Mobi-IoST: Mobility-aware Cloud-Fog-Edge-IoT Collaborative Framework for Time-Critical Applications

Shreya Ghosh, Student Member, IEEE, Anwesha Mukherjee, Student Member, IEEE, Soumya K. Ghosh, Member, IEEE and Rajkumar Buyya, Fellow, IEEE

Abstract—The design of mobility-aware framework for edge/fog computing for IoT systems with back-end cloud is gaining research interest. In this paper, a mobility-driven cloud-fog-edge collaborative real-time framework, Mobi-IoST, has been proposed, which has IoT, Edge, Fog and Cloud layers and exploits the mobility dynamics of the moving agent. The IoT and edge devices are considered to be the moving agents in a 2-D space, typically over the road-network. The framework analyses the spatio-temporal mobility data (GPS logs) along with the other contextual information and deploys machine learning algorithm to predict the location of the moving agents (IoT and Edge devices) in real-time. The accumulated spatio-temporal traces from the moving agents are modelled using probabilistic graphical model. The major features of the proposed framework are: (i) hierarchical processing of the information using IoT-Edge-Fog-Cloud architecture to provide better QoS in real-time applications, (ii) uses mobility information for predicting next location of the agents to deliver processed information, and (iii) efficiently handles delay and energy consumption. The performance evaluations yield that the proposed mobility prediction algorithm has approximately 93% accuracy and reduced the delay and power by approximately 23-26% and 37-41% respectively than compared to the existing mobility-aware task delegation system.

Index Terms—Cloud computing, Edge computing, Fog computing, Internet of Things (IoT), Mobility analytics, Spatio-temporal data.

1 INTRODUCTION

The advancements of Internet of Things (IoT) have manifested significant improvements on the quality of human lives in varied aspects [1]. To facilitate real-time applications, high-end processing and storage units are required. For computation and storage of these large volume of raw data generated by IoT devices, cloud computing is the most significant infrastructure. However, the cloud-only set-up is not an energy-efficient and delay-aware solution for handling such a high volume of data. To address this problem, edge and fog computing have been introduced [2]. However, the seamless connectivity due to the mobility of IoT devices is a crucial factor to process the data in the remote cloud servers. For time-critical applications such as health care, connection interruption and consequently the increase in delay in delivering the processed information, result in poor Quality of Service (QoS). If the device gets disconnected due to mobility, the delivery of the result becomes a challenge. In [3], the task delegation using push notification or serializing session information and application offloading using virtual machine migration have been discussed based on user mobility. Nevertheless, the device has to receive the result from the cloud by serializing session information when gets connected [3]. Hence, the user has to request explicitly for the result. For critical real-time applications, this may not be a good solution. This necessitates a hierarchical infrastructure, where each layer (IoT, edge, fog or cloud) either accumulates, stores and processes the information for reducing the delay.

On the other side, movement traces, i.e., time-stamped location information of moving agents (say, mobile-users or client) are accumulated on a large scale from GPS-enabled smart phones or IoT devices. This spatio-temporal movement information opens up diverse opportunities to explore the intent of movement [4] and thus fostering varied location based services, namely, efficient package delivery [5], traffic resource management etc. Internet of Spatial Things (IoST) brings IoT in the spatial context [6]. As discussed before, mobility or continuous change of locations of users is a challenging issue in task delegation or data offloading. However, analysing these mobility information helps to explore the intent of the move and subsequently extracts the frequent movement path of a user in different contexts. If the probable location sequences of an agent in the near future can be predicted from the historical mobility information, then an effective and delay-aware solution for a time-critical application can be provided.

To address the above-mentioned challenges, in this work, we propose a Cloud-fog-edge based collaborative framework for the processing of IoT data and delivering the result based on mobility analysis to reduce the delay. We have considered a hierarchical mobility-based infrastructure composed of four layers: IoT layer, edge layer, fog layer, and cloud layer. Nowadays smart phone has become a
popular medium for ubiquitous Internet access and varied user-specific IoT applications are accessible through smart phones. These mobile devices serve as edge devices and constantly change the locations. The users of our system utilize these time-critical applications while traveling across. The edge layer contains such edge devices i.e. mobile devices. The fog layer contains the fog devices such as RSUs which are large cell base stations. While the IoT and the edge devices change their locations, the RSU and the cloud data centers of the framework have static locations. The raw data generated in the IoT layer is sent to the edge layer, which is connected with the fog layer. The fog layer is connected with the cloud layer where high-end processing and mobility analysis tasks are performed.

![Image](image.png)

Fig. 1: Mobi-IoST for health care application
(a): Ambulance sends health data generated from IoT devices to RSU.
(b): RSU sends the result to cloud along with the location information
(c): Cloud predicts the nearby health care centre, shortest path and helps to actuate traffic signal

### 1.1 Motivating Scenario

We have considered a well-known time-critical application, health care, where the proposed framework Mobi-IoST can be utilized. The pictorial representation of this use-case is shown in Fig. 1.

Suppose a patient, travelling in an ambulance (am), needs continuous monitoring of her/his vital health parameters. The health parameters such as blood-pressure, pulse-rate, body-temperature etc. are collected using IoT devices and the raw data are sent to the RSU through a client application. The RSU processes the information and sends the current status as normal/abnormal to the client-app. If an abnormality is detected, the RSU sends the data to the cloud to find out the nearest health facility. Given the current location and health-data feed from the RSU, the cloud can suggest the nearby hospital. On the other side, based on the route followed by the ambulance, the probable health centre also gets notified. Further, the mobility analysis module of cloud can help to reduce the commuting time of the ambulance by predicting less congested path in the road-network. This can be achieved when cloud analyses the traffic states (congestion, traffic breakdown etc.) of the roads and notifies the RSUs of the path. The RSU will work as a fog device and the respective RSUs can actuate the signal synchronizing mechanism such that am can reach avoiding the congested route as well as without waiting in the traffic signals. Although the scenario is motivated by the dysfunctional public health system and limited access to improved transportation and medical care in the rural areas, specifically in developing countries, such as India, Mobi-IoST is beneficial in other time-critical applications as well. For instance, in the time of emergency, a police-vehicle or a fire extinguisher car needs to commute with a minimal delay avoiding the congested regions of a city. Mobi-IoST predicts the less congested route by analyzing the traffic states in real-time and notifies the RSUs of the route. These RSUs actuate the signal synchronizing mechanism such that the vehicle can reach the destination avoiding the congested route as well as without waiting in the traffic signals. The hierarchical placement of IoT, Edge, Fog devices and cloud servers in Mobi-IoST framework facilitates an effective and delay-aware solution for several time-critical applications. We believe that Mobi-IoST will act as a foundation of mobility aware network resource management for varied location-based service planning in real-time.

### 1.2 Contributions

The focus of our work is to develop a Cloud-fog-edge collaborative framework which facilitates real-time IoT information processing and delivery of results based on the mobility information analytics. The key contributions of this paper can be summarized as follows:

- Mobi-IoST (Mobility-aware Internet of Spatial Things) is designed for information processing and delivering result based on the prediction of the user’s current location. The framework exploits the mobility knowledge of the agents to predict the probable user location and delivery of processed information at low delay and low power consumption of the user-device.
- A novel mobility modelling network has been proposed to explore the movement patterns of the user. The huge amount of spatio-temporal trajectory data is stored efficiently along with other contextual information in the cloud data centre.
- A real-time mobility prediction module has been designed to predict the location sequences of the user effectively.
- The experimental results demonstrate that the proposed system has outperformed other existing approaches in accuracy and takes much less time to learn the patterns.
- The simulation results demonstrate that the proposed framework reduces the delay in delivering information and power consumption of mobile device (user-device) compared to the existing mobility-aware task delegation approach.

