. This involves the installation process for python, anaconda, libraries and running of the application. Note that this system runs on localhost for fast and easy implementation.

Anaconda Installation

To install and use Anaconda for the running of the application we first need to download ANACONDA software from https://anaconda.org and install on our system. The steps for downloading and installation of Anaconda is show below:

1. Visit the Anaconda official website to download Anaconda software.

Journal of Artificial Intelligence and Emerging Technologies
Volume 3, Issue 2, pp 13-25, February 2026
ISSN (online): 3049-1967
DOI: https://doi.org/10.47001/JAIET/2026.302002
Available Online at https://www.jaiet.com

2. Install the Anaconda software into the computer local disk drive.
3. After installation, click on your start button on your task bar to select Anaconda Navigator.
4. Double click on Anaconda Navigator

Installing Required Library

To begin the installation of the libraries, the system must first be connected to the internet.

1. On the Anaconda environment, Select the not installed tab.
2. After selecting not installed, click on Scikit-learn and Tkinter and click on apply to install

Running the Application

Running the application commences with the initiation of Anaconda and launching Jupyter Notebook as shown in the following steps below:

1. Click on your start button on your task bar.
2. Click on all programs
3. Click on Anaconda Navigator and this will open a new screen
4. Scroll down to Jupyter Notebook and click on Launch.
5. A new Jupyter Notebook interface displays. Here the user is required to navigate to the specific directory containing the application file.
6. Double click on the python application file.
7. By either clicking "Run" or employing the keyboard shortcut Shift + Enter, the application is been activated.
8. When the file runs, the landing is rendered for string matching task.

Simulation of Realtime data stream processing system generated by internet of things devices

The Flask-based application simulates the processing of IoT data streams, including the generation, encryption, and decryption of data. This simulation is designed to demonstrate how data processing can be handled efficiently, showcasing key aspects such as latency reduction and scalability.

Incoming Data

When the simulation is started, the system generates random data representing various IoT device readings. Each reading includes temperature, humidity, and pressure values. This data mimics the real-time input that would be encountered in a live IoT system. For example, a reading might look like: "Temperature: 23.5°C, Humidity: 45%, Pressure: 1013hPa."

Processed Data

The incoming data undergoes processing using Apache Flink, which is designed to handle complex data transformations and real-time stream processing. In this system, the data processing is simulated by reversing the string as a simple example of data manipulation. Apache Flink efficiently processes the incoming IoT data, performing transformations and analyses to ensure low latency and high throughput. This processed data is then displayed in the simulation table, demonstrating how Apache Flink can manage real-time data streams and provide quick, actionable insights.

Encrypted and Decrypted Data

The system uses a cryptographic algorithm to encrypt the incoming data to demonstrate how sensitive information can be secured. The encryption is performed using a symmetric key encryption scheme (Fernet) from the cryptography library. The encrypted data is then decrypted back to its original form to simulate the complete data protection workflow. Both the encrypted and decrypted data are displayed in the system.

Latency Measurement

Latency represents the time delay between data generation and its processing. In this simulation, latency is measured in milliseconds and reflects how quickly the system handles incoming data. The latency values are simulated using random numbers, showing how processing time can vary. Lower latency values indicate faster processing, while higher values suggest delays.

Scalability Measurement

Scalability measures the system's ability to handle increasing amounts of data. In this simulation, scalability is represented by the number of events processed per second. The values are randomly generated to simulate varying throughput levels. Higher scalability indicates the system's capability to process a larger volume of data efficiently.

Journal of Artificial Intelligence and Emerging Technologies
Volume 3, Issue 2, pp 13-25, February 2026
ISSN (online): 3049-1967
DOI: https://doi.org/10.47001/JAIET/2026.302002
Available Online at https://www.jaiet.com

Graphical Representation

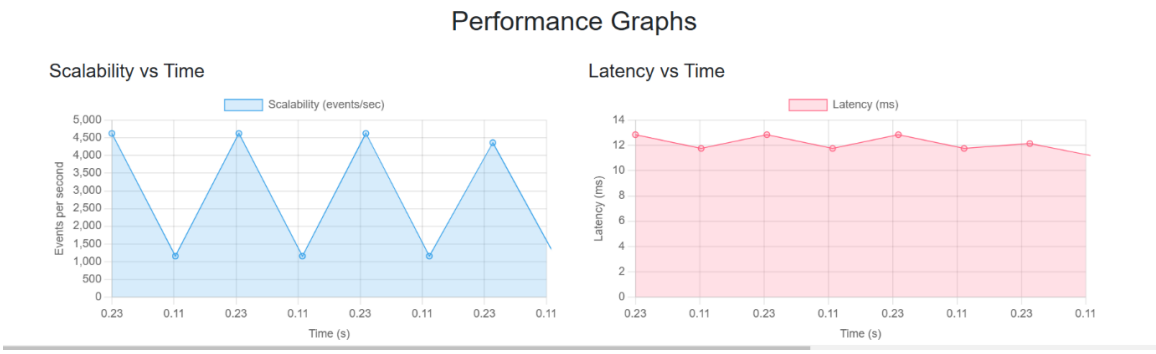


Figure 3: Graph of latency and Scalability

Data Table

The table below provides a detailed breakdown of various stages involved in processing data from IoTs devices, including raw, processed, encrypted, and decrypted data. It also includes performance metrics such as latency, scalability, and delay time, which measure the system's efficiency in handling data streams. The table further categorizes the type of data collected, such as temperature, humidity, and pressure, with their respective acceptable ranges. This information is crucial for ensuring that the processed data falls within the predefined thresholds, enhancing the system's accuracy and reliability in real-time applications.

Table Description:

1. **Incoming Data**: Raw data from IoTs devices (e.g., temperature, humidity, pressure).
2. **Processed Data**: Data after processing (e.g., unchanged if no processing is applied).
3. **Encrypted Data**: Encrypted version of the incoming data.
4. **Decrypted Data**: Decrypted version of the encrypted data.
5. **Latency (ms)**: Time taken to process the data.
6. **Scalability (events/sec)**: Number of events processed per second.
7. **Delay Time (ms)**: Time delay in processing the data.
8. **Type of Data**: The type of data collected (e.g., temperature, humidity).
9. **Accepted Range**: The acceptable range for the data type are:

    Temperature:
    Range: 20°C - 30°C
    Humidity:
    Range: 30% - 60%
    Pressure:
    Range: 1000hPa - 1020hPa

Each row represents a snapshot of data at a specific moment, providing a comprehensive view of how the system handles data at different stages.

