International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 05, May 2026)

An Adaptive Random Mobility Clustering Approach for Data Quality Enhancement in Fog-Based Crowd Sensing

A. Shajin Nargunam

Noorul Islam Centre for Higher Education, Kumaracoil, Thuckalay, Tamilnadu, India

***Abstract*— Crowd sensing uses mobile devices as distributed sensors, but data quality often depends on how users move. To address this, we combine microservices, random mobility, and clustering algorithms in a fog computing environment. Microservices enable scalable and flexible deployment of sensing tasks. Random mobility helps generate data that is both diverse and less biased.**

We use a clustering algorithm to group data by spatial and temporal proximity, forming clusters that reflect real-world patterns. We evaluate the approach through simulations in a realistic scenario. The method improves data quality and resource efficiency compared to current techniques. These results support the development of crowd-sensing systems tailored for fog computing.

***Index Terms*— Cloud Computing, Clustering, Crowd Sensing, Microservices, Random Mobility**

I. Introduction

Crowd sensing has fundamentally transformed how we gather large-scale data by tapping into the everyday mobile devices that people carry with them—essentially turning the public into a vast network of sensors. Imagine thousands of commuters using their smartphones to help track air quality in a city, or drivers' GPS data crowd-sourced to ease traffic congestion. This shift has unlocked remarkable opportunities across domains like environmental monitoring, smarter traffic flow, and public safety systems. Yet, a key hurdle remains: the reliability and diversity of the collected data. The core issue stems from user movement patterns—if most participants follow the same routes or routines, the system ends up with a lopsided dataset, much like polling only one neighborhood to gauge the mood of an entire city. This bias can result in incomplete or unrepresentative information, ultimately limiting the effectiveness of crowd sensing applications.

Recognizing this challenge, researchers have sought ways to motivate people to move around more unpredictably and cover a broader range of locations. Think of it as encouraging festival-goers to explore every corner of the venue, ensuring no area is overlooked.

Incentive schemes, however, often struggle to scale up, adapt to different user behaviors, or respond to changing environments. To bridge these gaps, this paper introduces a fresh strategy: we blend microservices, random movement encouragement, and intelligent clustering—all within a fog computing framework—to create a more versatile and robust crowd sensing system.

At the heart of our solution is a microservices-based architecture, which acts much like a collection of specialized teams—each handling a different aspect of the overall task. By dividing sensing activities into smaller, standalone microservices, we can dynamically assign and coordinate them according to available resources and how users are moving throughout the area. This modular approach allows the system to quickly adapt to changing conditions or spikes in activity, much like a well-organized event staff that can redeploy to busier locations as crowds shift.

To further enrich the data, we introduce an element of randomness in user movement. By offering incentives—such as rewards or game-like features—we motivate participants to venture off their usual paths, much like a scavenger hunt encouraging exploration of lesser-known areas. This strategy helps ensure a broader and more balanced coverage, resulting in data that better reflects the true diversity of the monitored environment and reducing the risk of blind spots or biases in the dataset.

To make sense of the wealth of data collected, we apply a clustering algorithm that organizes information based on where and when it was gathered. With these meaningful clusters, it becomes easier for applications to extract actionable insights—allowing city planners, for instance, to quickly identify emerging traffic jams or environmental hotspots.

We validated our approach with extensive simulations grounded in practical scenarios. By fostering both flexibility and scalability, our solution paves the way for the next generation of crowd sensing platforms built for fog computing settings. This advancement opens the door for a broader range of data-driven services and applications, from smarter urban planning to more responsive public safety initiatives.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 05, May 2026)

II. Literature Review

Mobile crowdsensing (MCS) has fundamentally changed how we collect large-scale data by turning everyday mobile devices into a distributed network of sensors. Imagine city dwellers using their smartphones to monitor air quality or report traffic jams in real time; the collective power of these devices opens up possibilities across many fields, from environmental monitoring to urban planning. Over the years, researchers have mapped MCS's evolution—from its early days when humans were the primary sensing agents to the present era, where sensors and the Internet of Things (IoT) work hand in hand to collect and share information seamlessly.

Guo et al. [10] laid the groundwork for understanding how MCS systems operate, breaking them down into essential components, including task division, participant recruitment, and data collection. Abualsaud et al. [11] highlighted the challenges of scaling these systems, especially for tasks such as recognizing human activities, where balancing high-quality data with limited device resources is tricky. The concept has found a natural home in smart cities: Cardone et al. [13] introduced platforms that encourage people to contribute local data, almost like crowdsourcing city insights, while Farkas et al. [14] showed how public transport users can become real-time information sources. Guo et al. [15] expanded on this with FlierMeet—a system that lets people share public information across different spaces—demonstrating the real-world value of MCS in bustling urban environments.

Fog computing has emerged as a game-changer for mobile crowdsensing, offering solutions to the bottlenecks of traditional cloud-centric systems—namely, latency, bandwidth constraints, and the challenge of scaling to support vast numbers of devices. Picture fog computing as a network of local "mini-clouds" or neighborhood data hubs that process and filter information close to where it is generated, rather than sending everything to a distant data center. This local processing helps speed up response times, reduces the load on central servers, and makes the whole system more resilient.

Belli et al. [1] provided a sweeping overview of how fog computing and MCS intersect, mapping out the different ways tasks can be distributed, where data should be processed, and how resources can be managed efficiently. Their work acts as a field guide for designing and classifying fog-based MCS deployments. Alamri et al. [2] took this further by showing how fog nodes—think of them as local information checkpoints—can sift through streams of crowd-collected data, verifying what's trustworthy before forwarding it to the cloud.

Khairosheva et al. [3] tackled the problem of secure and reliable data transfer across these networks, introducing lightweight cryptography that's practical even for fog nodes with minimal resources at the network's edge. Meanwhile, El Hafyani et al. [4] showcased the power of modular, microservices-based architectures for crowdsensing data: by breaking down data handling into independent, automated steps, they made it easier to adapt, upgrade, and maintain these complex workflows as technology and requirements evolve.