To the best of our knowledge, this work is the first attempt to utilize the movement knowledge to enhance the QoS for facilitating time-critical IoT applications. The rest of the paper is structured as follows. Section 2 briefs the existing work in related areas. We propose our framework, Mobi-IoST in
section 3 and discuss several modules of the framework. The delay and power consumption models are discussed in section 4. Section 5 demonstrates the experimental and simulation results. The paper is concluded in section 6 along with future directions.

2 RELATED WORK

The IoT refers to the connection of embedded devices within an existing Internet infrastructure where the devices are uniquely identified and the computing environment is created [1]. The sensor nodes are usually used as IoT devices, which collect the object status. The raw data collected by IoT devices are processed inside the cloud servers. However, storing and processing of the raw data inside the remote cloud enhances the delay and energy consumption. To overcome this, fog computing has been introduced [2]. Nowadays IoT and fog become interrelated to each other [2]. The raw data of IoT devices are processed inside the fog device instead of the remote cloud to reduce the delay and energy consumption. However, during data processing connection interruption becomes a challenge if the client is a mobile device. IoT has several sub-domains depending on its applications e.g. Internet of Multimedia Things (IoMT), Internet of Health Things (IoHT), Internet of Vehicles (IoV) etc [6]. IoST is a new sub-domain of IoT, which focuses on spatial data management [6]. IoST refers to “ubiquitous and embedded computing devices that transmit and receive information so often includes numerical values about a physical object that can be represented in a geographic coordinate system for geospatial interoperability requirements over networks”[6]. In fog-based IoT, the switch, routers etc. work as fog devices for faster processing of the raw data collected using IoT devices. The mobile device that usually works as an edge device, is a connector between the IoT devices and the network. Though nowadays different apps are present in smart phones, resource hindrance is a major difficulty for these handheld devices. Battery life of the device, memory, storage become major constraints. Therefore, the cloud servers have to be used by mobile devices to store their data [7]. The mobile devices also offload heavy computations inside the cloud in case of resource limitation and saving battery life. However, for small amount of data processing the use of remote cloud increases delay and power consumption of the mobile device. But the device is unable to do the processing due to resource limitation. Energy and latency in offloading has been focused on several existing approaches [8], [9]. The use of cloudlet for reducing energy consumption in the multi-cloudlet scenario has been highlighted in [9]. Fog computing has also provided solutions for reducing delay and energy in the processing of IoT data [2]. In fog computing, a hierarchical architecture is followed, where the intermediate devices between the end node and cloud servers participate in data processing, and these nodes are called fog devices [2]. The edge devices allow users to connect with the network and transfer data accordingly to a network which is external to the user.

For the offloading of massive amount of video at scale, a cloud and edge computing based collaborative system has been proposed in [10]. For balancing the traffic and computing load, a method has been discussed in [11], where the IoT devices are allocated to the base station or fog nodes to reduce the latency.

Network connectivity is a challenge in vehicular network [12]. For task offloading in such networks, edge computing has been used in [12]. The mobile edge computing servers are deployed inside the road side access points in that case. These servers are used for offloading tasks. However, the use of access points may not be energy-efficient if exhaustive computations have to be performed and there are a large number of users. Based on user mobility, an opportunistic computation offloading method has been discussed in [13]. Based on information gain, task allocation in spatial crowdsourcing has been discussed in [14]. A fog based architecture of spatial crowdsourcing has been proposed in [15]. In this work, the authors have focused on privacy-aware task allocation and data aggregation. In [3], task offloading to cloud and delivery of result based on serialization of session information has been discussed. However, the user if gets disconnected has
to retrieve the result by sending a request to the cloud. For the cloudlet based scenario, virtual machine migration has been discussed in [3]. Nevertheless, the VM migration has to take place from one cloudlet to another until the processing gets finished and the result is delivered to the user. This is expensive as well as increases the energy consumption of the cloudlet. Hence, there should be a mechanism where the user mobility will be predicted and result will be delivered at the optimum delay and power consumption of the mobile device.

Given the abundance of mobility traces (GPS log) of individuals, there are several research initiatives to extract knowledge or meaningful information (i.e., making sense of raw GPS log) from the huge amount of trajectory traces. Several works are reported to predict next location from movement traces such as GPS log, check-in data or social network information [16], [17], [18]. There are challenging applications, namely, urban land-use classification from taxi-traces [19], categorizing users in an academic campus [20] or catching pick-pockets from large-scale transit records [21]. It is well known that human movement traces follow spatio-temporal regularity. In this regard, Song et al. [22] provide a high degree of spatio-temporal uniformity by mining movement traces of 50,000 people for a period of three months. All of these studies depict that since people follow some spatio-temporal regularity in their movement history, therefore an appropriate and effective mobility pattern modelling can help to facilitate several location-aware services.

To this end, the Mobi-IoST framework aims to deliver mobility-driven efficient data processing in cloud-fog-edge based IoT setup to facilitate intelligent decision making in real-time. One of the major aspects of the proposed framework is mobility-aware service provisioning, which helps to reduce the delay, power consumption of the user-device and as well as facilitating intelligent recommendations based on the present location of the user-device. For example, recommending the nearest health-care center based on the health condition of the patient (say, cardiac problem) and the available facilities of the health-care centers. Furthermore, the RSUs can help to synchronize the signaling mechanism based on the traffic states of a road network (Fig. 1). To the best of our knowledge, there are several research efforts in the domain of mobile big data mining and fog based IoT, but is somewhat fragmented and no other existing works have clearly depicted the significance of mobility-aware service provisioning framework in fog based IoT. In Mobi-IoST, the movement pattern modelling and location prediction approaches are novel propositions which deliver result in real-time. Moreover, the experimental observations and performance analysis show the effectiveness of Mobi-IoST in terms of accuracy, delay and power consumption. In summary, designing and deploying an end-to-end mobility driven framework for efficient data processing in IoT setup is a challenging issue in the present era.

3 Mobi-IoST Framework

The hierarchical structure of Mobi-IoST is represented in Fig. 2. Fig. 3 depicts the overall flow of the framework, Mobi-IoST. IoT or Internet of Spatial Things deals with IoT data along with spatial perspective. As depicted in Fig. 2, in the bottom layer several IoT sensors such as accelerometer, GPS, temperature, blood-pressure, proximity sensors capture application specific data. These IoT sensors are either present within the edge devices or connected with the edge devices, namely, mobile phone, vehicles, which change their locations. When any of these edge devices needs assistance, it contacts the nearest RSU. In this work, RSU is used as fog device and it is capable of small scale processing. If the processing is beyond the computational capability of the RSU, then it delegates the task to the cloud. The top layer of the hierarchical structure consists of cloud servers, which store spatial data, specifically, mobility traces, location-specific information, city-structure (POIs placements...
and other contextual information). The cloud processing unit executes the task and sends the result to the RSU, where the tasks are application-specific. For example, in case of an emergency, an ambulance needs to reach the destination without any delay. The cloud processing unit extracts and discovers the present traffic status and predicts the path with minimum commuting time. As the cloud storage unit has all the RSU information, it notifies all the RSUs within the predicted path for signaling and actuating such that the ambulance does not face any traffic congestion. Further, for any personalized recommendation, a mobile user can always send a request to the nearest RSU. Suppose, a user captures her health-related data, namely, temperature, blood-pressure and weight using the IoT devices and sends a request to the RSU through the mobile device for predicting the current health status. Furthermore, the resources can be efficiently managed by this framework: movement analysis module can predict variation of travel demand apriori and notify the RSUs accordingly, while the RSUs can decide about the dissemination of resources (traffic or network) efficiently.