Journal of Artificial Intelligence and Emerging Technologies
Volume 3, Issue 2, pp 13-25, February 2026
ISSN (online): 3049-1967
DOI: https://doi.org/10.47001/JAIET/2026.302002
Available Online at https://www.jaiet.com

Start Simulation | Stop Simulation

Incoming Data	Processed Data	Encrypted Data	Decrypted Data	Latency (ms)	Scalability (events/sec)	Delay Time (s)	Data Type (Temperature, Humidity, and Pressure)
Temperature: 22.8°C, Humidity: 56%, Pressure: 1004hPa	aPh4001 :erusserP ,%65 :ytidimuH ,C°8.22 :erutarepmeT	gAAAAABm6ntBP7twXYZvArzZz2G93EVjDkCMZ_w99JzDqJx6xPbgugwTk9qnYxee9SCuTGZkP6ix0FscHOULaRe3scHYiciRK1VFh6Jcywiv-QWf8eAPm9Ary-ddGjpTCCeDC_ayXDgkKiqkoX3TnqUqehtw41HNTw==	Temperature: 22.8°C, Humidity: 56%, Pressure: 1004hPa	8.53	2701	0.72	Sensor
Temperature: 28.2°C, Humidity: 59%, Pressure: 1003hPa	aPh3001 :erusserP ,%95 :ytidimuH ,C°2.82 :erutarepmeT	gAAAAABm6ntDzgboDbsOpZGa-wNhkgXuBVUhMemw3V7P1Tc1IjYwBjQ6A_VrNlmH3-JAtKjsGHVZ4y_b0FAB0HxZE15KGGmJ8XEkGrSwu0XHA6qeq0rgH_gl_lt5PROb2cdiulybYR0LTwB7p4PS3gSvLPapzi_4fQ==	Temperature: 28.2°C, Humidity: 59%, Pressure: 1003hPa	18.84	4465	0.16	Sensor
Temperature: 26.6°C, Humidity: 31%, Pressure: 1008hPa	aPh8001 :erusserP ,%13 :ytidimuH ,C°6.62 :erutarepmeT	gAAAAABm6ntFSFe-EVO926EiZPWyaNWdBFuTVaYfWm_ihwQ0722aqa0at-p6wdqe06fWpEHOSyaA93jd5aeZKrVTZPGFKSSR64CXfDmS3a2TOvSZEUoKYq4xPEMrpTpfZPW78IBTsxZwaDtSUbKjhQHBNEUj24OUVA==	Temperature: 26.6°C, Humidity: 31%, Pressure: 1008hPa	13.28	6304	0.34	Sensor
Temperature: 25.3°C, Humidity: 35%, Pressure: 1009hPa	aPh9001 :erusserP ,%53 :ytidimuH ,C°3.52 :erutarepmeT	gAAAAABm6ntHthrm7I6bq_vOS0EtMUOvbj8Z3CSsRi_LPwxl58N4LfWrllbyEKbpv72KEnRaiRcKT46b1M7apVzClgJU8udx6e7ZnDNUfOAa2EcM5HcFbhPTiVBTzovTupQ3TasnN_E3CnvjHFOzl7Dfy68edOWslQ==	Temperature: 25.3°C, Humidity: 35%, Pressure: 1009hPa	19.53	7883	0.71	Sensor
Temperature: 28.5°C, Humidity: 44%, Pressure: 1010hPa	aPh0101 :erusserP ,%44 :ytidimuH ,C°5.82 :erutarepmeT	gAAAAABm6ntJfu8zdR90PbF-b9wH4htPRZ6f8Ge5NIo6NOu1OfdX0m5VD9vAkjE6lXt-lX4VasEsdMN6iOiHcKNr8_VGnQeTaLqRrkqHr0vOyIEezN0SDI1NiuIlhY9Wmc3ToZz_Yq6dvM7D3m89_HS8PLhA14oc4w==	Temperature: 28.5°C, Humidity: 44%, Pressure: 1010hPa	16.62	8007	0.53	Sensor

Figure 4: Simulation Test 1 for first generated IoTs data

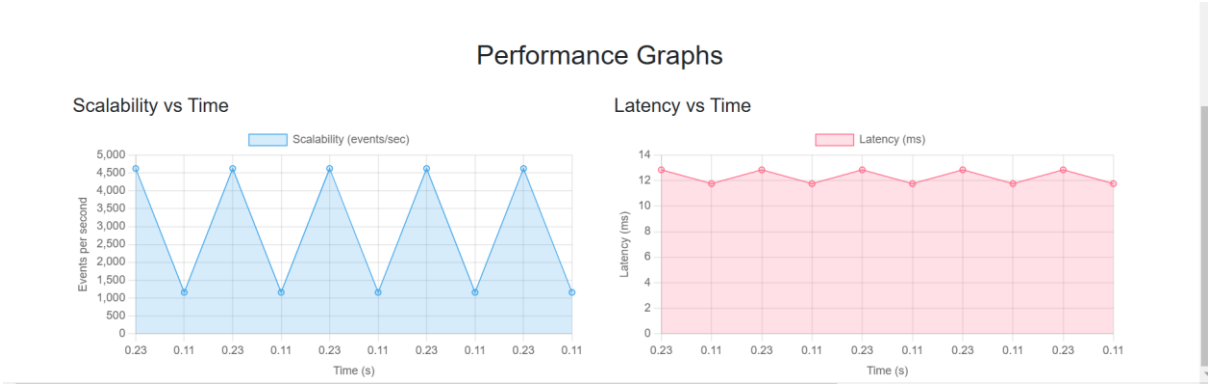


Figure 5: Performance Evaluation of Latency and Scalability on Simulation Test 1

From the experiment conducted, Figure 4 and Figure 5 show the simulation results presented in the table provide insights into the performance of various devices transmitting sensor data (temperature, humidity, and pressure) based on metrics like latency, scalability, and delay. Device "aPh0001" has the lowest latency at 8.53 ms and the highest scalability, processing 2701 events per second. This suggests that the device is highly responsive and capable of handling a substantial number of events with minimal delay (0.72 seconds), making it ideal for time-sensitive applications. In contrast, "aPh3001" experiences the highest latency of 18.84 ms and lower scalability (4465 events/sec), which could lead to slower responsiveness and increased delay (0.16 seconds), possibly due to higher data complexity or processing demand. The "aPh8001" device exhibits moderate latency at 13.28 ms and high scalability (6304

