The Journal of Supercomputing (2023) 79:17495–17518
https://doi.org/10.1007/s11227-023-05300-5

Mobi-Sense: mobility-aware sensor-fog paradigm for mission-critical applications using network coding and steganography

Anwesha Mukherjee[1] · Shreya Ghosh[2] · Soumya K. Ghosh[3] · Rajkumar Buyya[4]

Accepted: 12 April 2023 / Published online: 8 May 2023

Abstract
Mission-critical applications refer to the real-time applications, which require fast and secure service provisioning, such as defense sector and disaster management. This paper proposes a delay-aware and secure service provisioning model for such types of applications. As a use-case, we have considered the defense sector, which is a vital sector for a country's all-round well-being including security, safety, society, and economy. In the conventional sensor-cloud model, the sensor data is stored and processed in the cloud. However, the sensor nodes have small coverage and the use of the long distant cloud servers increases the delay. Therefore, the conventional sensor-cloud model may not be efficient for defense application. Moreover, data hiding for security purposes is another important aspect of this field. To address these challenges, this paper proposes a mobility-aware sensor-fog paradigm for mission-critical applications based on network coding and steganography, referred to as *Mobi-Sense*. In Mobi-Sense, steganography is used for hiding the data during transmission. The theoretical results demonstrate that Mobi-Sense outperforms the existing frameworks with respect to delay and power consumption by $\sim(40-80)\%$. The simulation results present that Mobi-Sense reduces the delay by $\sim(18-40)\%$ than the conventional sensor-cloud framework for mission-critical applications. An optimal path finding algorithm based on deep learning has been deployed in the context of disaster scenario. The experimental analysis shows that the proposed optimal path finding method achieves precision and accuracy above 90%. This is observed that our proposed modules have outperformed existing baselines in terms of accuracy, delay, and power consumption.

Keywords Mission-critical · Sensor-fog · Delay-aware · Power-aware · Data hiding · Path extraction

✉ Shreya Ghosh
shreya@psu.edu

Extended author information available on the last page of the article

1 Introduction

The mission-critical applications are real-time applications and require fast and secure service provisioning. The defense sector and disaster management are two examples of such applications. In this paper, we have focused on the delay-aware and secure service provisioning for mission-critical applications. We have considered the military or defense sector as a use-case. Usually, military sector is categorized into three sub-sectors: army, air force, and navy. Mission-critical applications related to such sectors require continuous monitoring of different regions of the country specially the sensitive regions. If any adverse situation occurs, prompt action needs to be taken. Therefore, continuous monitoring and fast service provisioning are highly recommended for such applications. In the existing methods for military services, sensor-cloud paradigm is generally used. There are different types of sensor nodes used in the military sector, for example, motion sensor, Global Positioning System (GPS) sensor, infrared sensor, gas sensor, light sensor, ultrasonic sensor, etc. These sensors are used to collect the environmental object status of a region [1]. In the conventional sensor-cloud paradigm, the sensor nodes collect data and transmit to the cloud for processing and storage [2, 3]. However, the conventional sensor-cloud model has the following drawbacks:

- The sensor nodes have small coverage and limited battery life.
- The use of long distant remote cloud increases the delay in service provisioning.

Hence, the conventional sensor-cloud model may not be efficient for mission-critical applications. Another important issue of military sector is data security. Hiding of data during transmission is highly recommended for military sector. To address all the issues, we propose network coding and steganography-based sensor-fog paradigm in this paper. As the sensor nodes have small coverage and limited battery life, relay nodes are used in the sensor network. In the proposed framework, network coding [4, 5] is used during data transmission by the relay nodes, steganography [6–8] is used to hide the data during transmission to the fog node and cloud, and fog computing [9–11] is used to reduce the delay in service provisioning.

1.1 Motivating example

Figure 1 illustrates a motivating example, where sensitive information of the defense sector (military data) is being transferred among army camps. As shown in the figure, dotted lines (red) illustrate the affected region, where road network has been affected. Such vulnerable regions should be avoided. Road side units (RSUs) send information regarding vulnerable regions and any other sensitive information along with predicted path (with low risk) to the nearby camp or volunteers. These information is required to send using secure data transmission approach. Now, the requirements are:

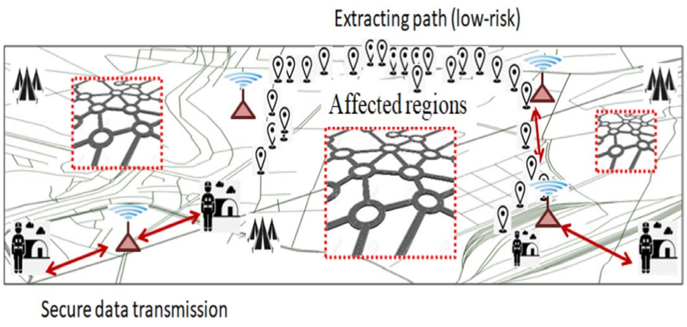


Fig. 1 Motivating scenario of proposed framework

- The data must be protected as the data is confidential.
- Based on the collected and processed sensor data, continuous monitoring is required. If any exigency is detected, that information needs to be communicated to the relief centres.
- In case of emergency, relief materials need to be sent to the victim region in minimal time, and the volunteers also need to take the route with minimal risk (avoiding road-blockage) to reach the destination.

1.2 Contributions

The key contributions of the paper are:

- A mobility-aware sensor-fog paradigm is proposed for mission-critical applications, named as *Mobi-Sense* (Mobility-aware Sensor-Fog Paradigm for Mission-Critical Applications using Network coding and Steganography). The sensor-fog framework is implemented in iFogSim. The theoretical and simulation results show that Mobi-Sense has lower delay and power consumption than the existing frameworks.
- In Mobi-Sense, network coding is used during data transmission by the relay nodes, and image steganography is used to provide data confidentiality during transmission to the fog node and cloud servers.
- A risk-modelling module is proposed by mining road-graph data and other contextual information using *opportunistic network*. A deep learning-based path recommendation algorithm is presented for finding the optimal path to reach the victim region in minimal time with minimal risk in case of exigency situation. The experimental analysis demonstrates that Mobi-Sense determines the optimal path with better accuracy than the existing models.