The major modules of the framework are: (i) movement pattern modelling, (ii) predicting next location sequences, (iii) delivery of result after processing in a timely manner. Finally, the experimental and simulation results yield the effectiveness of the framework. The detailed approaches are discussed in the remainder of the paper.

### 3.1 Exploring Movement Semantics from Trajectory Traces

This section presents the methodology to model movement patterns and predicts the next location sequences efficiently and timely manner. Location prediction of moving agents, such as, people, vehicles is a challenging task for varied location-based services [23]. Specifically, in our work, location prediction helps to locate the moving agent’s locations in near future and subsequently data is sent to the appropriate RSU. Whenever a mobile device gets connected to a RSU, the GPS log of the mobile device is extracted and stored in the cloud dynamically.

It may be noted that location prediction depends on several factors, namely, day of the week, time-slot of a day and road-structure. The first step of movement behaviour modelling is to find out the frequent pattern followed by the users in varied contexts. For example, the path followed by an individual differs significantly in weekends compared to his/her weekdays’ trajectory signature. Moreover human movements follow some intent [4] and extracting the purpose behind any move is the fundamental step to predict next location effectively. Few preliminary concepts which are used in this paper are defined as follows:

- **GPS log** ($G$): GPS log is the collection of time-stamped latitude, longitude information. The GPS trajectory or trace is formed by connecting the location information on increasing time-ordering. $Traj(p_1,\ldots,p_n)$ : $\langle p_1(lat_1, lon_1), t_1 \rangle \rightarrow \langle p_2(lat_2, lon_2), t_2 \rangle \rightarrow \cdots \rightarrow \langle p_n(lat_n, lon_n), t_n \rangle$, where $t_1 < t_2 < \cdots < t_n$
- **Stay-Point** ($S$): Stay-point of a trajectory is defined as a location (typically, polygon), where the moving agent stops for a time-value $\delta t$ and $\delta t > t_{thresh}$. Here, $polygon$ is a data-type of spatial data [24] and the area of the polygon is less than $area_r$. $t_{thresh}$ is the time-threshold for detecting stay-points from the trajectory. In our analysis, we have considered the parameter values as, $t_{thresh} = 12\text{mins}$ and area$_r = 2\text{km}^2$.
- **POI and Geo-tagged Trajectory**: Point-of-interest (POI) of a GPS location denotes the nearby landmark of a location, such as, residential area, supermarket etc. We have followed the POI taxonomy\(^1\) to extract such POI information using Google Place API. Geotagged trajectory is generated by appending the geo-tagged information of the stay-points within the trajectory.
- **Trajectory window** ($TrajW$): Trajectory window stores the location sequence information between two such stay-points in an uniform sampling rate.

### Augmenting Semantic Information with GPS log:

Human movement semantics can be analysed if additional information such as, POI, road-network structure and stay-point information are appended with the raw GPS traces.

- Road network of the study region is extracted from OpenStreetMap (OSM)\(^2\). The road network is represented by a directed graph $R = (V, E)$, where $e \subseteq |E|$ denotes the road-segments of the region and $v \subseteq |V|$ is the intersection points of such road segments. Map-matching algorithm [25] has been deployed, which considers both geometric and topological structure of the road-network to associate the road-segments along with the trajectory traces.
- Each stay-point of the trajectory is geo-tagged with the nearby POI location. Here, we have implemented the iterative reverse geo-coding technique to extract nearest landmark of the stay-point.

After the addition of semantic information with the raw traces, a trajectory trace takes the form:

- $< p_{ta}, residential >, TrajW[(p_i, t_i, e_x), (p_{i+1}, t_i + \delta t, e_x), (p_{i+2}, t_i + 2 \times \delta t, e_x), \ldots]$,
- $< p_{tb}, super market >, TrajW[(p_j, t_j, e_y), (p_{j+1}, t_j + \delta t, e_y), (p_{j+2}, t_j + 2 \times \delta t, e_y), \ldots]$,
- $< p_c, residential >$.

Here, $p_a, p_b$ and $p_c$ are three stay-points with geo-tagged information residential building and supermarket. $TrajW$ stores the route information followed by the trajectory, where $e_x, e_y$ are the road-segments of the road-network of the study region.

### Processing of Large Mobility Datasets in Cloud

With the advances in sensor technologies and the proliferation of smartphones, a huge amount of mobility traces are generated by moving agents. One of the major challenges is to analyze the vast amount of data due to computational complexity and storage limitations. To this end, we propose to migrate the computation of mobility analysis and storage of movement traces in the cloud for faster response. It may be noted that the locations and coverage areas of the RSUs

---

2. OpenStreetMap: https://www.openstreetmap.org
need to be maintained in the cloud storage such that it can predict the next location of the moving agent to determine the appropriate RSU, which will serve the agent at that time. Here, large cell base stations [26] are the RSUs. The macrocell base station is referred to as macro RSU and microcell base station is referred to as micro RSU. The coverage area of macro RSU and micro RSU are 1-20km and 200m-1km respectively [26]. The framework is implemented in Google Cloud Platform (GCP) by utilizing several storage and computational components of GCP. In our framework, cloud storage is of four types:

- **Grid based storage of the study region:** The study region is segmented into uniform hexagonal grids and information, such as, road structure or POIs, RSUs are associated with each such grids. Our proposition is to segment the spatial region into grids such that each grid encloses the coverage area of at least one micro RSU. The grid-segmentation process initiates with the location of one RSU. Suppose, the location of the RSU (say, RSU$_i$) is $p = (x, y)$ and the length of the side of the hexagon ($g_i$) is $a = 8m$. An iterative process is deployed until the complete study region is segmented with hexagonal grids. In the first iteration, center points of the 6 neighbouring grids of $g_i$ are calculated and subsequently, the neighbouring hexagonal grids are constructed.

  In the next step, Geohash code of all hexagonal grids are computed. Geohash code of the grids represent the spatial location on the earth surface using unique alphanumeric strings. Cloud Spanner of GCP is used to store these information which supports horizontal scalability.

- **Road network information storage:** This module stores the road network information, namely, connections among different road-segments and road-type (highway, lane etc.). The information is stored in an adjacency matrix format, where each vertex maintains a list of outgoing edges (outdegree of the vertex). The data-type of the list is polyline, which is a spatial-data type [24] and represents the road-segments on the map.

- **RSU information storage:** It stores the list of RSUs along with the unique-id, coverage area, location (latitude, longitude) and other information such as, type (micro RSU or macro RSU) etc. Cloud BigQuery of GCP is utilized to store the road network information and RSU information as well.

- **Frequent path storage:** The frequent path followed by individual moving agents are extracted and modelled in our work. Details are presented in section 3.1.1. Cloud Bigtable of GCP is utilized to store the road network information.

The computational cost of the mobility traces is huge since it deals with time-series data with very high sampling rate. The key challenge is to reduce the processing time of the location prediction, and therefore, an efficient scheme is required. Here, we have deployed a hash-based indexing scheme, where nearby locations are stored in the subsequent buckets of the hash-table.

3.1.1 **Movement behaviour modelling**

In this section, we discuss how movement behaviour of users can be modelled to explore the frequent paths followed by them in different contexts. The process of semantic enrichment of GPS log of users has already discussed in section 3.1. Here, we propose **User movement graph**, a multi-layer graphical model to model the users’ movement patterns from the spatio-temporal context. The objective to use the multi-layer network is that human movement patterns typically depend on temporal variations (weekdays or weekends, morning or evening), road networks and stay-points. All of these information need to be encoded and interconnections of the information cannot be properly captured in a single layer.