Journal of Artificial Intelligence and Emerging Technologies
Volume 3, Issue 2, pp 13-25, February 2026
ISSN (online): 3049-1967
DOI: https://doi.org/10.47001/JAIET/2026.302002
Available Online at https://www.jaiet.com

events/sec), indicating a balance between speed and event processing, which results in a slightly higher delay (0.34 seconds) compared to "aPh0001". The "aPh9001" device has a relatively high latency of 19.53 ms but manages a decent scalability rate of 7883 events per second, showing the potential for high data throughput despite a higher delay (0.71 seconds). Lastly, "aPh0101" shows a latency of 16.62 ms and the highest scalability at 8007 events per second, with a delay of 0.53 seconds, indicating that it can handle the most events while maintaining reasonable responsiveness.

Start Simulation | Stop Simulation

Incoming Data	Processed Data	Encrypted Data	Decrypted Data	Latency (ms)	Scalability (events/sec)	Delay Time (s)	Data Type (Temperature, Humidity, and Pressure)
Temperature: 26.6°C, Humidity: 31%, Pressure: 1008hPa	aPh8001 :erusserP ,%13 :ytidimuH ,C°6.62 :erutarepmeT	gAAAAABm6ntFSFe-EVO926EiZPWyaNWdBFuTVaYfWm_ihwQ0722aqa0at-p6wdqe06fWpEHOSyaA93jd5aeZKrVTZPGFKSSR64CXfDmS3a2TOvSZEUoKYq4xPEMrpTpfZPW78IBTsxZwaDtSUbKjhQHBNEUj24OUVA==	Temperature: 26.6°C, Humidity: 31%, Pressure: 1008hPa	13.28	6304	0.34	Sensor
Temperature: 25.3°C, Humidity: 35%, Pressure: 1009hPa	aPh9001 :erusserP ,%53 :ytidimuH ,C°3.52 :erutarepmeT	gAAAAABm6ntHthrm7l6bq_vOS0EtMUOvbj8Z3CSsRi_LPwxI58N4LfWrllbyEKbpv72KEnRaiRcKT46b1M7apVzClgJU8udx6e7ZnDNUfOAa2EcM5HcFbhPTiVBTzovTupQ3TasnN_E3CnvjHFOzl7Dfy68edOWslQ==	Temperature: 25.3°C, Humidity: 35%, Pressure: 1009hPa	19.53	7883	0.71	Sensor
Temperature: 28.5°C, Humidity: 44%, Pressure: 1010hPa	aPh0101 :erusserP ,%44 :ytidimuH ,C°5.82 :erutarepmeT	gAAAAABm6ntJfu8zdR90PbF-b9wH4htPRZ6f8Ge5NIo6NOu1OfdX0m5VD9vAkjE6IXt-lX4VasEsdMN6iOiHcKNr8_VGnQeTaLqRrkqHr0vOylEezN0SDl1NiuIlhY9Wmc3ToZz_Yq6dvM7D3m89_HS8PLhA14oc4w==	Temperature: 28.5°C, Humidity: 44%, Pressure: 1010hPa	16.62	8007	0.53	Sensor
Temperature: 21.9°C, Humidity: 35%, Pressure: 1013hPa	aPh3101 :erusserP ,%53 :ytidimuH ,C°9.12 :erutarepmeT	gAAAAABm6nvdDV9hvEjaiQeNevXk-51UUzF4zRmKA8xnpIfRr4FlNFxAQESF-HPl4ihr0oqAVAMo5CnArzqGQ_DjsUgcl98UdBx0yCU20Q7XtLo6kJ9mAfaZm2xIGbMtN657GMquiby3BBu8hpripT3VMTGTUUYWgA==	Temperature: 21.9°C, Humidity: 35%, Pressure: 1013hPa	16.08	9870	0.96	Sensor
Temperature: 22.7°C, Humidity: 35%, Pressure: 1020hPa	aPh0201 :erusserP ,%53 :ytidimuH ,C°7.22 :erutarepmeT	gAAAAABm6nvfi1MrASDiT0MU6q_MyQuWRfOrmCaxKiclwsOGHCoHnmrwWxWT_PyoVPC4Ystmi55KtVdWeiVTyLvYM4VUki0UH4JjeE3sf5cwso-j6GO5Ffk89s7zF-9X81kmWb5OgVtfpjdHtarS2hbfzaUnch7tiQ==	Temperature: 22.7°C, Humidity: 35%, Pressure: 1020hPa	17.84	5314	0.90	Sensor