Organization of the paper: The existing works related to the problem is discussed in Sect. 2. We demonstrate the proposed paradigm Mobi-Sense in Sect. 3, and Sect. 4 presents the delay and power consumption model of Mobi-Sense. In Sect. 5, we present the experimental evaluations and the conclusion is offered in Sect. 6.

2 Related work

Cloud computing has gained its popularity for on-demand service provisioning in various sectors, such as healthcare, home monitoring, and military. Internet of Things (IoT) is a key component of smart applications in different fields, such as healthcare and agriculture [12]. The traditional wireless sensor network (WSN) alone cannot support the primary requirements of smart applications, such as data collection at low latency, parallel processing, and dynamic resource sharing [13]. In such a case, integration of WSN with cloud virtualization is required [13]. The use of IoT in military sector has been highlighted in [14]. A cloud-based paradigm for military tri-services has been proposed in [2]. However, the use of remote cloud may enhance the delay in service provisioning [1, 15], which can be fatal for such critical sectors. For time-sensitive task scheduling the fog-cloud architecture plays an important role [16]. For time-critical applications like health care, fog computing is widely used nowadays [17, 9, 18]. The use of fog computing for mission-critical applications has been discussed in [1], and the authors have showed that the use of fog computing reduced the delay than the cloud-only paradigm. In the sensor-fog-cloud framework for defense sector, the geolocation information is very important because if any adverse situation occurs, then the optimal path to the victim region has to be found. In such a situation, the existing well-known path may not be available due to road-blockage. Hence, an alternate path has to be found to reach the victim region in minimal time with minimal risk (avoiding road-blockage). For health care application, optimal path finding method has been discussed in [18]. The physical layer security for military IoT links have been highlighted in [19]. The enabling of civil-military collaboration for disaster management in smart city scenario has been discussed in [20]. The use of IoT-fog architecture for military sector with an emphasis on energy conservation has been discussed in [21]. In the present work, the geospatial information is analysed to reach the affected region from the source during disaster management. The data related to a place if is represented in terms of geographic coordinates, it is referred as geospatial information [22, 23]. The cloud usually stores, process, and analyses the geospatial information as the data volume is huge [24].

Data confidentiality is a critical aspect of defense application. The security issues of military cloud have been discussed in [25]. For secure military cloud application a hybrid routing protocol has been discussed in [26]. In [1], the security issues in fog-based mission-critical applications have been discussed. However, the data hiding process has not been highlighted in [1]. In the present work, we use steganography for hiding data while transmission takes place to the fog and to the cloud. In the present work, we have also proposed a deep learning-based method for optimal route identification to the victim region in case of adverse situation. In Table 1, the proposed framework (Mobi-Sense) is compared with the existing frameworks for mission-critical applications. Table 1 shows that Mobi-Sense uses fog computing, network coding, and steganography together for the first time in the field of mission-critical applications. The use of steganography provides data confidentiality during transmission, and fog computing helps to

Table 1 Comparison of proposed framework with existing frameworks for mission-critical applications

Work	Contribution	For defense/ military sector	Fog device is used	Network coding is used	Steganography is used	Data analysis for optimal path selection with high accuracy and precision using deep learning
Misra et al. 2014 [2]	For military tri-services a sensor-cloud framework has been proposed	✓	×	×	×	×
Burmaoglu et al. 2019 [14]	The use of IoT in military sector has been discussed	✓	×	×	×	×
Mishra et al. 2019 [1]	A sensor-fog model has been proposed for mission- critical applications	✓	✓	×	×	×
Bichi et al. 2022 [21]	For military- based IoT- fog architecture, sequential and master-worker application modules have been presented and analyzed with respect to energy conservation	✓	✓	×	×	×
Sulyman and Henggeler 2022 [19]	The designing methods for multiple input multiple output- beamforming systems for physical layer security enhancement for military IoT links have been discussed	✓	×	×	×	×
Campioni et al. 2023 [20]	For disaster relief operations in smart city scenario, civil- military collaboration has been enabled	✓	✓	×	×	×
Mobi-Sense (proposed work)	A steganography and network coding-based sensor-fog model is proposed for mission-critical applications	✓	✓	✓	✓	✓

reduce the delay in service provisioning. Another unique feature of Mobi-Sense is to use deep learning for selecting optimal path to the affected region in emergency situation. The use of network coding, steganography, deep learning, and fog computing together makes Mobi-Sense novel and advantageous with respect to the existing models.

3 Mobi-Sense: proposed framework

In this section, we will discuss the architecture and mathematical model of Mobi-Sense (refer to Sect. 3.1), the use of network coding and steganography in Mobi-Sense (refer to Sect. 3.2), and the optimal path finding method to reach the victim region in case of emergency situation (refer to Sect. 3.3).

3.1 Architecture and mathematical model

In Mobi-Sense, the sensor nodes are spatially distributed in a region, and attached with various objects to collect their status continuously. The sensor nodes are connected with the sensor network director (SND), which receives data continuously from the sensor nodes. If there are X sensor nodes, there will be Y number of SNDs, where $Y < X$, i.e. under each SND there are multiple sensor nodes. Here, it may be noted that due to limited coverage and battery life of the sensor nodes, relay nodes may be used. In that case, the data transmission from sensor nodes to the SND will take place through relay nodes. The SNDs are connected with the sensor fog organizer (SFO). Multiple SNDs are connected with a SFO, i.e. if there are Y SNDs and Z SFOs, then $Z < Y$. The SFOs are connected with the cloud. In this framework, different types of sensor nodes can be used for environmental objects' status detection, such as ultrasonic sensor, temperature sensor, humidity sensor, light intensity detecting sensor, GPS sensor, and explosive material detecting sensor. The sensor-fog paradigm is presented in Fig. 2, and the