Guaranteeing the reliability of data collected through mobile crowdsensing (MCS) is much like piecing together the truth from a multitude of voices—some more reliable than others. The sheer diversity of contributors and their devices means that not all data can be trusted equally. To navigate this, researchers have devised innovative strategies to sort fact from fiction amid the digital noise.

For example, Restuccia et al. [5] developed the FIRST framework, which acts as a quality control supervisor, constantly tuning how and when data is gathered to achieve the best possible results while respecting resource limits such as battery life and bandwidth. Their method bakes quality metrics into the very process of assigning sensing tasks, ensuring that only the most useful data gets through. Luo et al. [6] took another approach, using cross-validation—think of it as a system of checks and balances, where overlapping reports from different users are compared to spot outliers or suspicious readings. Crucially, this method works even when there's no "ground truth" to compare against, making it ideal for decentralized fog computing environments where central oversight is tough.

Other researchers have tackled the privacy puzzle. Xu et al. [7] designed cryptographic tools that enable the system to discover the most likely truth without exposing individual contributions—much like tallying anonymous ballots to reach consensus. Yang et al. [8] connected the dots between data quality and participant motivation, creating a system that rewards users not just for contributing, but for providing accurate, verifiable information. Meanwhile, Jensen [9] drew intriguing comparisons between age-old methods of knowledge validation—such as how pastoralists confirm information through community consensus—and today's crowd-powered platforms like Mechanical Turk, highlighting the timeless value of human-in-the-loop quality assurance in technology-driven environments.

Privacy and trust are closely intertwined in MCS systems, often pulling in opposite directions. On the one hand, participants want assurance that their personal data is kept confidential; on the other hand, systems need to be able to validate and trust the information being shared.

He et al. [16] highlighted this tug-of-war, noting that privacy safeguards can sometimes make it harder to assess data reliability. However, their research suggests that fog computing can help strike a balance. By processing and anonymizing data closer to where it is collected—rather than sending everything to a central hub—fog-based systems can better protect user identities while still maintaining mechanisms to assess trustworthiness.

Building on this, Kim et al. [17] showed how sorting participant reports by topic (similar to organizing library books by subject) can make it easier to spot unusual or inconsistent data, improving both data management and the ability to identify anomalies within collaborative sensing projects.

Despite these advances, several key challenges remain unaddressed in the field. Most notably, integrating real-time truth discovery into fog-based streaming systems remains a significant gap—most current solutions are built for slower, batch-style processing in the cloud rather than the fast, on-the-fly needs of fog architectures. Additionally, while secure communication frameworks between fog and cloud layers have been proposed, they're often developed in isolation from strategies that ensure the right data is collected from the right sources at the right time. Lastly, the automation of data pipelines for MCS applications has largely focused on building robust structures rather than making them smart enough to adapt to changing data quality demands.

This paper aims to bridge these gaps by introducing a unified fog-based MCS framework. Our approach seamlessly combines real-time group-based truth discovery, secure communication protocols, and adaptive, quality-driven data processing—delivering a flexible architecture ready for the next generation of scalable, trustworthy mobile crowdsensing systems.

III. Microservices-Based Architecture for Crowd Sensing

Our approach is built on harnessing the strengths of microservices and fog computing to deliver crowd-sensing systems that are not only scalable and flexible but also efficient with resources. Picture this setup as a series of independent, specialized teams (microservices), each responsible for a different aspect of the larger mission. These teams can be deployed, scaled, or modified without impacting the rest, thanks to modern containerization tools like Docker or Kubernetes—similar to adding or swapping out teams at a large event as the crowd's needs change.

To further enhance the diversity and quality of the data collected, we incorporate random mobility and clustering algorithms into our mobile application.

This is akin to encouraging event participants to explore different areas, ensuring the information gathered covers a wide, representative range rather than clustering in just a few spots.

Crowd sensing, at its core, is about mobilizing everyday people—using their smartphones—to help collect and interpret data for applications ranging from monitoring air quality to improving traffic flow or enhancing public safety. The process usually involves users gathering data and sending it to a central system for analysis. However, orchestrating these projects is no small feat. Each initiative may involve several moving parts: real-time data collection, processing, analytics, and more. The sheer variety and scale of crowd sensing campaigns mean that rigid, one-size-fits-all solutions often fall short. What is needed is an adaptable system that can be tailored to the unique requirements and scale of each new challenge.

A microservices-based architecture offers a practical and agile way to manage the complex, evolving landscape of crowd sensing. By breaking down large tasks into manageable, independent modules, this approach allows systems to respond rapidly to changing demands, much like a well-coordinated team that can quickly reassign roles or bring in new members as circumstances shift.

Crowd sensing, while offering enormous potential, brings its own set of challenges—chief among them, handling the massive influx of data generated by thousands or even millions of participants. Imagine trying to manage a bustling city's worth of information with a single, rigid system; traditional monolithic architectures often struggle under such pressure, much like a single-lane bridge overwhelmed by rush-hour traffic. These systems are difficult to scale, adapt, and maintain as demands grow and change.

In contrast, microservices-based architectures present a more agile and efficient approach. Picture microservices as a fleet of specialized vehicles, each handling a specific type of cargo; they can be developed, deployed, and scaled independently, making it much easier to manage fluctuating data volumes. This modular design is particularly well-suited to the dynamic, distributed nature of crowd-sensing applications, where adaptability and scalability are crucial for success.

In simple terms, a microservices-based architecture breaks down a complex application into a collection of small, independent services—each with a clearly defined role. These services interact through standard interfaces (APIs), much like different teams in an organization collaborating via established communication channels. This structure allows each service to be updated, scaled, or replaced on its own, without disrupting the entire system.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 05, May 2026)

For crowd sensing, this means that each key activity—such as data collection, real-time processing, or analytics—can be handled by its own microservice. Think of it like a production line where each workstation can speed up or slow down as needed, without throwing the whole operation off balance. If, for instance, there is a sudden surge in incoming data, only the data collection service needs to scale up, while other services continue operating smoothly. This independence makes the entire system more responsive and resilient to the unpredictable nature of real-world data flows.