**User movement graph (MG):** User movement graph is defined as $MG = (N, L, la)$, where $N$ denotes the nodes, $L$ denotes the links and label is represented by $la$. The user movement graph has four labels:

- Road network: The layer 1 consists of road network information, where nodes are intersection points of road-segments.
- RSU network: The RSU information (location and coverage area) is stored in layer 2.
- Stay-point information: The stay-point information including location and type of stay-point are stored in layer 3.
- Frequent path: The movement paths frequently followed by the user is stored in layer 4.

It may be noted that each layer is interconnected with each other. As the construction of layer 1, layer 2 and layer 3 are straightforward, we discuss the frequent pattern mining process of layer 4 in detail.

The frequent path network of layer 4 is represented by probabilistic graphical model or Dynamic Bayesian network $FPN(V, E, \Upsilon)$ where $V$ is the set of stay-points, $E$ denotes the direction of visit among different stay-points and $\Upsilon$ is the network quantify parameter. The key intuitive to utilize probabilistic graphical model is that the visit sequence of a person somewhat follows conditional dependency. In simple words, whether a person will visit location $l_1$ or $l_2$ at $t + 1$, depends on her present stay-point at $t$. In this work, we have considered both spatial location and temporal span of a visit-sequence to model FPN of user movement graph. Each node (stay-points: $v \subseteq V$) of the network is conditionally independent of its non-descendants given its parent node ($Pa(v)$). Suppose, a visit-sequence is given as $V = (V_1, \ldots, V_N)$, the probability distribution is computed as follows:

$$P(V) = \prod_{i=1}^{N} P(V_i|Pa(V_i))$$ (1)

FPN captures the dynamic nature of the mobility information by representing multiple copies of the spatial-information, one for each time-slice $V_i = (V_{1,t}, \ldots, V_{d,t})$. Subsequently, the transition distribution from one state to other ($P(V_{t+1}|V_t)$) is computed from two time-slice Bayesian network. The spatial location information ($V_t$) is typically divided into two sets, namely, unobserved state variables ($S_t$) and the observed state variables ($L_t$, in our
case, location information from RSUs). The joint probability distribution is calculated by unrolling two time-slice Bayesian networks:

\[
P(S_0, \ldots, S_T, L_0, \ldots, L_T) = \\
P(S_0)P(L_0|S_0) \prod_{i=1}^{T} P(S_i|S_{i-1})P(L_i|S_i)
\]  

(2)

It may be noted that we have represented the stay-points as grid-location and transition from one state to another state signifies that the agent is moving from one grid to another.

Next, we deploy a spatio-temporal trajectory clustering (TrajCS) on FPN which captures the signature or frequently visited paths of the individual. The process follows a hierarchical top-down approach, and based on the distance measure new clusters are generated and appended in the list. The trajectory clustering distance measure is computed as follows:

\[
\text{TrajCS}(S_i, S_j) = \begin{cases} \\
0 & \text{if} (i == 0) \\
\text{TrajCS}(S_{i-1}, S_{j-1}) & \text{if} ((S_i == S_j) \\
+C \times \min(T_{\text{Score}_i}, T_{\text{Score}_j}) & \text{if} ((S_i == S_j) \text{ and } (s_{i+1} \neq (s_{j+1})) \\
\text{MAX} (\text{TrajCS}(S_{i-1}, S_j), \text{TrajCS}(S_i, S_{j-1})) & \text{if} (s_i \neq s_j)
\end{cases}
\]

where \( S_i \) represents a set of locations, \( s_i \) denotes one GPS point of the set \( S_i \) and \( C \) is the parameter to augment the probability of the path taken. \( T_{\text{Score}} \) computes the temporal similarity between two different stay-points of the trajectory and TrajCS method is recursively called for extracting the signature pattern. The proposed distance measure append temporal information with the conventional LCSS clustering method [27]. Algorithm 1 describes the basic steps of generating FPN from the trajectory traces of agents.

This section describes how users’ frequent movement patterns are extracted and stored along with other contextual information. Furthermore, the trajectory clustering and multi-layer graphical model help to effectively model the movement behaviour of agents in the cloud server.

3.1.2 Next location sequence prediction from mobility information

This helps to calculate the probable path visit by the user a-priori. The location prediction task is formulated as follows: Given the historical observations of an moving agent \( m \) and the current location \( L \) at time \( t \), predict the agent’s anticipated location sequences (S) following \( \delta \) time-instances

Typically, the task is formulated as information retrieval task considering the fact that people’s movement patterns follow spatio-temporal regularity and effective movement behaviour modelling leads to accurate location prediction. Here, we have deployed Hidden Markov model (HMM) (say \( \chi \)) based prediction technique with two kinds of stochastic variables, state variables (hidden) and observable variables. Each individual’s movement is modelled as \( k^{th} \) order Markov chain and the transition from one place to another place is modelled based on \( MG \) and \( \chi \).

Algorithm 2 describes the basic steps of location prediction. The first step map locates the current location (grid location) of the moving agent and then the model predicts the sequences of locations based on the context and finally, update process updates the result based on the current input from the mobile device. It may be noted that the order \( (k) \) of the markov chain is dependent on the user’s frequent movement pattern and extracted from FPN of user movement graph (MG).

\[
P(s_i|s_{i-1}, s_{i-2}, \ldots, s_1) = P(s_i|s_{i-1}, \ldots, s_{i-k})
\]

(4)

In the next step, forward algorithm [28] is deployed to find out the sequences of stay-points, given as:

\[
P(L_k|\chi) = \sum_{s_{\text{seqmax}}} P(L_k|s_{\text{seqmax}}) \prod_{j=1}^{k} P(L_j|s_j) \tag{5}
\]

where, \( \text{seqmax} \) and \( s_j \) represent the maximum number of hidden state sequences and hidden states. Here, model is represented as \( k^{th} \) order markov chain where the next location depends on \( k \) recent observations. Next, a variant of verterbi algorithm using time-relationships is deployed to discover the possible sequences of states. The transition and emission probabilities of \( \chi \) are computed by adjusting the model parameters. An iterative version of forward backward algorithm is implemented to produce the sequences effectively.

Three types of location prediction tasks have been carried out in this work: (i) location sequence prediction in a specific time-threshold, (ii) predicting appropriate POI (say, health-care center) and the path and finally, (iii) given the destination and present location of the agent predicting the path with less commuting time based on the traffic states of the road-network. The location sequence prediction in specific time-thresholds is computed directly from \( \chi \) on \( MG \). The POI and path prediction is carried out by overlapping the road-network structure (layer 1 of \( MG \)) and frequent path pattern (layer 4 of \( MG \)). Finally, given source and destination, the markov model is used along with the traffic-state of the region, where \( s_1 \) and \( s_n \) are specified. It may be noted that our algorithm (Algorithm 2) is self-adaptive, i.e., \( \text{update} \) function (also, refer ‘Update’ arrow of Fig. 3) changes the modelling algorithm in case any of the prediction result fails.