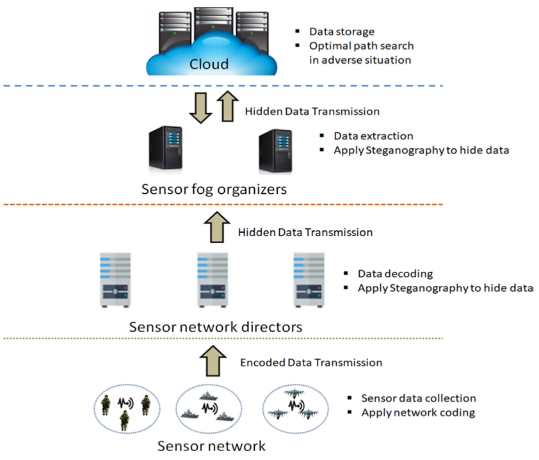


Fig. 2 Sensor-fog paradigm for mission-critical applications

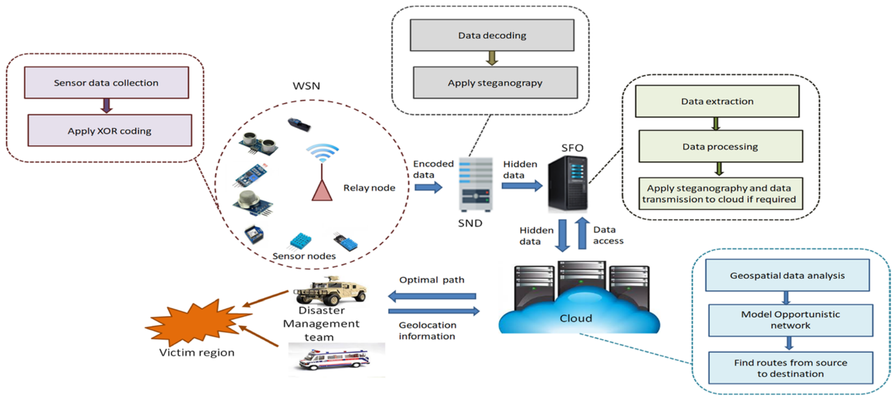


Fig. 3 Working flow diagram of proposed framework Mobi-Sense

working model is presented in Fig. 3. As observed from the figures, the sensor and relay nodes are connected with SND, the SND is connected with SFO, and the SFO is connected with the cloud.

The components of Mobi-Sense (sensor node, relay node, SND, SFO, and cloud computing instance) are mathematically defined here.

- *Definition 1: Sensor node (Sensor):* A sensor node is defined as,

$$\text{Sensor} = \langle \text{ID}_{\text{sensor}}, \text{Obj}_{\text{sensor}}, \text{Geo}_{\text{sensor}} \rangle \tag{1}$$

  where $\text{ID}_{\text{sensor}}$ represents the unique ID of the sensor node, $\text{Obj}_{\text{sensor}}$ denotes the object type, and $\text{Geo}_{\text{sensor}}$ denotes the geolocation information of the sensor node.
- *Definition 2: Relay node (Relay):* A relay node is defined as,

$$\text{Relay} = \left\langle \text{ID}_{\text{relay}}, \text{Geo}_{\text{relay}} \right\rangle \tag{2}$$

  where ID_{relay} represents the unique ID of the relay node and $\text{Geo}_{\text{relay}}$ denotes the geolocation information of the relay node.
- *Definition 3: Sensor Network Director (SND):* A SND is defined as,

$$\text{SND} = \langle \text{ID}_{\text{SND}}, \text{HW}_{\text{SND}}, \text{Geo}_{\text{SND}}, \text{Sec}_{\text{SND}} \rangle \tag{3}$$

  where ID_{SND} represents the unique ID of the SND, HW_{SND} denotes the hardware specification of the SND, Geo_{SND} denotes the geolocation information of the SND, and Sec_{SND} denotes the security scheme adopted by the SND.
- *Definition 4: Sensor Fog Organizer (SFO):* A SFO is defined as,

$$\text{SFO} = \langle \text{ID}_{\text{SFO}}, \text{HW}_{\text{SFO}}, \text{Geo}_{\text{SFO}}, \text{Sec}_{\text{SFO}} \rangle \tag{4}$$

  where ID_{SFO} represents the unique ID of the SFO, HW_{SFO} denotes the hardware specification of the SFO, Geo_{SFO} denotes the geolocation information of the SFO, and Sec_{SFO} denotes the security scheme adopted by the SFO.

- *Definition 5: Cloud computing instance (CI):* A cloud computing instance is defined as,

$$\mathrm{CI} = \left\langle \mathrm{ID}_{\mathrm{cloudcomp}}, \mathrm{IDs}_{\mathrm{proc}} \right\rangle \tag{5}$$

  where $\mathrm{ID}_{\mathrm{cloudcomp}}$ denotes the cloud component ID and $\mathrm{IDs}_{\mathrm{proc}}$ represents a set that contains the processing unit IDs of all the necessary cloud servers of the instance.

Here, the mapping from sensor/relay nodes to SND is *many-to-one*, the mapping from SND to SFO is *many-to-one*, and the mapping from SFO to cloud computing instances is *many-to-many*. In Sect. 4, delay and power consumption of the Mobi-Sense (considering the sensors, relay nodes, SND, SFO, and cloud) are discussed. In Sect. 5, the sensor-fog model of Mobi-Sense is implemented in iFogSim to evaluate the performance.

3.2 Application of network coding and steganography

In Mobi-Sense, network coding is used during data transmission by the relay nodes, and steganography is used to hide the data during transmission to the fog node and cloud.

3.2.1 Application of network coding

As sensor nodes have small coverage area and limited battery life, relay nodes may be used. In that case, relay node applies XOR network coding [4] while transmitting data. A relay node maintains two queues: one queue (Q_s) containing sending data packets and another queue (Q_r) containing receiving data packets. If Q_s contains data packets, XOR network coding is performed between one packet picked from Q_s and one packet picked from Q_r. Otherwise, two consecutive packets from Q_r are picked, and XOR network coding is performed. The encoded packet is transmitted. The SND when receives an encoded packet, it decodes and retrieves the original data packet.