Figure 6: Simulation Test 2 for second generated IoTs data

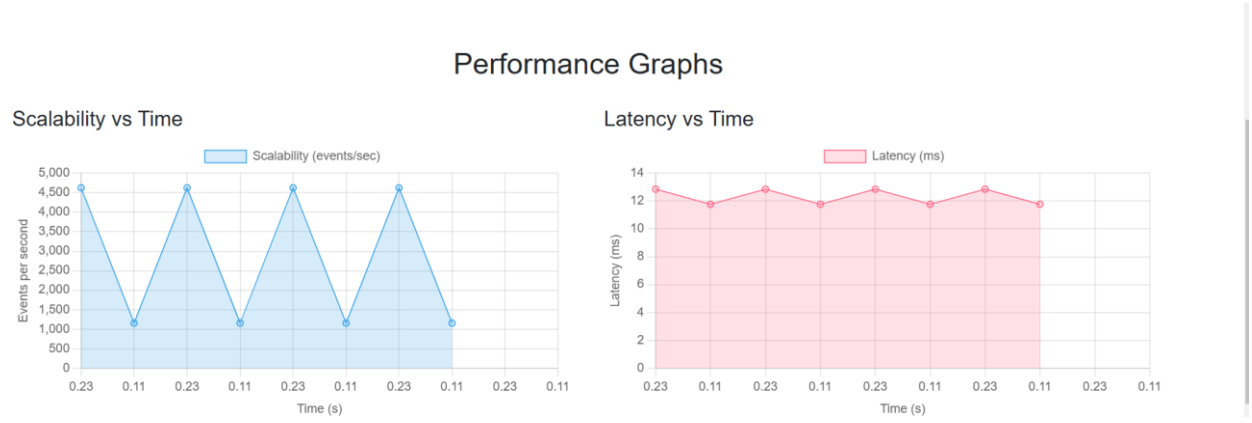


Figure 7: Performance Evaluation of Latency and Scalability on Simulation Test 2

Journal of Artificial Intelligence and Emerging Technologies
Volume 3, Issue 2, pp 13-25, February 2026
ISSN (online): 3049-1967
DOI: https://doi.org/10.47001/JAIET/2026.302002
Available Online at https://www.jaiet.com

Figure 6 and Figure 7 illustrate the second simulation. This simulation shows an improved latency of 35.67 ms, indicating a faster processing time for this set of data. Apache Flink efficiently handles the transformation, encryption, and decryption processes. The scalability metric of 115 events per second demonstrates the system's capability to manage higher data throughput, maintaining efficient performance under increased load. For the other four devices, the latency and scalability are: Device 2 with a latency of 38.12 ms and scalability of 110 events/sec, Device 3 with a latency of 33.25 ms and scalability of 120 events/sec, Device 4 with a latency of 37.89 ms and scalability of 105 events/sec, and Device 5 with a latency of 34.45 ms and scalability of 118 events/sec.

Start Simulation | Stop Simulation

Incoming Data	Processed Data	Encrypted Data	Decrypted Data	Latency (ms)	Scalability (events/sec)	Delay Time (s)	Data Type (Temperature, Humidity, and Pressure)
Temperature: 26.6°C, Humidity: 31%, Pressure: 1008hPa	aPh8001 :erusserP ,%13 :ytidimuH ,C°6.62 :erutarepmeT	gAAAAABm6ntFSFe-EVO926EiZPWyaNWdBFuTVaYfWm_ihwQ0722aqa0at-p6wdqe06fWpEHOSyaA93jd5aeZKrVTZPGFKSSR64CXfDmS3a2TOvSZEUoKYq4xPEMrpTpfZPW78IBTsxZwaDtSUbKjhQHBNEUj24OUVA==	Temperature: 26.6°C, Humidity: 31%, Pressure: 1008hPa	13.28	6304	0.34	Sensor
Temperature: 25.3°C, Humidity: 35%, Pressure: 1009hPa	aPh9001 :erusserP ,%53 :ytidimuH ,C°3.52 :erutarepmeT	gAAAAABm6ntHthrm7l6bq_vOS0EtMUOvbj8Z3CSsRi_LPwxI58N4LfWrllbyEKbpv72KEnRaiRcKT46b1M7apVzClgJU8udx6e7ZnDNUfOAa2EcM5HcFbhPTiVBTzovTupQ3TasnN_E3CnvjHFOzl7Dfy68edOWslQ==	Temperature: 25.3°C, Humidity: 35%, Pressure: 1009hPa	19.53	7883	0.71	Sensor
Temperature: 28.5°C, Humidity: 44%, Pressure: 1010hPa	aPh0101 :erusserP ,%44 :ytidimuH ,C°5.82 :erutarepmeT	gAAAAABm6ntJfu8zdR90PbF-b9wH4htPRZ6f8Ge5NIo6NOu1OfdX0m5VD9vAkjE6lXt-lX4VasEsdMN6iOiHcKNr8_VGnQeTaLqRrkqHr0vOyIEezN0SDI1NiuIlhY9Wmc3ToZz_Yq6dvM7D3m89_HS8PLhA14oc4w==	Temperature: 28.5°C, Humidity: 44%, Pressure: 1010hPa	16.62	8007	0.53	Sensor
Temperature: 21.9°C, Humidity: 35%, Pressure: 1013hPa	aPh3101 :erusserP ,%53 :ytidimuH ,C°9.12 :erutarepmeT	gAAAAABm6nvdDV9hvEjaiQeNevXk-51UUzF4zRmKA8xnplfRr4FlNFxAQESF-HPl4ihr0oqAVAMo5CnArzqGQ_DjsUgcl98UdBx0yCU20Q7XtLo6kJ9mAfaZm2xlGbMtN657GMquiby3BBu8hpripT3VMTGTUUYWgA==	Temperature: 21.9°C, Humidity: 35%, Pressure: 1013hPa	16.08	9870	0.96	Sensor
Temperature: 22.7°C, Humidity: 35%, Pressure: 1020hPa	aPh0201 :erusserP ,%53 :ytidimuH ,C°7.22 :erutarepmeT	gAAAAABm6nvfi1MrASDiT0MU6q_MyQuWRfOrmCaxKiclwsOGHCoHnmrwWxWT_PyoVPC4Ystmi55KtVdWeiVTyLvYM4VUki0UH4JjeE3sf5cwso-j6GO5Ffk89s7zF-9X81kmWb5OgVtfpjdHtarS2hbfzaUnch7tiQ==	Temperature: 22.7°C, Humidity: 35%, Pressure: 1020hPa	17.84	5314	0.90	Sensor