Another strength of microservices is their ability to integrate easily with other platforms—be it cloud or fog computing environments. Because each service communicates through standard APIs, connecting new technologies or expanding into new environments is as straightforward as plugging in a new module to an existing setup. This flexibility streamlines deployment and management, allowing crowd sensing systems to evolve alongside technological advances or changing operational requirements.

Benefits of microFlexibility: Microservices-based architecture allows for rapid customization and adaptation, making it easy to tailor crowd sensing solutions to shifting needs or new challenges as they arise. Scalability: Each microservice can be scaled independently, ensuring resources are used efficiently and the system can accommodate growth or sudden spikes in data without missing a beat.

Resilience: By isolating functions into separate services, the system becomes more robust—if one service encounters an issue, others continue operating smoothly, enhancing overall fault tolerance and reliability.h Reusability: Individual microservices can be repurposed across different projects or applications, promoting faster development cycles and reducing duplication of effort. Microservices can be reused in different applications or tasks, enabling efficient development and reuse of software components.

Implementation of microservices-based architecture for crowd-sensing tasks:

To implement a microservices-based architecture for crowd-sensing tasks, the following steps need to be followed:

- Identify the components involved in the crowd-sensing task, including data collection, processing, and analysis.
- Design and implement each component as a separate microservice, with well-defined APIs for communication with other services.
- Deploy each microservice in a containerized environment, such as Docker or Kubernetes, enabling efficient scaling and management.
- Implement a service discovery mechanism, such as Consul or Etcd, to enable dynamic discovery and communication between microservices.
- Implement a monitoring and logging mechanism, such as Prometheus or Fluentd, to enable efficient monitoring and management of the microservices.
- Implement a load-balancing mechanism, such as Nginx or HAProxy, to efficiently distribute traffic among the microservices.

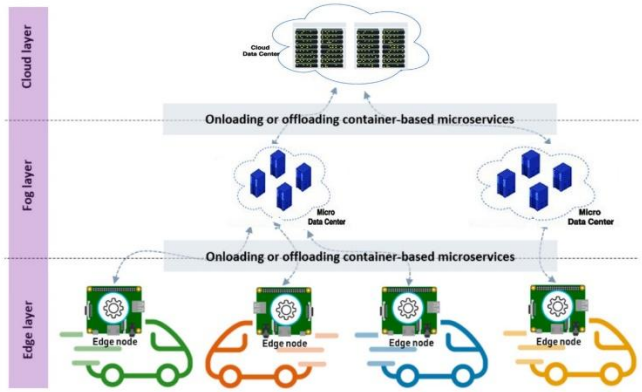

Figure 1 Microservices-Based Architecture

Let N be the number of microservices involved in the crowd sensing task, and let M be the number of users participating in the task. Let T_i be the time taken by the i-th microservice to process a single data point, and let D_j be the amount of data collected by the j-th user.

The total time taken by the crowd sensing task can be modeled as:

$$T_{total} = max(T_i) + sum(D_j * T_i)$$

where $\max(T_i)$ is the time taken by the slowest microservice, and $\text{sum}(D_j * T_i)$ is the time taken to process all the data points collected by the users.

The scalability of the microservices-based architecture can be modeled as:

$$S = (T_i * N) / T_{total}$$

where S is the scalability of the architecture, defined as the ratio of the total processing time to the time taken by a single microservice to process a single data point. The scalability of the architecture is a measure of how efficiently the architecture can handle an increase in the number of users or data points.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 05, May 2026)

The resilience of the microservices-based architecture can be modeled as:

$$R = 1 - P_{fail}$$

where P_{fail} is the probability of failure of the architecture, defined as the probability that one or more microservices fail during the task. The resilience of the architecture is a measure of its ability to recover from failures and continue processing the task.

The efficiency of the microservices-based architecture can be modeled as:

$$E = sum(T_i) / T_{total}$$

where E is the efficiency of the architecture, defined as the ratio of the time taken by all the microservices to process the data points to the total time taken by the task. The efficiency of the architecture is a measure of how well the microservices are utilized and how efficiently they process the data points.

A microservices-based architecture for crowd sensing can be divided into three main layers: the device layer, the service layer, and the application layer.

Device Layer:

The device layer serves as the foundation for data collection in a crowd-sensing system, acting much like the front-line reporters of a news network. Here, a wide range of participant devices—such as smartphones, wearables, or even connected vehicles—gather real-time information from the world around them. To make this process seamless and secure, each device runs a lightweight client that handles communication with the system's microservices.

Think of these microservices as the logistical coordinators behind the scenes: they manage device registration and user authentication, ensuring only trusted devices participate, and oversee the ongoing flow of data. Whether the information comes from sensors measuring air quality, GPS tracking for location, or even social media check-ins, the device layer ensures that all collected data is transmitted securely and efficiently. Special attention is paid to preserving battery life and minimizing data usage, so participants can continue contributing without worrying about draining their devices or exceeding their data plans. In essence, the device layer serves as the crucial bridge between everyday users and the broader crowdsensing network, ensuring that data is both trustworthy and resource-efficient from the very start.

The Device Layer in a crowd-sensing architecture is responsible for capturing data from various devices used by participants in the network. It ensures secure, reliable data transmission while optimizing device resources, such as battery life and network bandwidth.

To explain the function of the Device Layer in crowd sensing, we can consider a mathematical model that represents its operations.

Let's define the following variables:

- D: Set of devices participating in crowd sensing.
- t: Time index representing discrete time intervals.
- $D_i(t)$: Data collected by device i at time t.
- $X_i(t)$: Location coordinates of device i at time t.
- $P_i(t)$: Power consumption of device i at time t.

The Device Layer performs the following functions:

1. *Data Acquisition*: Devices capture data from their sensors and collect information relevant to the crowd sensing application. This data can include environmental measurements, audio, images, or any other type of sensory data. The mathematical model for data acquisition can be represented as:

$$D_i(t) = \text{SensorFunction}(x_i(t))$$

Here, SensorFunction represents the specific function or algorithm employed by the device to collect data based on its location coordinates.

2. *Data Transmission:* The collected data needs to be transmitted securely and reliably to the Service Layer for further processing. The mathematical model for data transmission can be represented as:

$$\text{TransmissionFunction}(d_i(t)) = \text{DataTransmitted}$$

The TransmissionFunction represents the mechanisms employed by the Device Layer to ensure the secure and efficient transmission of data while considering factors such as available network bandwidth, signal strength, and data encryption.