3.2.2 Application of steganography

In the proposed architecture, we have used steganography-based data transmission from the SND to SFO, and from the SFO to cloud servers. We have observed that the data from multiple sensors are received by the SND. The SND accumulates and organizes the data, and forwards to the SFO. In Mobi-Sense, before forwarding the data to the SFO, the sensor data are hidden inside an image using steganography. Least Significant Bit (LSB)-based steganography is a popular approach of data hiding [27, 28]. In [28], modified LSB-based audio steganography has been used. In our work, we encode the sensor data using XOR and one's complement, and then hide the encoded data into the image using XOR-based LSB image steganography. The image is sent to the SFO. The SFO extracts the encoded data from the image, decodes the data, and performs processing on the data. If any abnormality is detected, the processed sensor data is encoded using XOR and one's complement, and then hidden inside an image using XOR-based LSB image steganography, and sent to the cloud. The cloud extracts the encoded data from the image and decodes the data. Inside the cloud spatial data analysis is performed, and the optimal path is extracted using the method described in Sect. 3.3. As in our approach, we first perform encoding on the original sensor data and then the encoded data is hidden inside the image, we have referred the approach as modified LSB-based image steganography. As the latency is a vital factor for real-time applications, we have used a simple encoding approach based on XOR and one's complement.

Modified LSB-based image steganography: In LSB-based image steganography method, the last bit of a pixel of an image is modified without affecting the visible change in the colour [27]. In our approach, we first encode the data using XOR and one's complement, and then use XOR-based LSB image steganography to embed the encoded data into the image. The process is stated in Algorithm 1. For decoding the image, first the number of pixels containing the data is calculated. After that one pixel of the image is traversed at a time. The LSB of each pixel is extracted, and the encoded data packet is generated subsequently. After that decoding is performed on the encoded data packet and the original data packet is obtained. Use of the modified LSB-based steganography helps to hide the data during transmission to fog node and cloud servers.

Algorithm 1 Modified LSB-based Image Steganography for Mobi-Sense

```
Input: Sensor data, Input image
Output: Output image
 1: transform the input image into gray scale
 2: resize the image if required
 3: if the sensor data is not in binary form, convert it into binary form then
 4:     $endata_i \leftarrow data_i \oplus data_{i+1}$ where $0 \leq i < length(data) - 1$   ▷ XOR is performed between consecutive bits of sensor data, $length(data)$ denotes number of bits in sensor data, $endata$ denotes data obtained after XOR-ing
 5:     $Encodata \leftarrow One's\ complement(endata)$   ▷ One's complement $endata$, $Encodata$ denotes encoded data
 6:     initialize the output image to the input image
 7:     for each pixel of the image do
 8:         convert the pixel value to respective binary form
 9:         take the next bit of $Encodata$
10:         take a variable $v$
11:         $v \leftarrow db \oplus pixlsb$   ▷ XOR the encoded data bit $db$ with the LSB of the pixel $pixlsb$
12:         $pixop \leftarrow pixip + v$   ▷ pixel value of the output image $pixop$ is updated to (pixel value of the input image $pixip + v$)
13:     end for
14:     update the output image until all bits of $Encodata$ are embedded
15:     send the output image to the destination node
16: end if
```

As the image data is transmitted, the size of the image will be a parameter while data load is considered. The time consumption in embedding the data into an image and extracting from the image will be considered while calculating the total delay for the proposed framework. Here, the authors wish to mention that audio/video-based steganography [29] can also be used to hide the data. However, the size of an image file is usually lower than an audio or video file, therefore considering the data load we have used image-based steganography in Mobi-Sense.

3.3 Optimal path selection strategy

One of the major components of Mobi-Sense is finding an optimal path to avoid the regions with risk (or affected areas). We have deployed an *opportunistic network* of the study region. The conventional opportunistic network [30] is a network, where the nodes are wirelessly connected and the connection is temporary. Due to mobility or other issues like node deactivation, the network topology changes with time. In this work, we form the opportunistic network with the sensor nodes, and each of the SND has the road-network information between any two such connected regions. The SND sends the information to the connected SFO, and forwards it to the cloud. Since the optimal route finding algorithm is compute intensive, Mobi-Sense deploys the algorithm in the cloud server. Algorithm 2 presents the basic steps of movement network modelling. Here, for sensor node we have used the term sensor node, and by node we have denoted the node of road network. The road network is represented by a directional graph, where road-segments are edges and intersection points of the road-segments are denoted as nodes.

```
Algorithm 2 : Movement network generation
Input: Set of sensor node location $G$
Output: Movement network $< Traj\_Window(V, E, \Upsilon) >$
 1: $S, V, E \leftarrow NULL$;    ▷ S: Stay-point, V: Nodes, E: Edges (Road network)
 2: for $each\ GPS\ point\ p_i \in G$ do
 3:     j ← i+1
 4:     for $each\ GPS\ point\ p_j \in G$ do
 5:         $S_j \leftarrow$ Compute (boundingbox, time-interval)
 6:         $S_j \leftarrow compute\_geotagg()$
 7:         $S.insert(S_j)$
 8:     end for
 9: end for
10: for $each\ trajectory\ trace\ tr \in T$ do
11:     for $each\ un - visited\ sensor - node\ s \in S$ do
12:         $V.append(s)$    ▷ A new node with sensor node is created
13:         $visited \leftarrow s$
14:         $CPT \leftarrow$ Create Conditional Probability table(s)
15:         $t \leftarrow extractTemporal(s)$    ▷ Time interval of the edge creation between two such sensor nodes
16:         $E.append(dirEdge(S, t))$
17:     end for
18: end for
```

We propose a deep architecture to extract path to reach the destination considering: (a) finding paths without any risk by analysing all environmental features and contexts, and (b) avoiding the affected regions. The overall process has three stages: (i) *mobility-feature set preparation*, (ii) *mobility feature learning*, and (iii) *path prediction.*

Deep Neural Network (DNN), a feed-forward neural network, is not capable to deal with context-shifts, since it does not have any feedback loop [31, 32]. On the other hand, the Recurrent Neural Network (RNN) feeds the activations from historical observations, thus, influences the classification in the present time [33]. This internal states of the architecture are capable to capture long-term dependency of contextual information. Briefly, *Long Short Term Memory (LSTM)* is suitable for learning the long term dependency of the time-series data, and overcomes the limitations of conventional RNN, such as vanishing gradient and exploding gradient issues.