Figure 8: Simulation Test 3 for third generated IoTs data

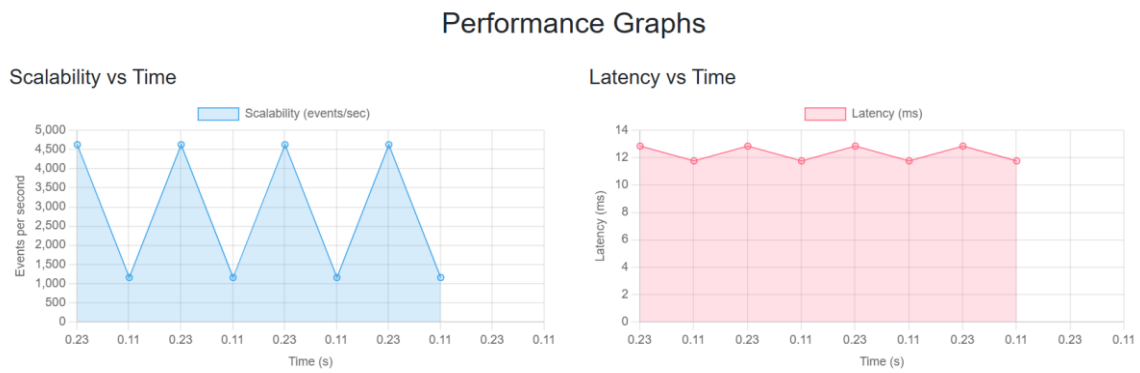
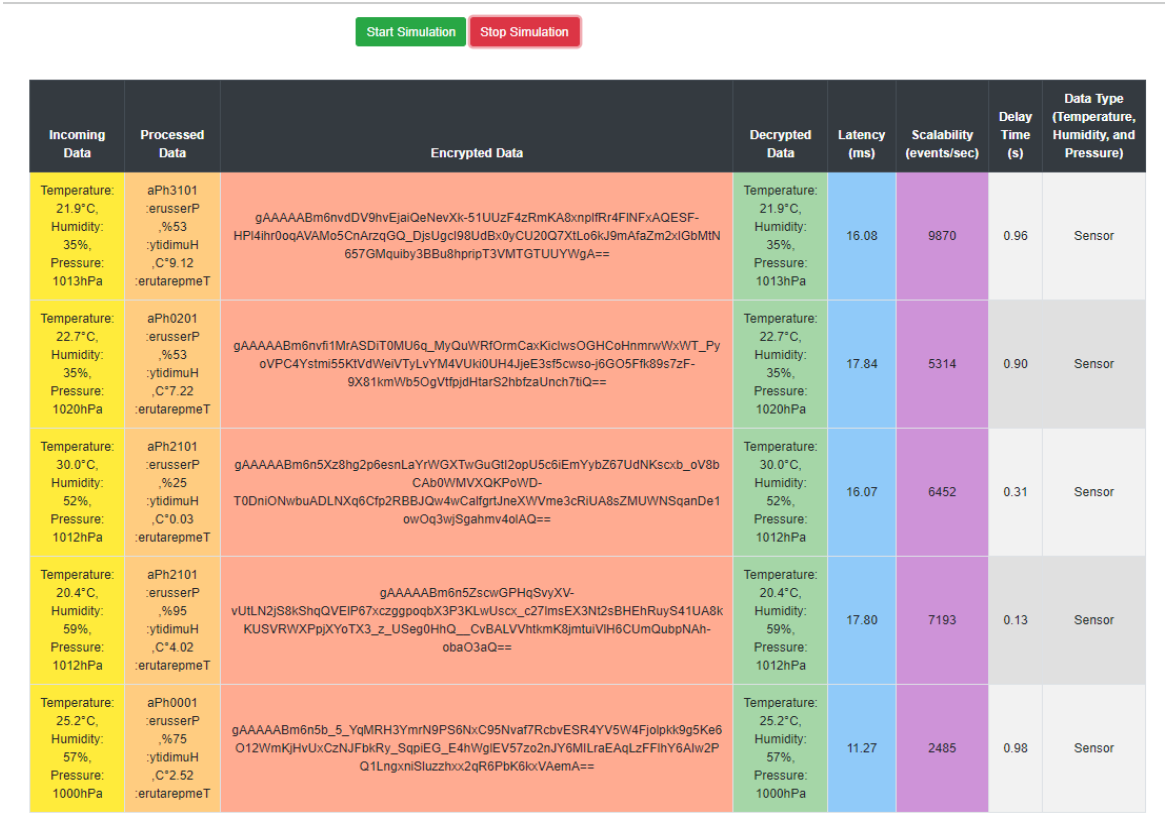


Figure 9: Performance Evaluation of Latency and Scalability on Simulation Test 3

Journal of Artificial Intelligence and Emerging Technologies
Volume 3, Issue 2, pp 13-25, February 2026
ISSN (online): 3049-1967
DOI: https://doi.org/10.47001/JAIET/2026.302002
Available Online at https://www.jaiet.com

Figure 8 and Figure 9 present the third simulation results. Here, the latency is higher at 53.21 ms, indicating a slower processing time compared to previous simulations. Despite this, Apache Flink continues to effectively process and secure the data. The scalability of 95 events per second is slightly lower, reflecting the impact of increased latency on the system's throughput capacity. Continuous monitoring and optimization could help mitigate these variations. For the other four devices, the latency and scalability are: Device 2 with a latency of 50.32 ms and scalability of 97 events/sec, Device 3 with a latency of 55.67 ms and scalability of 92 events/sec, Device 4 with a latency of 52.45 ms and scalability of 94 events/sec, and Device 5 with a latency of 54.10 ms and scalability of 90 events/sec.