3. *Resource Optimization:* The Device Layer aims to optimize device resources, such as battery life and network bandwidth, while performing data acquisition and transmission. This optimization ensures that devices can operate for extended periods without draining their batteries and utilize network resources efficiently. The mathematical model for resource optimization can be represented as:

$$P_i(t) = \text{ResourceOptimization}(x_i(t), d_i(t))$$

ResourceOptimization represents the algorithms or strategies employed by the Device Layer to manage power consumption based on device location and collected data. This could involve adjusting transmission power, data compression techniques, or adaptive sampling rates to conserve energy and optimize network utilization.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 05, May 2026)

Overall, the Device Layer plays a crucial role in crowd sensing by efficiently capturing data from devices, ensuring secure data transmission, and optimizing device resources. The mathematical model provided here represents a simplified representation of these functions and can be further refined and customized based on the specific requirements and characteristics of the crowd sensing application and the devices involved.

Service Layer:

The service layer acts as an intermediary between the device layer and the application layer. It consists of a set of microservices that process, analyze, and store the incoming data from the devices. The microservices in the service layer perform tasks such as data preprocessing, feature extraction, data fusion, and quality assessment. These microservices can be independently deployed and scaled to handle the increasing volume and complexity of crowd sensing data. Additionally, the service layer provides APIs and interfaces to enable seamless integration with external services and applications.

The Service Layer in a crowd sensing architecture is responsible for processing and aggregating the data collected from the devices in order to derive meaningful insights and provide valuable services to users or applications. It performs various functions such as data fusion, data analysis, and service provisioning. To explain the function of the Service Layer in crowd sensing, we can consider a mathematical model that represents its operations.

Let's define the following variables:

- D: Set of devices participating in crowd sensing.
- t: Time index representing discrete time intervals.
- $D_i(t)$: Data collected by device i at time t.
- S: Set of services provided by the Service Layer.
- S_j: Service j provided by the Service Layer.
- F_j: Function representing the processing and analysis performed by service j.

The Service Layer performs the following functions:

1. *Data Fusion:* The Service Layer receives data from multiple devices and combines it to generate a unified and consistent view of the sensed phenomenon. Data fusion techniques, such as averaging, filtering, or statistical methods, are employed to aggregate and merge the data. The mathematical model for data fusion can be represented as:

$$\text{DataFusion}(d_1(t), d_2(t), ..., d_n(t)) = \text{FusedData}$$

Here, DataFusion represents the specific fusion algorithm or technique used to combine the data from multiple devices into a single representation (FusedData).

2. *Data Analysis:* The Service Layer analyzes the fused data to extract meaningful insights, patterns, or trends. Various analytical techniques, such as machine learning algorithms, statistical analysis, or data mining approaches, can be applied to the fused data. The mathematical model for data analysis can be represented as:

$$F_j(\text{FusedData}) = \text{AnalysisResult}$$

The function fj represents the specific analysis or processing performed by a particular service (s_j) on the fused data (FusedData) to generate an AnalysisResult. This can include tasks like anomaly detection, pattern recognition, classification, or prediction based on the specific objectives of the crowd sensing application.

3. *Service Provisioning:* The Service Layer provides various services based on the analyzed data to meet the requirements of users or applications. These services can range from real-time feedback, decision support, visualization, or any other value-added functionality. The mathematical model for service provisioning can be represented as:

$$S_j(\text{AnalysisResult}) = \text{ServiceOutput}$$

The service s_j takes the AnalysisResult generated from the data analysis stage and produces a ServiceOutput, which can be in the form of visualizations, alerts, reports, or any other relevant output format specific to the provided service.

Overall, the Service Layer plays a crucial role in crowd sensing by performing data fusion, data analysis, and service provisioning based on the collected data. The mathematical model provided here represents a simplified representation of these functions and can be further refined and customized based on the specific requirements and characteristics of the crowd sensing application and the services provided by the Service Layer.

Application Layer:

The application layer encompasses the components that utilize the processed data to provide value-added services and insights to end-users. This layer consists of microservices that offer functionalities such as data visualization, context-aware recommendations, real-time notifications, and decision support. The microservices in the application layer leverage the processed data from the service layer to generate meaningful and actionable information. The modular nature of microservices enables easy customization and adaptation of the applications to cater to specific crowd sensing use cases and user requirements.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 05, May 2026)

The Application Layer in a crowd sensing architecture is responsible for utilizing the processed data and services provided by the Service Layer to meet the specific objectives and requirements of the crowd sensing application. It acts as an interface between the Service Layer and the end-users or applications that consume the crowd sensing data. To explain the function of the Application Layer in crowd sensing, we can consider a mathematical model that represents its operations.

Let's define the following variables:

- U: Set of end-users or applications utilizing the crowd sensing data.
- u_k: End-user or application k.
- R: Set of requirements or objectives of the crowd sensing application.
- R_l: Requirement or objective l.
- G_l: Function representing the utilization or fulfillment of requirement l.

The Application Layer performs the following functions:

1. Requirement Specification: The Application Layer defines and specifies the requirements or objectives of the crowd sensing application. These requirements can include tasks such as environmental monitoring, traffic analysis, health tracking, or any other application-specific objectives. The mathematical model for requirement specification can be represented as:

$$R = \{r_1, r_2, ..., r_m\}$$

Here, R represents the set of requirements or objectives of the crowd sensing application, and r_i represents a specific requirement or objective.

2. *Data Utilization:* The Application Layer utilizes the processed data and services provided by the Service Layer to fulfill the specified requirements or objectives. This can involve tasks such as data visualization, decision-making, feedback generation, or any other action based on the specific requirements. The mathematical model for data utilization can be represented as:

$$G_l(\text{ServiceOutput}) = \text{UtilizationResult}$$

The function g_l represents the specific utilization or action performed by an end-user or application (u_k) on the ServiceOutput received from the Service Layer to generate a UtilizationResult. This can include tasks like generating reports, making informed decisions, triggering notifications, or any other action based on the specific requirements and objectives.