Our framework utilizes *deep RNN* architecture, where multiple networks are stacked for annotating the trajectory segments. Initially, in level 1, all the segments of the study-area are passed through the input layer and affected regions are clustered together. In the next step (level 2), each path is weighted based on the risk-factor. Finally, the route with the minimal risk is selected to reach the destination.

The first step is to extract affected regions, for which the road-segments are not-reachable. The region is segmented into fixed-length *traj_slider*s, and each traj_slider contains one sensor-node. This *traj_slider* is utilized as a sliding window, and the environment behaviour changes are captured. For example, the attributes can be *noise-intensity* (t_{sd}), *light-intensity* (Δv), and *timestamp*. Similarly, we extract the other features (F) (impacts on the neighboring regions) by traversing the record of the *traj_slider*. In the next step, the fixed length deep representation of the trajectory is generated by computing the differences of the features.

The initial phase utilizes a *Gated recurrent unit (GRU)*, which is similar to *LSTM* [34]. GRU has reset and update gates, which are formally defined as:

$$\begin{aligned} z_t &= \sigma(W_z \cdot [h_{t-1}, x_t]) \quad r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \\ h_t &= (1 - z_t) * h_{t-1} + z_t * \tanh(W \cdot [r_t * h_{t-1}, x_t]) \end{aligned} \tag{6}$$

where W_z, W_r, and W are weight matrices. The *update gate* (z_t) helps to extract the required information from the past time-step, and pass to the future, h_t and x_t are forget gate and input gate, respectively. It is specifically important to remember the context or model long-term dependencies. The *reset gate* (r_t) and forget gates are crucial to capture the context-shift problem of the movement behaviours, since it eliminates the vanishing gradient problem, but passes the relevant information to the next steps of the network.

Finally, a *softmax* function is used and the representation (l_{Tr}) produced by the previous block is fed into the next layer.

$$l_{Tr_i} = \text{softmax}(W_{Tr_i} h_{Tr_i} + b_{Tr_i}) \tag{7}$$

where W_{Tr_i} is the weight matrix for layer i that is applied to the hidden representation (h_{Tr_i}) in the model. h_{Tr_i} is the hidden representation of the road-sequence (l_{Tr}) at layer i. The bias term for layer i is defined as b_{Tr_i}. The parameter set (ξ) including the weight metrics are learnt as follows: Given a road-sequence (l_{Tr}) and the semantic labels (risk factor) S_l, the log-likelihood is maximized with respect to ξ:

$$S(l_{Tr}) \vdash \sum_{l_{Tr} \in Tr'} \log p(S|l_{Tr}, \xi) \tag{8}$$

where Tr and Tr' are the labelled trajectory segment and training data, respectively. $p(S|lT_r, \xi)$ is the probability of the semantic labels (risk factors) S given the road-sequence (l_{Tr}) and the parameter set ξ. The parameter set is estimated by deploying *stochastic gradient descent*:

$$\xi \leftarrow \xi + \mu \frac{\partial \log p(S|l_{Tr}, \xi)}{\partial \xi} \tag{9}$$

where μ is the learning rate. $\partial \log p(S|l_{Tr}, \xi)/\partial \xi$ is the gradient of the log-likelihood with respect to the parameter set ξ, used to update the parameters during optimization.

The representation learning module encodes different mobility semantics at varied contexts. Then, the *GRU* and *softmax* modules map the trajectory segments to appropriate semantic labels and point of interests (POIs). The aggregated movement pattern over the underlying road network is initially mapped to find out the footprint deviation in different temporal scales in varied POIs. To achieve this, the region of interest (ROI) is divided into different uniform grids (gr), and the time period is divided into n slots. This is a necessary step since the GPS trajectory traces are timestamped continuous numerical variables. The moving agents (people or vehicle) depart from a region (POI), arrive in the destination (POI). All of these trips comprise the overall aggregated movement flow of the complete study region. This mobility flow represents the semantics of the region, and helps to identify different optimal paths in a region. First, convolutional neural network (CNN) is used to model the mobility flows in different regions and in fixed timestamp. Several layers of CNNs are stacked. Each CNN takes the input of aggregated GPS footprints (ag_t) and represented by:

$$AG^l_{i,t} = ReLU(W^k * AG^{l-1}_{i,t} + b^k) \tag{10}$$

where ReLU (rectified linear unit) is the activation function. $AG_{i,t}$ represents the flow matrices including all footprint count of the grid gr_i. All L CNN layers are stacked into a fully connected layer. The representation of grid (gr_i) is $ag_{i,t}$, and b^k is the bias-vector. Next, a LSTM network is used to capture the spatial relations of mobility flow in different temporal scales.

$$gr_{i,t} = LSTM([ag_{i,t};c_{i,t}], gr_{i,t-1}) \tag{11}$$

where $gr_{i,t}$ is the output representation of grid gr_i considering spatio-temporal relation and context-information ($c_{i,t}$). Then, we append different penalty-values for each instance to learn the model. Finally, after this step, a *softmax layer* is implemented to compute the optimal path considering all risk-factors and minimal time to reach the destination. Algorithm 3 presents the path extraction process.