3.2 Delivery of Result After Data Processing

The IoT devices are connected with the edge device, e.g., the sensors within the mobile device. With the increasing availability of smartphones, we have considered mobile devices as the edge devices. The mobile device will process raw data received from the IoT devices. If the mobile device is able to perform the processing, it does the same by working as an edge device and generates the result. Otherwise, the mobile device sends the data to the RSU, which will act as a fog device. Each RSU maintains a look-up table, which holds the mobile device IDs present under its coverage. The International Mobile Equipment Identity (IMEI) number is considered as the mobile device ID. The RSU after receiving the raw data from the mobile device, checks its current load and ability to process the data. In this regard, two cases appear as follows:
Algorithm 1: Frequent path mining - A trajectory clustering approach

Input: Set of trajectory \( T \), stay-points \( S \)
Output: Frequent path network \( < FPN(V, E, T) > \)

1: \( clus, S, V, E \leftarrow NULL; \)
2: for each trajectory window \( tr \in T \) do
3: for each unvisited stay-point \( s \in S \) do
4: \( V.append(s) \)  
5: \( visited \leftarrow s \)
6: \( Y \leftarrow \text{CPT} \leftarrow \text{Create ConditionalProbabilityTable}(s) \)
7: \( t \leftarrow \text{extractTemporal}(s) \)
8: \( E.append(genEdge(S, t)) \)
9: \( visit \leftarrow s \)
10: \( \text{Append trajectory window} \) \( tr \)
11: \( D \leftarrow \text{computeLCSS}(Ne, tr) \)
12: if \( D > \text{thresh} \) then
13: \( \text{Modify CPT of all nodes in } clust \)
14: \( \text{Create new cluster } clus \)
15: \( \text{Modify CPT of all sequences containing } clust \)
16: end if
17: if \( D < \text{thresh} \) then
18: \( \text{Create new cluster } clus \)
19: end if
20: end for
21: end for
22: end for

Algorithm 2: Location prediction - map and update process

Input: User movement graph \( MG \), Present location \( s \), Trajectory log \( T \)
Output: \( < s', Edge - list > \)

1: function MAPPER(s, MG, T) :
2: \( j \leftarrow \text{geo - hascode}(s) \)
3: \( E' \leftarrow \text{extract_pattern}(MG, j) \)
4: for all \( t_i \in T \) do
5: \( L \leftarrow \text{predictLoc}(x(t_i, s)) \)
6: \( p \leftarrow \text{ComputeProb}(s, arraylist[L, t_i]) \)
7: \( dist \leftarrow \text{ComputeTrajCS}(E', t_i) \)
8: \( s' \leftarrow \text{SORT}(arraylist[p], arraylist[dist]) \)
9: end for
10: function Update(s, arraylist[s']) :
11: for all \( t_i \) do
12: for all \( e_a \in arraylist[s'] \) do
13: \( \text{Append trajectory window } T_e \text{ containing } e_a \text{ in } FPN \)
14: \( \text{Modify} \) \( CPT \) \( e_a, t_i \)
15: \( \text{ModifyProb}(e_a, t_i) \)
16: end for
17: end for

- If the RSU’s current load is equals to the maximum load it can handle or an exhaustive computation is required to perform which is beyond the capability of the RSU, it forwards the data to the cloud along with the device ID and request ID. After processing the data, the cloud finds the current location of the device based on the user’s geo-location information (see section 3.1). Based on the current location of the device, the cloud identifies the RSU under which the mobile device is currently located. The cloud sends the result to the RSU along with the device ID and request ID. The RSU forwards the result to the mobile device.
- If the RSU is able to process the raw data and its current load is less than the maximum load it can handle, the RSU processes the data and sends the result to the mobile device. However, as the device is in mobility, it may be possible that the RSU finishes processing and the mobile device moves away. In such a case, the RSU sends the result to the cloud along with the request ID and device ID. The current location of the device is predicted by the cloud based on the user’s geo-location information. Based on the current location of the device, the cloud identifies the RSU under which the mobile device is currently located. The cloud sends the result to the RSU along with the device ID and request ID. The RSU forwards the result to the mobile device.

In our approach as the mobility information is updated dynamically, the probability of the presence of the mobile
device under the predicted RSU is high. However, if the device losses connection with the network for a long duration, there is a probability that the mobile is not located under the coverage of the predicted RSU. In such cases, after receiving the result from the cloud, the predicted RSU sends feedback to the cloud that the mobile device is not present under its coverage. By this time, if the mobile device gets connected with a RSU, it will send a request for the result with the request ID to the RSU. The RSU then forwards it to the cloud. The cloud then sends the result to the mobile device through the RSU and updates the user mobility information accordingly. Algorithm 3 summarizes the steps of the working model of the proposed framework Mobi-IoST. As we observe in the proposed system the cloud performs data processing based on the complexity of the computation required for processing the data and the current load of the RSU. Here, two cases are considered as discussed previously:

- Information processing inside the cloud
- Information processing inside the cloud

The symbols used in the delay and power calculation of the proposed model, are defined in TABLE 1.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u$</td>
<td>Velocity of moving agent</td>
</tr>
<tr>
<td>$D_c$</td>
<td>Amount of data collected and transmitted</td>
</tr>
<tr>
<td>$D_r$</td>
<td>Amount of result data received</td>
</tr>
<tr>
<td>$U_{pmr}$</td>
<td>Data transmission rate in uplink between mobile device and RSU</td>
</tr>
<tr>
<td>$Dw_{cmr}$</td>
<td>Data transmission rate in downlink between mobile device and RSU</td>
</tr>
<tr>
<td>$U_{ptrc}$</td>
<td>Data transmission rate in uplink between RSU and cloud</td>
</tr>
<tr>
<td>$Dw_{trc}$</td>
<td>Data transmission rate in downlink between RSU and cloud</td>
</tr>
<tr>
<td>$f_{mr}$</td>
<td>Uplink data failure rate between mobile device and RSU</td>
</tr>
<tr>
<td>$f_{rm}$</td>
<td>Downlink data failure rate between mobile device and RSU</td>
</tr>
<tr>
<td>$f_{rc}$</td>
<td>Uplink data failure rate between RSU and cloud</td>
</tr>
<tr>
<td>$f_{cr}$</td>
<td>Downlink data failure rate between RSU and cloud</td>
</tr>
<tr>
<td>$d_{pr}$</td>
<td>Data amount processed per unit time by the RSU</td>
</tr>
<tr>
<td>$d_{pc}$</td>
<td>Data amount processed per unit time by the cloud</td>
</tr>
<tr>
<td>$T_{pr}$</td>
<td>Delay to determine the current location of the device</td>
</tr>
<tr>
<td>$P_c$</td>
<td>Power consumption of mobile device in active mode</td>
</tr>
<tr>
<td>$P_i$</td>
<td>Power consumption of mobile device in idle mode</td>
</tr>
</tbody>
</table>

The downlink data transmission delay between mobile device and RSU is given as:

$$T_{rm} = (1 + f_{rm}) \cdot (D_r/Dw_{trc})$$ (7)

The total data transmission delay between mobile device and RSU is given as:

$$T_t = T_{mr} + T_{rm}$$ (8)

The data processing delay inside the RSU is given as:

$$T_{pr} = D_c/d_{pr}$$ (9)

The total delay for data transmission and processing is:

$$T_{tot} = T_t + T_{pr}$$ (10)

The mobile device while located at point $i$ transmits the data to the RSU and requests for processing. Let the radius of the RSU’s coverage area is $R$ and the last visited point of the mobile device inside the RSU is $k$. The delay in movement from point $i$ to $k$ is given as:

$$T_{ik} = \sum_{i=1}^{k-1} \frac{(D_{ik(i+1)})}{u_i}$$ (11)

where $u_i$ is the velocity of the mobile device at location point $i$ and $D_{ik(i+1)}$ is the distance between two consecutive location points $i$ and $i+1$. If $T_{ik} > T_{tot}$, the mobile device is still inside the coverage of the RSU. Hence, the RSU will deliver the result to the mobile device. Hence, the round-trip delay is given as:

$$T_{del} = T_{tot}$$ (12)