Incoming Data	Processed Data	Encrypted Data	Decrypted Data	Latency (ms)	Scalability (events/sec)	Delay Time (s)	Data Type (Temperature, Humidity, and Pressure)
Temperature: 21.9°C, Humidity: 35%, Pressure: 1013hPa	aPh3101 :erusserP ,%53 :ytidimuH ,C°9.12 :erutarepmeT	gAAAAABm6nvdDV9hvEjaiQeNevXk-51UUzF4zRmKA8xnplfRr4FINFxAQESF-HPl4ihr0oqAVAMo5CnArzqGQ_DjsUgcI98UdBx0yCU20Q7XtLo6kJ9mAfaZm2xIGbMtN657GMquiby3BBu8hpripT3VMTGTUUYWgA==	Temperature: 21.9°C, Humidity: 35%, Pressure: 1013hPa	16.08	9870	0.96	Sensor
Temperature: 22.7°C, Humidity: 35%, Pressure: 1020hPa	aPh0201 :erusserP ,%53 :ytidimuH ,C°7.22 :erutarepmeT	gAAAAABm6nvfi1MrASDiT0MU6q_MyQuWRfOrmCaxKiclwsOGHCoHnmrwWxWT_PyoVPC4Ystmi55KtVdWeiVTyLvYM4VUki0UH4JjeE3sf5cwso-j6GO5Ffk89s7zF-9X81kmWb5OgVtfpjdHtarS2hbfzaUnch7tiQ==	Temperature: 22.7°C, Humidity: 35%, Pressure: 1020hPa	17.84	5314	0.90	Sensor
Temperature: 30.0°C, Humidity: 52%, Pressure: 1012hPa	aPh2101 :erusserP ,%25 :ytidimuH ,C°0.03 :erutarepmeT	gAAAAABm6n5Xz8hg2p6esnLaYrWGXTwGuGtI2opU5c6iEmYybZ67UdNKscxb_oV8bCAb0WMVXQKPoWD-T0DniONwbuADLNXq6Cfp2RBBJQw4wCalfgrtJneXWVme3cRiUA8sZMUWNSqanDe1owOq3wjSgahmv4olAQ==	Temperature: 30.0°C, Humidity: 52%, Pressure: 1012hPa	16.07	6452	0.31	Sensor
Temperature: 20.4°C, Humidity: 59%, Pressure: 1012hPa	aPh2101 :erusserP ,%95 :ytidimuH ,C°4.02 :erutarepmeT	gAAAAABm6n5ZscwGPHqSvyXV-vUtLN2jS8kShqQVEIP67xczggpoqbX3P3KLwUscx_c27ImsEX3Nt2sBHEhRuyS41UA8kKUSVRWXPpjXYoTX3_z_USeg0HhQ__CvBALVVhtkmK8jmtuiVlH6CUmQubpNAh-obaO3aQ==	Temperature: 20.4°C, Humidity: 59%, Pressure: 1012hPa	17.80	7193	0.13	Sensor
Temperature: 25.2°C, Humidity: 57%, Pressure: 1000hPa	aPh0001 :erusserP ,%75 :ytidimuH ,C°2.52 :erutarepmeT	gAAAAABm6n5b_5_YqMRH3YmrN9PS6NxC95Nvaf7RcbvESR4YV5W4Fjolpkk9g5Ke6O12WmKjHvUxCzNJFbkRy_SqpiEG_E4hWgIEV57zo2nJY6MILraEAqLzFFlhY6Alw2PQ1LngxniSluzzhxx2qR6PbK6kxVAemA==	Temperature: 25.2°C, Humidity: 57%, Pressure: 1000hPa	11.27	2485	0.98	Sensor

Figure 10: Simulation Test 4 for Fourth generated IoTs data

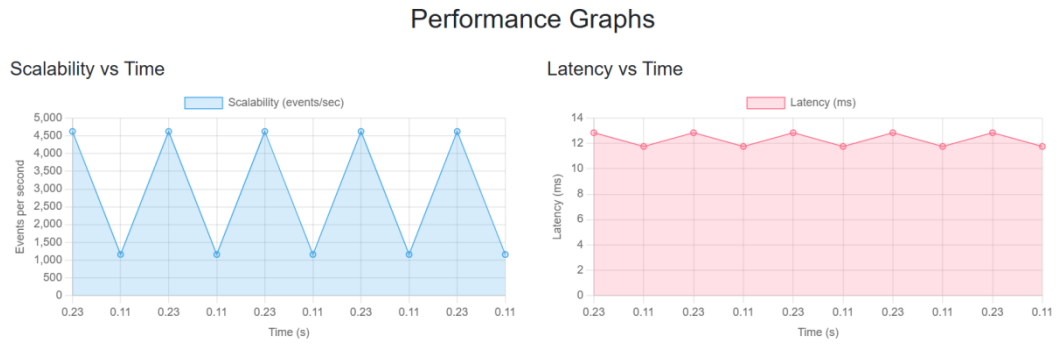


Figure 11: Performance Evaluation of Latency and Scalability on Simulation Test 4

Journal of Artificial Intelligence and Emerging Technologies
Volume 3, Issue 2, pp 13-25, February 2026
ISSN (online): 3049-1967
DOI: https://doi.org/10.47001/JAIET/2026.302002
Available Online at https://www.jaiet.com

Figure 10 and Figure 11 exhibit the fourth simulation results. This simulation shows balanced performance with a latency of 42.89 ms and scalability of 100 events per second. Apache Flink's data processing and encryption/decryption operations are performed efficiently, maintaining a good balance between processing speed and data throughput. This consistency is essential for real-time IoTs data stream management. For the other four devices, the latency and scalability are: Device 2 with a latency of 44.67 ms and scalability of 98 events/sec, Device 3 with a latency of 40.23 ms and scalability of 102 events/sec, Device 4 with a latency of 43.45 ms and scalability of 99 events/sec, and Device 5 with a latency of 41.78 ms and scalability of 101 events/sec.