Algorithm 3 Path extraction in emergency situation

Input: Data modalities (D_1, D_2, D_3) from sensor nodes and GPS log (G)
Output: Optimal path to reach the destination

```
1: Construct Mobility network with senor nodes $Sensor_i$ with last 1 hour data
2: Construct Mobility network with sensor nodes $Sensor_j$ with available data of present timestamp ▷ Missing links and labels are generated
3: Uniform Grid Construction
4: Aggregated Movement flow generation
5: Spatio-temporal feature generation of Movement Flow ▷ CNN layer construction
6: Extracting and representing correlations of Movement Flow in varied temporal scales and contexts ▷ LSTM network construction
7: Convert the mobility features of $Sensor_i$ in fixed-length representation
8: Minimize the risk value
9: Predict the path with minium risk ▷ Softmax layer
10: Refinement of the path using available few sensor-node data
```

4 Delay and power consumption in Mobi-Sense

In this section, the total delay and the total power consumption in Mobi-Sense are mathematically determined. Table 2 presents the parameters used in this calculation.

The delay in data collection by sensors, reception, encoding, and transmission by relay nodes, and reception by the SND (considering all the objects), is given as,

$$Lt_{\mathrm{SNDO}} = max(Lt_1, Lt_2, .., Lt_{N_o}) \tag{12}$$

Total delay of the paradigm is calculated as the sum of the delays in data collection, processing, transmission, reception, encoding, and decoding by all the components, given as,

$$\begin{aligned} Lt_{tot} =& Lt_{SNDO} + Lt_{dec} + Lt_{pSND} + Lt_{sten} \\ & + Lt_{SNDSFO} + Lt_{ext} + Lt_{pSFO} + Lt_{enSFO} \\ & + Lt_{SFOC} + Lt_{pC} + Lt_{popt} \end{aligned} \tag{13}$$

Total power consumption of the paradigm is calculated as the sum of the power consumption by all the components for data collection, processing, transmission, reception, encoding, and decoding, given as,

Table 2 Parameters used for delay and power calculation

Parameter	Definition
Lt_{oj}	Total delay in data collection by sensor, reception, encoding, and transmission by relay nodes, and reception by SND for object *oj*
N_o	Number of objects under consideration
Lt_{dec}	Delay in decoding sensor data by SND
Lt_{pSND}	Delay in accumulating and organising sensor data by SND
Lt_{sten}	Delay in encoding and embedding data into image by SND
Lt_{SNDSFO}	Delay in data transmission from SND to SFO
Lt_{ext}	Delay in extracting data from image and decoding by SFO
Lt_{pSFO}	Data processing delay at SFO
Lt_{enSFO}	Delay in encoding and embedding data into image by SFO
Lt_{SFOC}	Delay in data transmission from SFO to cloud
Lt_{pC}	Delay in data extraction from image, decoding, and processing at cloud
Lt_{popt}	Data processing delay for finding optimal path
$P_{Sensors}$	Power consumption of all the sensor nodes (including relay nodes) for data collection, transmission and reception
P_{rSND}	Power consumption of SND in data reception
P_{deSND}	Power consumption of SND in data decoding
P_{pSND}	Power consumption of SND in data processing (accumulation and organization)
P_{enSND}	Power consumption of SND in data encoding and embedding into image
P_{tSND}	Power consumption of SND in data transmission
P_{rSFO}	Power consumption of SFO in data reception
P_{deSFO}	Power consumption of SFO in data extraction from image and decoding
P_{pSFO}	Power consumption of SFO in data processing
P_{enSFO}	Power consumption of SFO in data encoding and embedding into image
P_{tSFO}	Power consumption of SFO in data transmission
P_{rC}	Power consumption of cloud in data reception
P_{pC}	Power consumption of cloud in data extraction from image, decoding, and processing
P_{popt}	Power consumption in processing data to find optimal path

$$\begin{aligned} P_{tot} =& P_{\text{Sensors}} + P_{rSND} + P_{deSND} + P_{pSND} \\ &+ P_{enSND} + P_{tSND} + P_{rSFO} + P_{deSFO} + P_{pSFO} \\ &+ P_{enSFO} + P_{tSFO} + P_{rC} + P_{pC} + P_{popt} \end{aligned} \tag{14}$$

Total delay and power consumption of the proposed framework will be compared with the existing frameworks in Sect. 5.

Table 3 Configurations used in simulation

Configuration (Config) No	Host storage (GB)	Cloud virtual machine (VM) CPU (GHz)	Cloud VM RAM (GB)	Fog CPU (GHz)	Fog RAM (GB)
1	1	3	4	3	4
2	1.5	3	4	3	4
3	2	3	4	3	4
4	2.5	3	4	3	4
5	3	3	4	3	4

5 Performance evaluation

To evaluate the performance of the proposed paradigm we have used iFogSim for implementation, MATLAB for theoretical analysis and steganography code implementation, and Google Cloud Platform for implementation of the code of the proposed optimal path finding algorithm.

5.1 Implementation using iFogSim

The sensor-fog framework for mission-critical applications has been simulated using iFogSim [35]. Eclipse IDE has been used for implementation, and with it the JProfiler has been integrated. The iFogSim has been used to create the sensor-fog topology (refer to Fig. 4), and the respective code has been written, complied, and executed. During execution of the topology and the code, with the Java Virtual Machine (JVM) the JProfiler has been attached for monitoring the CPU load and memory