The power consumption of the mobile device during this period is given as:

$$P_{del} = T_t \cdot P_a + T_{pr} \cdot P_i$$ (13)

As the RSU performs data processing instead of the cloud, the transmission delay is reduced. Hence, the delay in delivering the result is reduced in our system. Accordingly, the power consumption of the mobile device is also reduced. Else if $T_{ik} \leq T_{tot}$, the mobile device has moved to the coverage of another RSU. Hence, the previous RSU will forward the result to the cloud along with the request ID and the mobile device ID. The cloud contains the mobility information of the mobile devices. Using the location prediction strategy described in the section 3.1.2, the cloud will find out the current probable location of the mobile device. Let $t$ is the time instant when the mobile device is at location $i$. The cloud finds out the location point visited by the mobile device at time instant $(t + T_{tot})$, and the RSU which is currently serving the device. The cloud delivers the result to the selected RSU, which forwards the result to the mobile device. In this case, the round-trip delay is:

$$T_{del2} = T_{tot} + (D_r/U_{ptrc})(1 + f_{rc}) + T_{mp} + (D_r/Dw_{trc})(1 + f_{cr})$$ (14)

where $T_{mp}$ is the delay for determining the current location of the device and correspondingly the RSU currently serving the device, based on the mobility information of the user.
Algorithm 3: Working Model of Mobi-IoST

Input: Raw data received from IoT devices
Output: Result after processing the raw data

1: mobile device receives raw data from IoT devices
2: if mobile device is able to process the data then
3: mobile device works as edge device and processes the data
4: else
5: mobile device forwards the data to the fog device RSU
6: if current load of the RSU < maximum load the RSU can have then
7: go to step 11
8: else
9: go to step 34
10: end if
11: if RSU is able to process the data then
12: RSU processes the data
13: if the mobile device is still connected then
14: RSU delivers result to the device
15: else
16: RSU forwards result to the cloud along with device ID and request ID
17: cloud predicts current location of the mobile device from the mobility information using Algorithm 1 and 2
18: cloud identifies the RSU serving the predicted location
19: cloud forwards the result to the predicted RSU along with the device ID and request ID
20: if the mobile device is connected with the predicted RSU then
21: RSU sends the result to the mobile device
22: else
23: RSU sends a feedback to the cloud that the mobile device is not present in its coverage
24: cloud after receiving the feedback stores the result
25: if the mobile device gets connected with a RSU then
26: mobile device requests for the result to the RSU with the request ID
27: RSU forwards the request to the cloud
28: cloud sends the result to the RSU and updates the mobility information
29: RSU sends the result to the mobile device
30: end if
31: end if
32: end if
33: else
34: RSU sends the raw data along with device ID and request ID to the cloud
35: cloud processes the data
36: go to step 17
37: end if
38: end if

If \( p_s \) and \( p_u \) are the probabilities of the presence of the mobile device under the predicted RSU, the round-trip delay is given as:

\[
T_{del2} = p_s * T_{del21} + p_u * T_{del22}
\]  

(18)

The power consumption of the mobile device during this period is given as:

\[
P_{del2} = p_s * P_{del21} + p_u * P_{del22}
\]

(19)

However, though we have considered the case that the mobile device may not be present under the coverage of the predicted RSU, the probability of this case is very low, because the cloud is dynamically maintaining the user mobility information. As the user current location and the current RSU serving the mobile device is predicted in our system, the delay in delivering the result is reduced. Accordingly, the power consumption of the mobile device is reduced.

4.2 Delay model for Information Processing in Cloud

From the previous subsection the total delay in data transmission between RSU and mobile device (\( T_i \)) has been
determined using equation (10). Now, if cloud performs data processing, then the uplink data transmission delay between RSU and cloud is given as:

$$T_{rc} = (1 + f_{rc}) \times (D_c/U_{p_{trc}})$$

(20)

The downlink data transmission delay between RSU and cloud is given as:

$$T_{cr} = (1 + f_{cr}) \times (D_r/D_{w_{trc}})$$

(21)

Therefore, the total data transmission delay between mobile device and cloud is given as:

$$T_{tn} = T_t + T_{rc} + T_{cr}$$

(22)

The data processing delay inside the cloud is given as:

$$T_c = D_c/d_{pc}$$

(23)

The cloud after processing the data, predicts the current location of the user by analysing the mobility information and accordingly the RSU currently serving the mobile device. After that the cloud sends the result to that RSU along with device ID and request ID. Hence, the round-trip delay is:

$$T_{del31} = T_{tn} + T_c + T_m$$

(24)

where $T_m$ is the delay in predicting the current location and the RSU serving the device currently. The power consumption of the mobile device during this period is given as:

$$P_{del31} = T_t \times P_a + (T_{rc} + T_{cr} + T_c + T_m) \times P_i$$

(25)

If the device is not connected with the selected RSU, the RSU will send a feedback to the cloud. If the mobile device sends request to a RSU for the result, then the cloud will deliver the result to the device through the current RSU. In this case, the round-trip delay is given as:

$$T_{del32} = T_{del31} + T_f + T_{r1} + T_{r2} + (D_r/D_{w_{trc}}) \times (1 + f_{cr})$$

(26)

where $T_f$ is the delay for sending feedback from the RSU to the cloud, $T_{r1}$ is the delay for sending request by a mobile device for result to the RSU, under which the device is present, and $T_{r2}$ is the delay for forwarding the request by the RSU to the cloud. The power consumption of the mobile device during this period is given as:

$$P_{del32} = P_{del31} + T_{r1} \times P_a + (T_f + T_{r2} + (D_r/D_{w_{trc}}) \times (1 + f_{cr})) \times P_i$$

(27)

If $p_s$ and $p_u$ are the probabilities of the presence of the mobile device under the predicted RSU, the round-trip delay is given as:

$$T_{del3} = p_s \times T_{del31} + p_u \times T_{del32}$$

(28)

The power consumption of the mobile device during this period is given as:

$$P_{del3} = p_s \times P_{del31} + p_u \times P_{del32}$$

(29)

However, as the cloud is dynamically maintaining the user mobility information, the probability of the case that the mobile device is not present under the coverage of the predicted RSU is very low. As in the proposed model the RSU under which the user is currently present is predicted, the delay in delivering the result is faster and correspondingly the power consumption of the mobile device is reduced.

5 Performance Evaluation

5.1 Mobility Dataset

The mobility dataset is collected from 100 mobile users from their GPS-enabled smart phones and Google Map timeline for 6 months in the Kharagpur and Kolkata region of India. The dataset consists of timeseries data of GPS traces with the total time-duration of 26,8041 hours. The GPS points are logged in a high-sampling rate of 60-75 secs.