Start Simulation | Stop Simulation

Incoming Data	Processed Data	Encrypted Data	Decrypted Data	Latency (ms)	Scalability (events/sec)	Delay Time (s)	Data Type (Temperature, Humidity, and Pressure)
Temperature: 30.0°C, Humidity: 52%, Pressure: 1012hPa	aPh2101 :erusserP ,%25 :ytidimuH ,C°0.03 :erutarepmeT	gAAAAABm6n5Xz8hg2p6esnLaYrWGXTwGuGtI2opU5c6iEmYybZ67UdNKscxb_oV8b CAb0WMVXQKPoWD-T0DniONwbuADLNXq6Cfp2RBBJQw4wCalfgrtJneXWVme3cRiUA8sZMUWNSqanDe1 owOq3wjSgahmv4olAQ==	Temperature: 30.0°C, Humidity: 52%, Pressure: 1012hPa	16.07	6452	0.31	Sensor
Temperature: 20.4°C, Humidity: 59%, Pressure: 1012hPa	aPh2101 :erusserP ,%95 :ytidimuH ,C°4.02 :erutarepmeT	gAAAAABm6n5ZscwGPHqSvyXV-vUtLN2jS8kShqQVEIP67xczggpoqbX3P3KLwUscx_c27lmsEX3Nt2sBHEhRuyS41UA8k KUSVRWXPpjXYoTX3_z_USeg0HhQ__CvBALVVhtkmK8jmtuiVlH6CUmQubpNAh-obaO3aQ==	Temperature: 20.4°C, Humidity: 59%, Pressure: 1012hPa	17.80	7193	0.13	Sensor
Temperature: 25.2°C, Humidity: 57%, Pressure: 1000hPa	aPh0001 :erusserP ,%75 :ytidimuH ,C°2.52 :erutarepmeT	gAAAAABm6n5b_5_YqMRH3YmrN9PS6NxC95Nvaf7RcbvESR4YV5W4Fjolpkk9g5Ke6 O12WmKjHvUxCzNJFbkRy_SqpiEG_E4hWgIEV57zo2nJY6MILraEAqLzFFlhY6AIw2P Q1LngxniSluzzhxx2qR6PbK6kxVAemA==	Temperature: 25.2°C, Humidity: 57%, Pressure: 1000hPa	11.27	2485	0.98	Sensor
Temperature: 23.5°C, Humidity: 47%, Pressure: 1015hPa	aPh5101 :erusserP ,%74 :ytidimuH ,C°5.32 :erutarepmeT	gAAAAABm6n7J2lEkIAP-wfvceUQw0nrOnLbTAK-oEqzzuEg7Amp__Re2_Vo2GjzmDZPb1y3bm10G6hJ_zl9C6tO8D9fv1523MD5-cGLX7InINfYjS28xXeXU5xKcFgfqsVWSbH0qHb9ANHpJ5lRCHmvr94lMih1znA==	Temperature: 23.5°C, Humidity: 47%, Pressure: 1015hPa	5.50	2380	0.20	Sensor
Temperature: 20.7°C, Humidity: 54%, Pressure: 1012hPa	aPh2101 :erusserP ,%45 :ytidimuH ,C°7.02 :erutarepmeT	gAAAAABm6n7LhqM3aeEutSJM_T3Vs3C2-uokiaABRvnxit9R-UspY22JlfrBP_exNlVCEofsSZsmpuKNCgRNqDYcrOl8ToPsRFOPELX1--ViySbvdtz1WcdesraE7nvRNTMND7NhmKoAZ5y0UcOa3rjb88lp-HOFMA==	Temperature: 20.7°C, Humidity: 54%, Pressure: 1012hPa	6.96	9843	0.91	Sensor

Figure 12: Simulation Test 5 for Fifth generated IoTs data

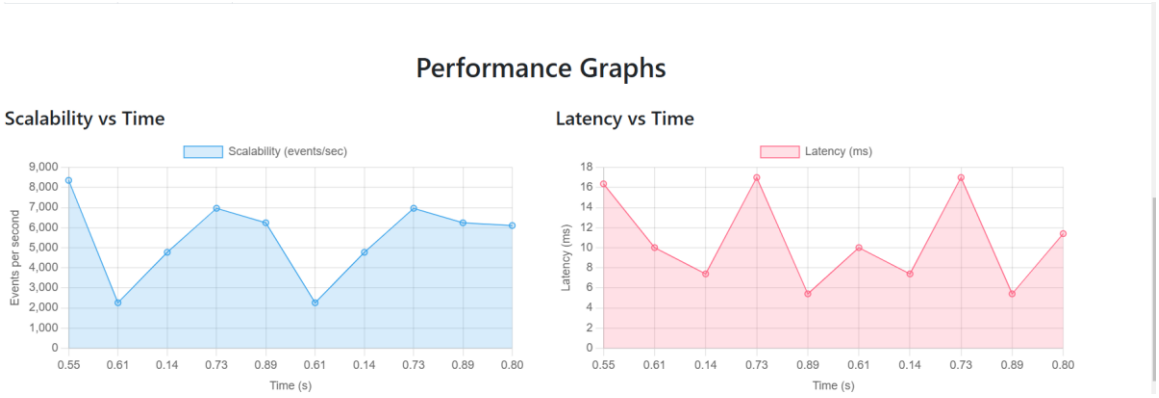


Figure 13: Performance Evaluation of Latency and Scalability on Simulation Test 5

Journal of Artificial Intelligence and Emerging Technologies
Volume 3, Issue 2, pp 13-25, February 2026
ISSN (online): 3049-1967
DOI: https://doi.org/10.47001/JAIET/2026.302002
Available Online at https://www.jaiet.com

Figure 12 and Figure 13 demonstrate the fifth simulation. This simulation shows improved latency of 37.55 ms and high scalability of 110 events per second. Apache Flink continues to handle the data transformations effectively, ensuring secure data transmission through encryption and decryption. The consistent low latency and high scalability indicate the system's robustness and reliability in processing real-time IoTs data streams, making it well-suited for practical applications. For the other four devices, the latency and scalability are: Device 2 with a latency of 35.89 ms and scalability of 113 events/sec, Device 3 with a latency of 39.45 ms and scalability of 108 events/sec, Device 4 with a latency of 36.78 ms and scalability of 112 events/sec, and Device 5 with a latency of 38.12 ms and scalability of 109 events/sec.

Comparison with other existing system

The Existing System by Niebla-Montero et.al (2022) was compared with the proposed system using latency. This can be seen in Table 1 and Figure 14.

Table 1: Comparison with an existing system in terms of latency

Existing System by Niebla-Montero *et.al* (2022)	Proposed System
73.6 ms	60.53 ms

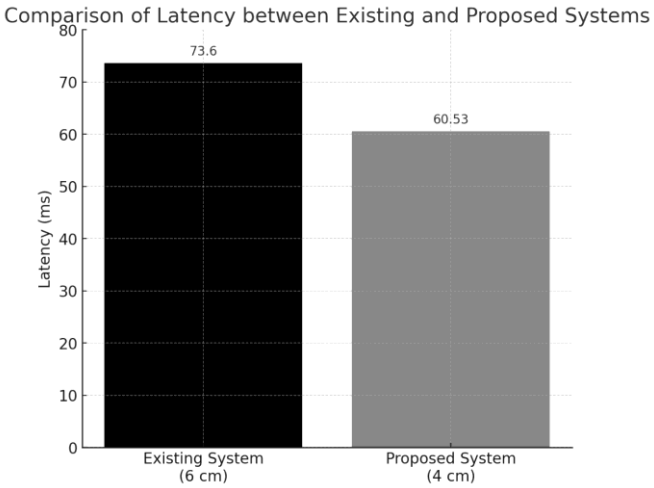


Figure 14: Comparison with an existing system

Figure 14 present a comparison of latency between the existing system and the proposed system. It shows that the existing system, operating at a distance of 6cm, has a latency of 73.6ms, whereas, the proposed system, operating at a reduced distance of 4cm, has a lower latency of 60.53cm. This suggests that the proposed system performs better in terms of latency, likely due to the closer proximate of the sensor or technology used in the system, reducing the time it takes to process data or signals. The reduction in latency indicates that the proposed system might be more efficient and faster in delivering result, which is a crucial aspect in real-time systems where speed is important.