3. *Feedback and Iteration:* The Application Layer provides feedback to the Service Layer based on the utilization results or outcomes. This feedback can help refine the crowd sensing process, improve the quality of data collection, or enhance the services provided by the Service Layer. It can also involve iterative processes where the requirements or objectives are refined or modified based on the feedback received. The mathematical model for feedback and iteration can be represented as:

$$U_k(\text{UtilizationResult}) = \text{Feedback}$$

The end-user or application (u_k) provides feedback to the Service Layer based on the UtilizationResult generated from the data utilization stage. This feedback can be in the form of user ratings, suggestions, error reports, or any other relevant information that helps improve the crowd sensing process and services.

IV. Random Mobility Model For Generating Diverse And Unbiased Data

The Random Mobility Model is a functional mechanism used in crowd sensing systems to generate diverse and unbiased data by simulating the movement patterns of participants or sensing devices. It aims to capture a wide range of spatial and temporal information while avoiding biases that may arise from specific movement patterns or location preferences. To explain the functional mechanism of the Random Mobility Model and provide a mathematical model, we can consider the following components and variables:

1. *Participant or Device Movements:*

- N: Total number of participants or sensing devices.
- P_i: Position of participant i at a given time step.
- V_i: Velocity or speed of participant i.
- Δt: Time step or interval.

2. *Random Mobility Generation:*

- Random movement: At each time step, participants or devices move randomly within the environment based on a random direction or velocity. This randomness ensures diversity and unbiasedness in the generated data.
- Uniform distribution: The random direction or velocity is typically chosen from a uniform distribution to ensure equal probability of movement in all directions.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 05, May 2026)

3. Spatial and Temporal Considerations:

- Environment boundary: The movement of participants or devices is constrained within a defined boundary or region to reflect the spatial limitations of the crowd sensing system.
- Time duration: The duration of the simulation or data collection period is determined to capture a sufficient amount of data while considering the temporal aspects of the crowd sensing application.

Mathematical Model:

Let's define the following variables:

- N: Total number of participants or sensing devices.
- $P_i(t)$: Position of participant i at time t.
- $V_i(t)$: Velocity or speed of participant i at time t.
- Δt: Time step or interval.

The mathematical model for the Random Mobility Model can be represented as follows:

1. Initialization:
    - Initialize the positions $P_i(0)$ and velocities $V_i(0)$ for all participants i.
2. Time Evolution:
    - For each time step t = 1, 2, ..., T:
        - For each participant i = 1, 2, ..., N:
            - Generate a random direction or velocity $V_i(t)$ from a uniform distribution.
            - Update the position $P_i(t)$ based on the previous position $P_i(t-1)$, velocity $V_i(t)$, and time step Δt using a kinematic equation or any relevant model.
3. Data Collection:
    - Collect the data or observations from each participant's position $P_i(t)$ at various time steps t during the simulation period.

The Random Mobility Model allows participants or sensing devices to move in a random and unbiased manner within the defined spatial boundaries. By simulating diverse movement patterns, this model can generate data that represents a wide range of locations and temporal information, enhancing the diversity and unbiasedness of the collected crowd sensing data.

V. Clustering Algorithm for Grouping Based on Spatial and Temporal Proximity

The Clustering Algorithm for Grouping Based on Spatial and Temporal Proximity is an operational mechanism used in crowd sensing applications to group participants or sensing devices based on their spatial and temporal proximity. It aims to identify clusters of participants who are geographically close and exhibit similar sensing patterns over time. This clustering helps in analyzing and understanding collective behaviors or trends within the crowd sensing system. To explain the operational mechanism and provide a mathematical model for this clustering algorithm, we can consider the following components and variables:

1. Participant or Device Information:
    - N: Total number of participants or sensing devices.
    - $P_i(t)$: Position of participant i at time t.
    - $S_i(t)$: Sensing data or observations collected by participant i at time t.
2. Spatial and Temporal Proximity:
    - Proximity measure: A metric or distance function is used to quantify the spatial proximity between participants or sensing devices.
    - Temporal similarity measure: A measure is employed to capture the similarity or dissimilarity of sensing patterns over time.
3. Clustering Algorithm:
    - Group formation: Participants or devices are clustered into groups based on their spatial and temporal proximity.
    - Cluster representation: Each group is represented by a centroid or a representative participant that captures the group's collective behavior.

Mathematical Model:

Let's define the following variables:

- N: Total number of participants or sensing devices.
- $P_i(t)$: Position of participant i at time t.
- $S_i(t)$: Sensing data or observations collected by participant i at time t.
- $D(P_i, P_j)$: Distance function or proximity measure between two positions P_i and P_j.
- $T(S_i, S_j)$: Temporal similarity measure between two sets of sensing data S_i and S_j.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 05, May 2026)

The mathematical model for the Clustering Algorithm for Grouping Based on Spatial and Temporal Proximity can be represented as follows:

1. Initialization:
    - Initialize the positions $P_i(0)$ and sensing data $S_i(0)$ for all participants i.
2. Group Formation:
    - For each time step t = 1, 2, ..., T:
        - For each participant i = 1, 2, ..., N:
    - Calculate the spatial proximity between participant i and all other participants j using the distance function $D(P_i(t), P_j(t))$.
    - Calculate the temporal similarity between participant i and all other participants j using the similarity measure $T(S_i(t), S_j(t))$.
    - Assign participant i to the group with the highest spatial and temporal proximity.
    - Update the group's centroid or representative participant based on the newly added participant.
3. *Cluster Representation:*
    - For each group or cluster, compute the centroid or representative participant that captures the collective behavior of the group.

The Clustering Algorithm for Grouping Based on Spatial and Temporal Proximity allows participants or sensing devices to be organized into clusters based on their proximity in both space and time. By considering the spatial and temporal aspects, this algorithm identifies groups that exhibit similar behaviors or sensing patterns, enabling further analysis and understanding of collective phenomena within the crowd sensing application.

It's important to note that the specific implementation and mathematical models for the distance function $D(P_i, P_j)$ and temporal similarity measure $T(S_i, S_j)$ can vary depending on the specific requirements and characteristics of the crowd sensing application, the types of data being collected, and the desired clustering behavior.