5.2 Experimental Set-up

We aim to demonstrate the efficacy of Mobi-IoST with the real-life mobility dataset. The performance analysis has been carried out in following aspects: (i) movement pattern modelling, (ii) next location (and sequence) prediction and (iii) route prediction given the source and destination pair. Typically, accuracy, recall and $F$-measure are used to evaluate Mobi-IoST and six baseline methods are implemented to compare with our method. 70% of the movement traces are used for modelling, 20% and 10% for testing and validating respectively. Location sequence prediction task is evaluated in different time-scales, from 5 mins to 60 mins. The path prediction task has been carried out in seven different time bins (commuting time) (i) $>10$ mins, (ii) $>10$ and $<15$ mins, (iii) $>15$ and $<20$ mins, (iv) $>20$ and $<30$ mins, (v) $>30$ and $<40$ mins, (vi) $>40$ and $<45$ mins and (vii) $>45$ and $<50$ mins. The intuition is to evaluate the efficacy of Mobi-IoST with respect to different trips with varying commuting time. For this purpose, the trips are divided into such seven classes based on their commuting time. The intuition is to evaluate the efficacy of Mobi-IoST with respect to different trips with varying commuting time. For this purpose, the trips are divided into such classes based on their commuting time. For each such classes, the tenfold cross validation policy has been deployed where all trips within the same class are randomly divided into ten folds, where nine folds are utilized for training and one fold for validation. It guarantees that any trip in the validation set will not appear in the training set. Next, the prediction accuracy for all the trips in the validation set are computed and the average value of the accuracy measure for all seven classes are reported.

5.3 Movement Analysis

The performance measurement of the movement analysis module have been carried out by three measurements, namely, accuracy, recall and $F$-measure. Apart from that, we evaluate the performance of the movement behaviour modelling framework by comparing with six baseline methods, semantic trajectory modelling [16], Bayesian network [29], Longest Common Sub-Sequence (LCSS) [27], Markov chain [30], Convolutional Neural Network Approach [18] and Spatio-temporal Recurrent Neural network (ST-RNN) [17]. It may be noted that the trajectory modelling modules of all of the cited works have been implemented with our dataset to depict the effectiveness of our framework.

3. Sample dataset available: https://drive.google.com/drive/folders/1BpM-K3cI6xYpSHkFe12aGsG8n1AclH4?usp=sharing
One of the major challenges of the proposed framework is to reduce the delay in delivering the processed information, and therefore if new GPS trace comes, the system should be able to re-learn the pattern effectively. TABLE 2 shows the performance measurements (accuracy, learning and re-learning time) compared to six baseline methods. The cardinality (number of trajectory-windows × day) of the test data of each agent for learning and re-learning are 18 × 150 and 6 × 20 respectively. Vlachos et al. [27] propose non-metric similarity function LCSS by computing the similarity between trajectory segments. Although the method outperforms other distance metrics, such as, Euclidean or DTW, but it only calculates the topological or geometrical similarity ignoring the semantics of the trajectories. Semantic enrichment of trajectories and modelling to predict next location has been studied in [16]. Mingqi et al. utilizes the Bayesian network place classifier [29] to categorize the semantic of stay-points. Cheng et al. [30] model the check-in sequences of individuals using matrix factorization method and Markov chain for personalized POI recommendations. Recently researchers are devoted to deploy neural networks [17], [18] to predict the next location sequences accurately. However, the neural network based methods are costly in terms of re-learning and stability.

It is observed from TABLE 2 that our framework, Mobi-IoST outperforms all other baselines, except ST-RNN by approximately 10-18%. Mobi-IoST not only provides next location prediction based on some prediction technique (such as, CNN, RNN or Markov-model), rather it models individual’s movement patterns over days, captures the frequent path followed in several contexts and makes the next location sequence prediction. Further, the learning and re-learning rate as well as stability of our framework is significantly better compared to others. It is observed that the learning and re-learning rates of CNN and ST-RNN are significantly higher by 10-20min and 14-24min than Mobi-IoST respectively. These measurements are important for our case, since the system needs to incorporate any sudden movement pattern change of user effectively. In summary, the deep learning architectures used in the existing works are computationally extensive, and it is shown that such deep architecture may not be beneficial for time-critical applications, where a delay-aware solution is necessary.

Figs. 4-6 show the performance metrics of Mobi-IoST with the existing work [17]. The accuracy to predict stay-point information is represented in Fig. 6. It can be observed that Mobi-IoST has gained 88% to 95% for trajectory stop sequences ranging from 2 to 10. There is a significant drop of accuracy percentage from 93% to 80% of [17] in the same set-up. The key reason behind this observation is the proposed movement modelling named User movement graph, where user movement pattern is modelled in a multi-layer graphical model. The frequent path network (FPN) (layer 4 of the User movement graph) learns the movement paths frequently followed by the user deploying Dynamic Bayesian network. Thus, the model captures all sequences of paths visited in different spatial and temporal contexts. Next, in the proposed model, k-order markov chain is used to effectively model the spatio-temporal regularity of the trajectory sequences, where the next location depends on k recent observations. On the other side, the existing work [17] use deep architecture to capture the spatial and temporal contextual information, however fall short to predict long sequences of trajectory stay-point or stop points. Fig. 4 and Fig. 5 show the recall and F-measure values of location sequence prediction in different time-scale, from 5mins to
60 mins. Since the major aim of this work is the efficient delivery of service while the agent is on move, the experimental evaluation based on the time-stamp value is justified. We have found the recall and F-measure values in the range of 0.95 to 0.81 and 0.96 to 0.78 for twelve time-stamp values respectively. The results indicate that Mobi-IoST not only provides accurate predictions, but performs better than ST-RNN [17] while the time-stamp values increase as shown in figures 4 and 5. The reason behind the consistent prediction result with increased time-stamp value is that we have considered both spatial location and temporal span of a visit-sequence to model the proposed FPN of user movement graph. The prediction algorithm is capable to predict next location sequences based on both recent $k$ locations and the time-duration of each stay-points. Thus, the framework is suitable to predict longer sequences of locations more effectively than existing work. Fig. 7 represents the accuracy of predicting path given source and destination, where x-axis is the commuting time. The experimental set-up is discussed in section 5.2. All of the performance measure values are computed based on the correct number of grid-prediction. The accuracy percentage lies in the range of 89% to 95.3% for seven time-bins.

The performance evaluation is carried out to validate whether Mobi-IoST is suitable to deliver the processed information to the user-device based on user mobility prediction. To assist users and make intelligent decisions in time-critical applications, it is very crucial to model and predict users’ next locations apriori based on varied spatio-temporal contexts. There are several challenges such as (i) how accurately the model predicts next location sequences (not only the immediate next location), (ii) whether the model is capable to accommodate any sudden change of movement pattern of users. It has been observed that our framework has outperformed in all of the performance measurements compared to the baseline methods. The experimental results present that to predict user’ next location Mobi-IoST takes approximately 4.23-9.02sec, which is used in section 5.4 to determine the delay and power consumption in Mobi-IoST.

### 5.4 Delay and Power Consumption

The mobile device sends data to the RSU for processing. The RSU/cloud performs processing and sends back the result to the mobile device. The mobile device may move to the coverage of another RSU before getting the result. In such cases, the connection interruption period is considered 10-30 sec. MATLAB2015 is used for the simulation. The parameter values considered in this analysis are presented in TABLE 3. In this analysis we consider the following two cases:

- Information processing inside the RSU
- Information processing inside the cloud

In the first case, we have considered health parameter data transmitted by the mobile device. Blood pressure level (systolic and diastolic), body temperature, pulse rate and e.g. data are considered. The RSU after processing the health data, sends back the current health status (normal/abnormal) as result to the mobile device. If abnormality is detected, then the parameters which seem to be abnormal are also notified in the result. Here, a preliminary health checking is performed by the RSU to predict the health status. The collected health parameter values are compared to the normal range, e.g. the normal blood pressure range, normal body temperature, normal pulse rate etc. If each of the collected value falls within the normal range with respect to the current ambience and his/her health profile, then health status is predicted as normal. Otherwise, the health status is predicted as abnormal. The amount of data transmission to serve each user request is considered 70-90 KB. The round-trip delay and power consumption of the mobile device (user-device) in the proposed approach, are presented in Fig.8 and Fig.9. The delay and power are compared with the existing mobility-aware task delegation method [3]. In our approach the RSU works as a fog device and performs the data processing. If the device moves to the coverage of another RSU, the cloud predicts the current RSU based on user mobility information. In conventional method, the cloud performs data processing and the user receives the result through RSU. However, if the user gets disconnected due to movement to another RSU, the user has to access the cloud to retrieve the result by serializing session information [3]. But in our approach, the cloud itself sends the result to the RSU, that is currently serving the device. The RSU then forwards the result to the mobile device. Hence, the delay and power consumption of the mobile device in the proposed system Mobi-IoST are less than the existing system [3]. This is observed that Mobi-IoST reduces the delay and power by approximately 23-26% and 37-41% respectively than the existing method [3].