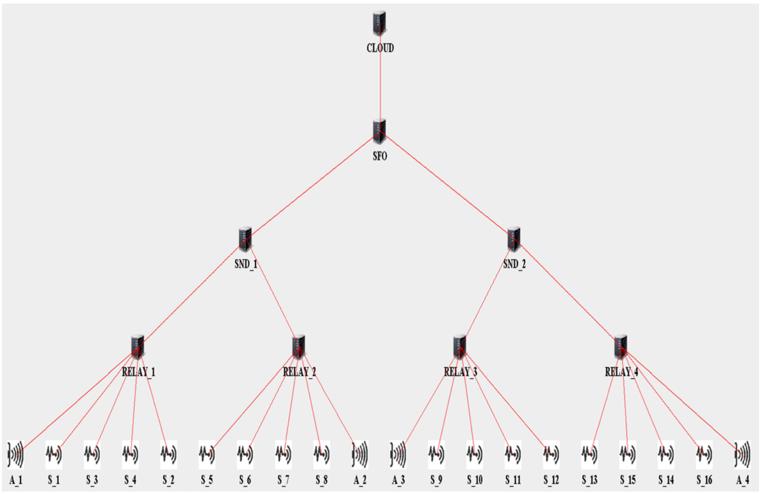


Fig. 4 Created Sensor-fog topology in iFogSim

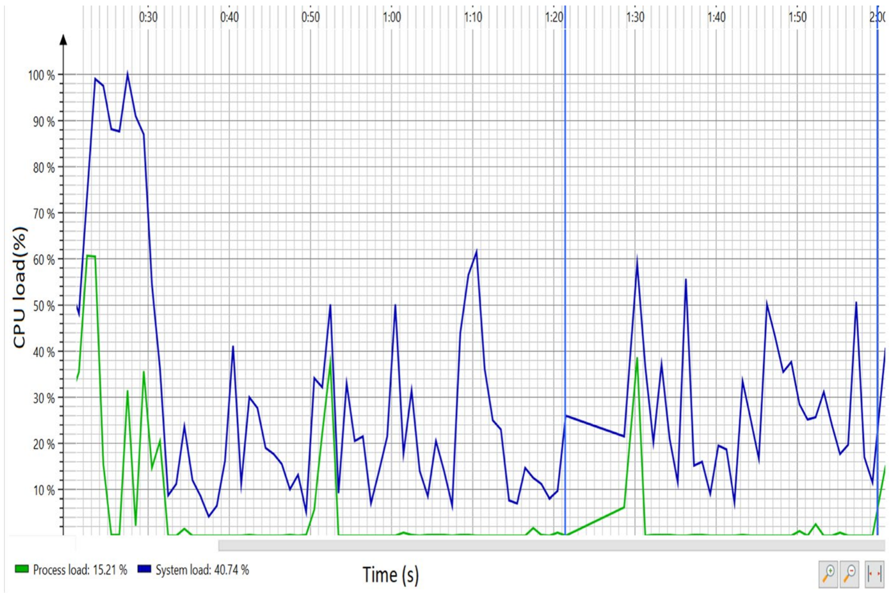


Fig. 5 CPU load during execution of the topology and code

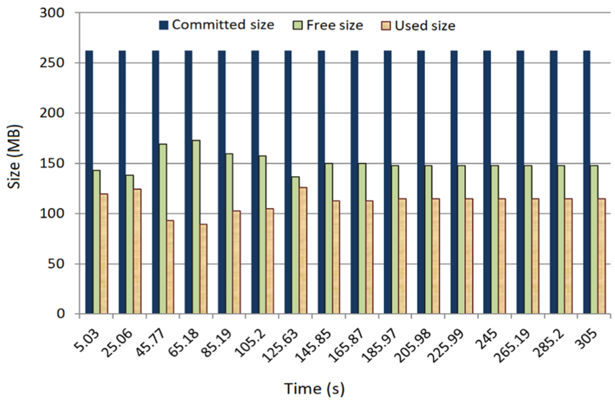


Fig. 6 Memory usage during execution of the topology and code

usage. As observed from Fig. 4, sixteen sensors, four actuators, four relay nodes, two SNDs, one SFO, and cloud are used in the created topology. Four sensors and one actuator are mapped into one relay node. Here, we have four subsets each containing four sensors and one actuator. Each subset is mapped into a relay node. Two relay nodes are mapped into one SND. Two SNDs are mapped into one SFO. The SFO is mapped into the cloud.

Figure 5 presents the CPU load. Here, the green and blue lines, respectively, represent the process load and system load. Figure 6 presents the memory usage, where the committed, used, and free memory (heap) size are presented. The code corresponding to the created topology has been executed to monitor the execution delay. We have simulated the sensor-cloud framework (without fog computing) for mission-critical applications to compare with the sensor-fog framework. The code corresponding to the sensor-cloud framework has been executed, and the execution delay has been monitored. Five different cases are considered (refer to Table 3). Figure 7 presents the execution delays (measured in second (s)) in fog-based (sensor-fog) and cloud-only (sensor-cloud) frameworks for mission-critical applications. This is observed that the fog-based framework (sensor-fog framework) has reduced the execution delay by $\sim (18-40)\%$ than the cloud-only framework (sensor-cloud framework).

5.2 Modified LSB-based Image Steganography for Data hiding

We have used MATLAB R2023a for implementation of the modified LSB-based image steganography. The sensor data in binary form is encoded and embedded into an image using modified LSB-based image steganography. We have used an image (refer to Fig. 8) as input image and then applied modified LSB-based image steganography. The output image is displayed in Fig. 9. As observed from Figs. 8 and 9, there is no visible change. After receiving this image, the SFO extracts the encoded data embedded into the image and decodes the data. The time consumption in encoding (encoding and embedding into image) and decoding (extracting from image and decoding) while using modified LSB-based image steganography are displayed in Fig. 10 with respect to the number of bits in the sensor data. The used sensor data stream (in bits) are:

00110001001100000011000101111100
0011000100110000001100010011000101111100
001100010011000000110001001100010011000101111100

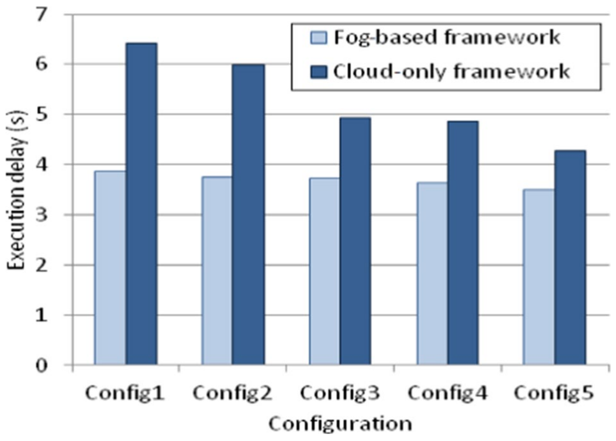


Fig. 7 Comparison of execution delay: Fog-based and cloud-only frameworks for mission-critical applications