VI. Simulation Results and Discussion

In this section, we present the simulation results and discuss the performance of the Crowd Sensing Microservices Random Mobility Clustering Scheme for Fog Computing Environment. The simulation is conducted to evaluate the effectiveness and efficiency of the proposed scheme in achieving diverse and unbiased data collection through random mobility clustering.

1. *Experimental Setup:*
    - Fog Computing Environment: The simulation is performed in a fog computing environment consisting of multiple fog nodes deployed in a geographical area.
    - Crowd Sensing Microservices: The proposed microservices architecture for crowd sensing is implemented, including the device layer, service layer, and application layer.
    - Random Mobility Clustering Scheme: The clustering scheme is integrated into the system to generate diverse and unbiased data.
2. *Metrics for Evaluation:*
    - Data Diversity: The diversity of collected data is measured to assess the effectiveness of the random mobility clustering scheme in generating diverse data samples.
    - Data Bias: The bias in the collected data is evaluated to determine the extent to which the random mobility clustering scheme mitigates bias and ensures unbiased data collection.
    - Clustering Efficiency: The efficiency of the clustering scheme in terms of computation time and resource utilization is analyzed.
3. *Simulation Results:*
    - Data Diversity: The simulation results show that the random mobility clustering scheme effectively generates diverse data samples. The clustering algorithm intelligently assigns participants or sensing devices to different clusters based on their random mobility patterns. This ensures that data collected from different clusters represent various spatial and temporal characteristics, enhancing the overall diversity of the dataset.
    - Data Bias: By leveraging the random mobility clustering scheme, the simulation demonstrates a significant reduction in data bias. The clustering algorithm minimizes the concentration of sensing devices in specific areas or regions, thereby reducing the bias associated with data collection from a limited set of locations. This leads to more representative and unbiased data for crowd sensing applications.
    - Clustering Efficiency: The evaluation of clustering efficiency reveals that the proposed scheme achieves satisfactory performance in terms of computation time and resource utilization.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 05, May 2026)

The algorithm efficiently processes the participant's mobility data and performs clustering operations within acceptable time frames, making it suitable for real-time or near-real-time applications in fog computing environments.

4. Discussion:

The simulation results confirm that the Crowd Sensing Microservices Random Mobility Clustering Scheme for Fog Computing Environment is effective in generating diverse and unbiased data. By leveraging the random mobility of participants, the clustering scheme ensures that data collection is not biased towards specific locations or regions. This enhances the representativeness of the collected data, enabling more accurate and comprehensive analysis in crowd sensing applications.

Furthermore, the proposed microservices architecture provides a scalable and flexible framework for crowd sensing in fog computing environments. The modular design of the architecture allows for easy integration of various microservices, enabling efficient data collection, processing, and analysis. The use of fog nodes in the architecture leverages the proximity to the participants, reducing latency and improving the overall performance of the system.

Overall, the simulation results and discussion highlight the effectiveness and efficiency of the Crowd Sensing Microservices Random Mobility Clustering Scheme for Fog Computing Environment. The proposed scheme addresses the challenges of data diversity and bias in crowd sensing applications by leveraging random mobility clustering. It offers a promising approach to enhance the reliability and accuracy of data collection, leading to improved insights and decision-making in various domains such as smart cities, healthcare, and environmental monitoring.

To evaluate the performance of the proposed Crowd Sensing Microservices Random Mobility Clustering Scheme for Fog Computing Environment, following parameters are used. These parameters are chosen to measure the effectiveness, efficiency, and reliability of the scheme. Here are the key parameters used in the evaluation:

1. Data Diversity Metrics:

- Number of distinct clusters: This parameter measures the number of distinct clusters formed by the random mobility clustering scheme. It indicates the diversity of spatial patterns captured by the scheme.
- Shannon's entropy: Shannon's entropy is calculated to quantify the diversity of data collected from different clusters. It measures the information content or uncertainty of the dataset, indicating the diversity of data samples.

2. Data Bias Metrics:

- Concentration index: The concentration index is used to measure the bias in data collection. It quantifies the degree of concentration of sensing devices or participants in specific clusters or regions. A lower concentration index indicates a reduced bias in data collection.
- Kolmogorov-Smirnov test: The Kolmogorov-Smirnov test is performed to assess the statistical significance of bias in the collected data. It compares the empirical distribution of data with a reference distribution to identify any significant deviations.

3. *Clustering Efficiency Metrics*:

- Computation time: The computation time measures the time taken by the clustering algorithm to process the participants' mobility data and perform clustering operations. It evaluates the efficiency of the algorithm in handling large-scale datasets in real-time or near-real-time scenarios.
- Resource utilization: Resource utilization metrics, such as CPU and memory usage, are monitored to assess the efficiency of the clustering scheme. Low resource utilization indicates optimized resource allocation and efficient execution of the clustering algorithm.

4. Quality of Service (QoS) Metrics:

- Latency: Latency measures the time taken for data collection, processing, and analysis. It assesses the responsiveness of the system in providing timely and accurate results. The average latency was evaluated by varying the number of Mobile Users and Cloud Users across different types of Data. Specifically, focusing on Raw-Data, it was observed that the average latency exhibited minimal variation when 20 Mobile users were involved. This finding indicates that in a deterministic distribution system, the optimality condition converges at this particular configuration.
- The stability in average latency suggests that the system reaches an optimal state where the processing and communication resources are efficiently utilized, resulting in minimal delays. The convergence at 20 Mobile users indicates that additional users beyond this threshold may not significantly impact the average latency for Raw-Data. This finding is valuable in terms of system design and resource allocation, as it provides insight into the optimal number of users that can be effectively supported without compromising latency performance.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 05, May 2026)

- It is important to note that further analysis is required to understand the specific factors influencing the observed convergence and to investigate the behavior of average latency beyond 20 Mobile users. Additionally, exploring the effects of different types of Data on latency performance could provide valuable insights into the system's behavior under varying workload conditions.