In the second case, we have considered video data

### TABLE 2: Performance comparison of Movement modelling module of Mobi-IoST with baseline methods

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning (min)</td>
<td>84.02%</td>
<td>78.65%</td>
<td>72.93%</td>
<td>80.23%</td>
<td>86.18%</td>
<td>91.7%</td>
<td>93.25%</td>
</tr>
<tr>
<td>(</td>
<td>Traj</td>
<td>W</td>
<td>: 18 × 150)</td>
<td>8.4</td>
<td>10.2</td>
<td>16.8</td>
<td>10.6</td>
</tr>
<tr>
<td>Re-learning (min)</td>
<td>3.8</td>
<td>6.6</td>
<td>14.2</td>
<td>14.2</td>
<td>28.6</td>
<td>18.6</td>
<td>4.2</td>
</tr>
<tr>
<td>(</td>
<td>Traj</td>
<td>W</td>
<td>: 6 × 20)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### TABLE 3: Simulation parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u$</td>
<td>40-85km/hr</td>
</tr>
<tr>
<td>$U_{Prin}$</td>
<td>50Mbps</td>
</tr>
<tr>
<td>$D_{wesr}$</td>
<td>70Mbps</td>
</tr>
<tr>
<td>$U_{Prin}$</td>
<td>75Mbps</td>
</tr>
<tr>
<td>$D_{wesr}$</td>
<td>150Mbps</td>
</tr>
<tr>
<td>$f_{mr}$</td>
<td>0.008-0.25</td>
</tr>
<tr>
<td>$f_{min}$</td>
<td>0.008-0.25</td>
</tr>
<tr>
<td>$f_{fc}$</td>
<td>0.05-0.5</td>
</tr>
<tr>
<td>$f_{ps}$</td>
<td>0.05-0.5</td>
</tr>
<tr>
<td>$d_{pc}$</td>
<td>500Mbps</td>
</tr>
<tr>
<td>$d_{pc}$</td>
<td>1Gbps</td>
</tr>
<tr>
<td>$Tr$</td>
<td>4.23-9.02sec</td>
</tr>
<tr>
<td>$P_a$</td>
<td>0.11W</td>
</tr>
<tr>
<td>$P_t$</td>
<td>0.055W</td>
</tr>
</tbody>
</table>
transmitted by the mobile device. The RSU after processing
the video data, sends back the processed data to the
mobile device. The amount of transmission to serve each
user request is considered 2-20 MB. The round-trip delay
and power consumption of mobile device in the proposed
approach, are presented in Fig.10 and Fig.11. The delay and
power are compared with the existing mobility-based task
delegation method [3]. In our approach the cloud performs
the data processing. After that based on user geo-location
information, the cloud predicts the current location of the
user and the RSU currently serving the device. The cloud
forwards the result to the RSU, which then sends back the
result to the mobile device. However, in the existing method,
the cloud performs data processing and the user receives the
result through RSU after accessing the cloud. Moreover, if
the user gets disconnected and moves to the coverage of an-
other RSU, the user has to access the cloud through the new
RSU to retrieve the result by serializing session information.
Whereas in our approach, the cloud itself sends the result
to the RSU, that is currently serving the device. The RSU
then forwards the result to the mobile device. Hence, the
delay and power consumption of the mobile device in the
proposed system are less than the existing system [3]. This
is observed that the proposed system reduces the delay and
power by approximately 55-60% and 57-74% respectively
than the existing system [3]. This is observed that for small
as well as large scale processing, our Mobi-IoST reduces the
delay and power consumption of the mobile device. As a
result, the QoS is enhanced.

6 CONCLUSIONS

Seamless connectivity is a major challenge during data and
computation offloading in any mobile network. The process-
ing of raw data collected using IoT devices and delivery of
the result to the client mobile device becomes a challenge
if the client frequently changes location. In this paper, we
have proposed a real-time cloud-fog-edge IoT collaborative
framework, namely Mobi-IoST, for efficiently delivering the
processed information to the user-device based on user mo-
bility prediction and intelligent decision making. The mobile
device acts as an edge device, and the RSU is used as fog
device for processing the raw data collected by the mobile
device from the IoT devices. If the user changes location and
gets disconnected from the RSU, it forwards the result to
the cloud. The cloud determines the current location of the
user based on the mobility information and delivers the re-
sult accordingly. The mobility prediction module primarily
stores the movement traces and models the frequent path
followed by the individual in different contexts. Further,
it deploys a hidden markov model based location predictor
for efficiently predicting the location sequences. The real-
life data of movement traces yield approximately 10-18%
improvement compared to other existing methods. More-
over, the simulation results demonstrate that the proposed
fog computing framework reduces the delay and power by
approximately 23-26% and 37-41% respectively than the ex-
isting mobility-aware task delegation system. In the future,
the Mobi-IoST framework will be extended to capture the
cellular data usage patterns from such time-series data and
prediction of location sequences may help to appropriately
manage the power and bandwidth related resources. The proposed Mobi-IoST will act as a foundation of mobility-aware network resource management in the future.

References


Shreya Ghosh received the B.Tech. degree from IIT Kharagpur, India, in 2015. She is currently a Research Scholar with the Department of Computer Science and Engineering, IIT Kharagpur, India working towards her Ph.D. Her current research interests include spatial informatics, trajectory data mining and cloud computing. Shreya is the recipient of TCS fellowship.

Anwesha Mukherjee received M.Tech. and Ph.D. degrees from the Department of Computer Science and Engineering, West Bengal University of Technology, India, in 2011 and 2018, respectively. She is currently working as Research Associate in the computer science department of IIT Kharagpur. Her research areas include IoT, Fog computing, mobile network and mobile cloud computing. She has received Young Scientist Award from International Union of Radio Science in 2014 at Beijing, China.

Soumya K Ghosh is currently a Professor with the Department of Computer Science and Engineering, IIT Kharagpur. He was with the Indian Space Research Organization, Bengaluru, India. He has authored or coauthored more than 200 research papers in reputed journals and conference proceedings. His current research aims include spatial data science, spatial web services, and cloud computing.

Rajkumar Buyya is a Redmond Barry Distinguished Professor and Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. He is also serving as the founding CEO of Manjrasoft, a spin-off company of the University, commercializing its innovations in Cloud Computing. He has authored over 625 publications and seven text books including “Mastering Cloud Computing” published by McGraw Hill, China Machine Press, and Morgan Kaufmann for Indian, Chinese and International markets respectively. He is one of the highly cited authors in computer science and software engineering worldwide (h-index=118, g-index=245, 73,200+ citations). Dr. Buyya is recognized as a “Web of Science Highly Cited Researcher” in 2016 and 2017 by Thomson Reuters, a Fellow of IEEE, and Scopus Researcher of the Year 2017 with Excellence in Innovative Research Award by Elsevier for his outstanding contributions to Cloud computing.