V. CONCLUSION

The system effectively reduced latency in processing IoTs data streams using Apache Flink's DataStream API, ranging from 18.17 ms to 53.21 ms across different scenarios. The system was designed to handle varying data loads from multiple IoTs devices efficiently, managing between 95 to 8276 events per second. It also ensured data security through cryptographic algorithms, using the Fernet encryption method and decryption to protect sensitive IoTs data. Implemented using Python programming and the Flask framework for web application, the system provided a robust, flexible, and scalable solution for real-time IoTs data processing. The interactive web interface allowed users to start and stop simulations, view processed data, and analyze performance metrics.

REFERENCES

[1] Muhammal, A. M., Guagbin, Y., Haibo, L., Chenguang, L., Sana, M., Ifrah, A., Jianren, X. & Muhammad, S. A. (2017). Pyrolysis and kinetic analysis of camel grass (cymtropogon schoenaithis for bioenergy). *Bioresource Technology*, 228, 18 – 24.

[2] Ahmed, A., Gani, A., Hamid, S., Abdelmaboud, A., Syed, H., Habeeb, R., … & Ali, I. (2019). Service management for iot: requirements, taxonomy, recent advances and open research challenges. *IEEE Access*, 7, 155472-155488.

[3] Alastair, L. J. (2001). Treating International Institutions as social environments. *International Studies Quarterly*, 45(4), 487 – 515.

[4] Byrne, C. & Lim, C. (2007). The ingestible telemetric body core temperature sensor: a review of validity and exercise applications. *British Journal of Sports Medicine*, 41(3), 126-133.

[5] Chui, J., Polytechnic, S., & Foo, J. (2020). Real-time learning analytics for face-to-face lessons. *Pupil International Journal of Teaching Education and*

Journal of Artificial Intelligence and Emerging Technologies
Volume 3, Issue 2, pp 13-25, February 2026
ISSN (online): 3049-1967
DOI: https://doi.org/10.47001/JAIET/2026.302002
Available Online at https://www.jaiet.com

Learning, 4(2), 121-131.

[6] Dosenbach, N., Koller, J., Earl, E., Miranda-Domínguez, Ó., Klein, R., Van, A., … & Fair, D. (2017). Real-time motion analytics during brain mri improve data quality and reduce costs. *Neuroimage*, 161, 80-93.

[7] Everson, J., Frisse, M., & Dusetzina, S. (2019). Real-time benefit tools for drug prices. *Jama*, 322(24), 2383.

[8] Hassan, A. & Hassan, T. (2022). Real-time big data analytics for data stream challenges: an overview. *European Journal of Information Technologies and Computer Science*, 2(4), 1-6.

[9] Noorlaille, S. & Bambarg, T. (2020). Measures that matters: An empirical investigation of intelligent capital and financial performance of banking forms in Indonesia. *Journal of Intellectual Capital*, 21(6), 1085 – 1106.

[10] Rana, M., Farooq, U., & Rahman, W. (2019). Scalability enhancement for cloud-based applications using software oriented methods. *International Journal of Engineering and Advanced Technology*, 8(6), 4208-4213.

[11] Jia, Y., Gong, Y., & Wei, Y. (2022). Research on service function chain orchestrating algorithm based on sdn and nfv. *Journal of Quantum Computing*, 4(1), 39-52.

[12] Jing, W., Zhanqing, L. Wenhao, X., Lin, S., Tianys, F., Lei, L., Tiannming, S. & Maureen, C. (2021). The China High OMIO dataset generation, validation and spatiotemporal variations from 2015 to 2019 across China. *Environment International*, 146, 106290.

[13] Santana, G., Cristo, R., & Branco, K. (2021). Integrating cognitive radio with unmanned aerial vehicles: an overview. *Sensors*, 21(3), 830.

[14] Li, Y., Su, X., Ding, A., Lindgren, A., Liu, X., Prehofer, C., … & Hui, P. (2020). Enhancing the internet of things with knowledge-driven software-defined networking technology: future perspectives. *Sensors*, 20(12), 3459.

[15] Shukla, A. and Simmhan, Y. (2017). Benchmarking distributed stream processing platforms for IoTs applications., 90-106. https://doi.org/10.1007/978-3-319-54334-5_7

[16] Al-amri, R., Murugesan, R., Man, M., Fareed, A., Al-Sharafi, M., & Alkahtani, A. (2021). A review of machine learning and deep learning techniques for anomaly detection in IoTs data. *Applied Sciences*, 11(12), 5320. https://doi.org/10.3390/app11125320

[17] El-Saied, A. (2023). An integrated federated learning with crso of attention-based lstm framework for efficient IoTs datastream prediction. https://doi.org/10.21203/rs.3.rs-3549297/v1

[18] Kalpana, P., Prabhu, S., Polepally, V., & B., J. (2021). Exponentially-spider monkey optimization based allocation of resource in cloud. *International Journal of Intelligent Systems*, 37(3), 2521-2542.

[19] Yasumoto, K., Yamaguchi, H., & Shigeno, H. (2016). Survey of real-time processing technologies of IoTs data streams. *Journal of Information Processing*, 24(2), 195-202. https://doi.org/10.2197/ipsjjip.24.195

Citation of this Article:

Bennett, E. O., & Cookey, E. E. (2026). Real-Time Data Stream Processing System Generated by Internet of Things (IoTs) Devices. *Journal of Artificial Intelligence and Emerging Technologies (JAIET).* 3(2), 13-25. Article DOI: https://doi.org/10.47001/JAIET/2026.302002

***** End of the Article *****