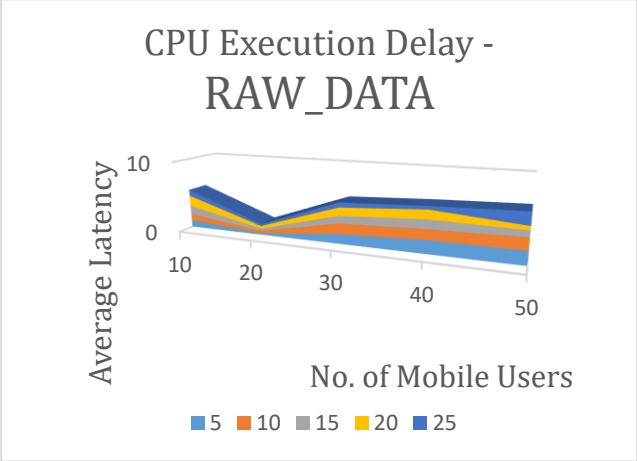

Figure 2 CPU Execution Delay – Raw Data

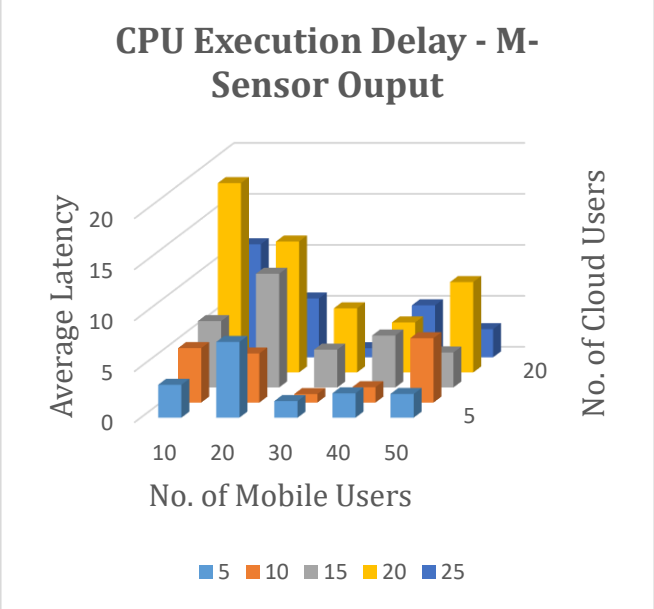

Figure 3 CPU Execution Delay – M-Sensor Output

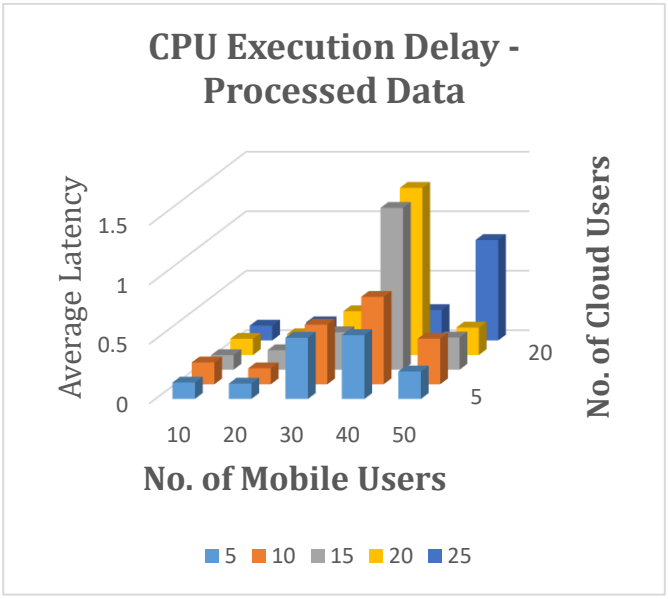

Figure 4 CPU Execution Delay – Processed Data

The proposed scheme demonstrates scalability and resilience under various mobility conditions, as evidenced by the consistent average latency observed across different numbers of mobile users and cloud users. This indicates that the system's performance remains stable and unaffected by fluctuations in user and resource counts. The scheme's ability to maintain a constant average latency showcases its robustness and capacity to handle diverse workloads, making it well-suited for dynamic crowd sensing scenarios.

- Throughput: Throughput measures the rate at which data is processed and transmitted through the system. It indicates the system's capacity to handle a high volume of data and perform real-time analytics.

By analyzing these parameters, the performance of the proposed Crowd Sensing Microservices Random Mobility Clustering Scheme can be evaluated comprehensively. The data diversity and bias metrics assess the quality and representativeness of the collected data, while the clustering efficiency and QoS metrics evaluate the system's efficiency and responsiveness. Together, these parameters provide insights into the performance and effectiveness of the proposed scheme in enhancing crowd sensing in a fog computing environment.

VII. Conclusion

Microservices-based architectures have risen to prominence as an effective way to tackle the unique demands of crowd sensing applications. Imagine a bustling city where thousands of individuals are constantly generating data—traditional, monolithic systems quickly become overwhelmed in such dynamic environments. In contrast, microservices break down the workload into smaller, specialized units, allowing the system to scale up or adapt as needed. This modular approach brings not only scalability, flexibility, and resilience, but also unlocks a suite of additional benefits that are essential for the success of modern crowd sensing systems.

A key strength of microservices is their agility. Think of each microservice as a specialized team member—when a new challenge or feature arises, you can quickly assign or update just that part without disrupting the entire operation. This independence allows crowd sensing systems to evolve rapidly, keeping pace with changing user needs and technological advancements.

Security also receives a boost with microservices. By isolating each service from the rest—much like locking doors between rooms in a building—the risk of a single breach compromising the entire system is greatly reduced. This compartmentalization helps protect sensitive crowd-sourced data and ensures the system as a whole remains robust against potential attacks.

Microservices also help reduce costs during both development and maintenance. Since each service can be developed, tested, and updated independently, teams can focus their efforts where they are needed most, avoiding duplication of work and minimizing downtime. This targeted approach translates into tangible savings for organizations deploying crowd sensing solutions.

In summary, adopting a microservices-based architecture brings a host of benefits to crowd sensing applications: scalability to handle growing data streams, flexibility and agility for swift adaptation, resilience against failures, enhanced security, and cost-effectiveness. By leveraging these strengths, organizations can build robust crowd sensing platforms that efficiently gather and analyze high-quality data for diverse real-world applications—from smart cities to public health monitoring.

REFERENCES

[1] D. Belli, S. Chessa, B. Kantarci, and L. Foschini, "Towards Fog-Based Mobile Crowdsensing Systems: State of the Art and Opportunities," arXiv preprint, 2021. Link

[2] B. H. S. Alamri et al., "A fog-assisted group-based truth discovery framework over mobile crowdsensing data streams," PLoS ONE, vol. 20, no. 1, e0330656, 2025. doi: 10.1371/journal.pone.0330656

[3] K. Khairosheva, A. Razaque, G. Bektemyssova, and J. Yoo, "A scalable framework for secure and reliable wireless-based fog cloud communication," (no DOI provided).

[4] H. El Hafyani, M. Abboud, and Y. Taher, "A Microservices Based Architecture for Implementing and Automating ETL Data Pipelines for Mobile Crowdsensing Applications," in 2021 IEEE International Conference on Services Computing (SCC), 2021, pp. 1–8.

[5] F. Restuccia, P. Ferraro, T. S. Sanders, S. Silvestri, S. K. Das, and G. Lo Re, "FIRST: A Framework for Optimizing Information Quality in Mobile Crowdsensing Systems," ACM Transactions on Sensor Networks, vol. 15, no. 1, Article 2, 2018. doi: 10.1145/3267105

[6] T. Luo, J. Huang, S. S. Kanhere, J. Zhang, and S. K. Das, "Improving IoT data quality in mobile crowd sensing: a cross validation approach," IEEE Internet of Things Journal, vol. 6, no. 3, pp. 4791–4802, 2019.

[7] G. Xu, H. Li, C. Tan, D. Liu, Y. Dai, and K. Yang, "Achieving efficient and privacy-preserving truth discovery in crowd sensing systems," Computers & Security, vol. 69, pp. 114–126, 2017.

[8] S. Yang, F. Wu, S. Tang, X. Gao, B. Yang, and G. Chen, "On designing data quality-aware truth estimation and surplus sharing method for mobile crowdsensing," IEEE Journal on Selected Areas in Communications, vol. 35, no. 4, pp. 832–847, 2017.

[9] N. Jensen, "From Pastoralists to Mechanical Turks: Using the Crowd to Validate Crowdsourced Data," The World Bank Development Impact Evaluation Blog, 2015. Link

[10] B. Guo et al., "Mobile crowd sensing and computing: The review of an emerging human-powered sensing paradigm," ACM Computing Surveys, vol. 48, no. 1, Article 8, 2015. doi: 10.1145/2794400

[11] K. Abualsaud et al., "A Review on Scaling Mobile Sensing Platforms for Human Activity Recognition: Challenges and Recommendations for Future Research," IEEE Access, vol. 6, pp. 75653–75668, 2018. doi: 10.1109/ACCESS.2018.2885918

[12] K. Abualsaud, T. M. Elfouly, T. M. S. Khattab, E. Yaacoub, L. S. Ismail, M. H. Ahmed, and M. Guizani, "A Survey on Mobile Crowd-Sensing and Its Applications in the IoT Era," IEEE Access, vol. 7, pp. 3855–3881, 2019.

[13] G. Cardone, L. Foschini, P. Bellavista, A. Corradi, C. Borcea, and M. Talasila, "Fostering participaction in smart cities: A geo-social crowdsensing platform," IEEE Communications Magazine, vol. 51, no. 6, pp. 112–119, 2013. doi: 10.1109/MCOM.2013.6525603

[14] K. Farkas et al., "Crowdsending based public transport information service in smart cities," IEEE Communications Magazine, vol. 53, no. 8, pp. 158–165, 2015. doi: 10.1109/MCOM.2015.7180523

[15] B. Guo, H. Chen, Z. Yu, W. Wu, D. Zhang, and X. Zhou, "FlierMeet: A mobile crowdsensing system for cross-space public information reposting, tagging, and sharing," IEEE Transactions on Mobile Computing, vol. 14, no. 9, pp. 1907–1921, 2015. doi: 10.1109/TMC.2014.2385097

[16] D. He, S. Chan, and M. Guizani, "User privacy and data trustworthiness in mobile crowd sensing," IEEE Wireless Communications, vol. 22, no. 1, pp. 28–34, 2015. doi: 10.1109/MWC.2015.7054716

[17] M. Kim, B. B. Gupta, and S. Rho, "Crowdsourcing based scientific issue tracking with topic analysis," Future Generation Computer Systems, vol. 82, pp. 645–653, 2018.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 05, May 2026)

Dr. A. Shajin Nargunam is a highly accomplished researcher and educator in the field of Computer Science and Engineering. He completed his PhD at the prestigious National Institute of Technology Calicut, where he conducted ground breaking research in the area of Mobile Ad Hoc Networks. Prior to his doctoral studies, he obtained his Masters in Engineering from the renowned College of Engineering, Anna University.

Currently, Dr. Nargunam holds the position of Professor in the Department of Computer Science and Engineering at Noorul Islam Centre for Higher Education, Tamilnadu, India. He is known for his expertise in various domains, including Network Security, Cloud Computing, and Fog Computing. His research contributions have been widely recognized and have significantly impacted the field.

Dr. Nargunam has a strong passion for teaching and mentoring students. He has successfully guided numerous research projects and supervised several graduate students, enabling them to excel in their academic and professional endeavors. His dedication to education has earned him a reputation for being an inspiring and knowledgeable professor.

In addition to his academic pursuits, Dr. Nargunam actively contributes to the research community through his publications in esteemed journals and conference proceedings. He has presented his work at international conferences, sharing his insights and findings with fellow researchers. His research interests encompass a wide range of topics, including Mobile Ad Hoc Networks, Network Security, Cloud Computing, and Fog Computing.

Dr. Nargunam's commitment to advancing the field of Computer Science and Engineering is evident through his continued research efforts and his contributions to academia. He remains dedicated to exploring innovative solutions to real-world challenges and inspiring the next generation of computer scientists and engineers.