Fig. 8 Input image: Original image used in Encoding

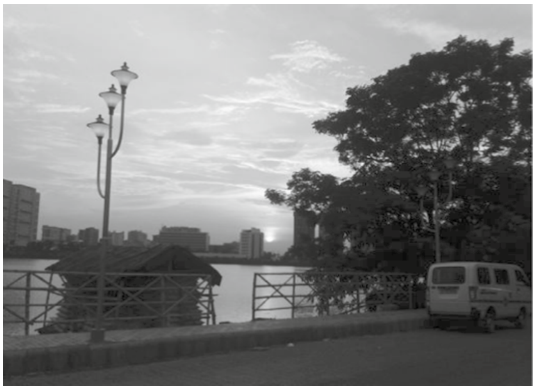


Fig. 9 Output image: Encoded image after using modified LSB-based image steganography

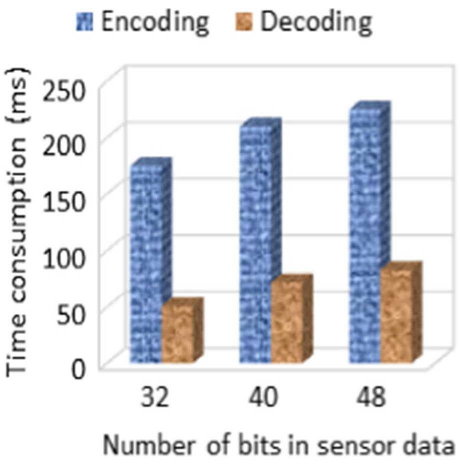


Fig. 10 Time consumption in encoding and decoding while using modified LSB-based image steganography

This is observed that the time consumption in encoding and decoding varies from 150–250 millisecond (ms) and 50–100 ms, respectively. The delay and power consumption of the proposed framework considering the data transmission, encoding,

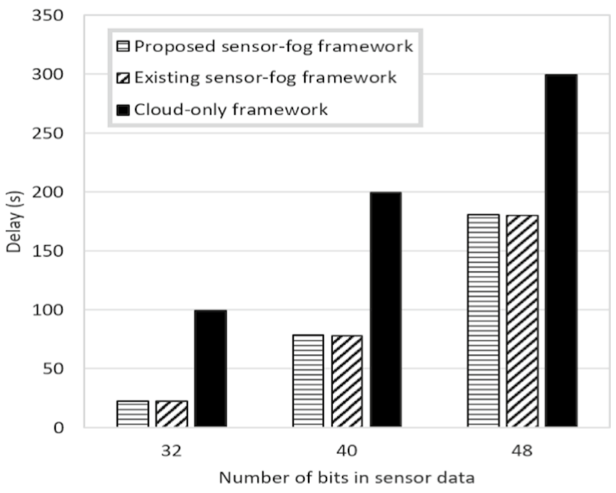


Fig. 11 Comparison of total delay: Proposed and existing frameworks for mission-critical applications

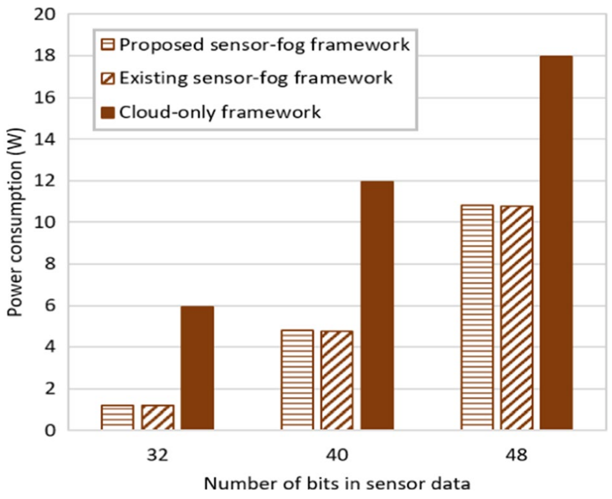


Fig. 12 Comparison of power consumption: Proposed and existing frameworks for mission-critical applications

decoding, and processing are presented in Figs. 11 and 12, respectively. The results are compared with the cloud-only framework (conventional sensor-cloud framework, where fog computing is not used) for mission-critical applications. This is observed that the proposed framework reduces the delay and power consumption by $\sim(40-80)\%$ and $\sim(40-80)\%$ than the cloud-only framework (sensor-cloud framework). The proposed framework is compared with the sensor-fog framework for mission-critical applications, where no encoding has taken place [1]. It is observed that the proposed framework has approximately same power consumption and delay though it uses network coding and steganography. This is because the use

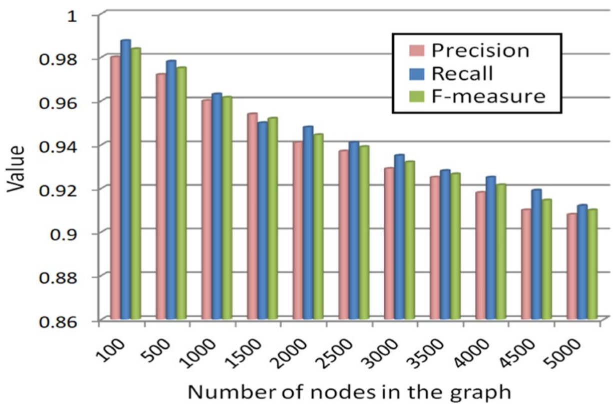
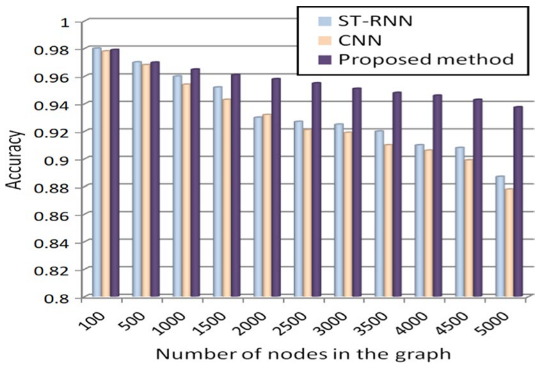


Fig. 13 Precision, Recall, and F-measure in path extraction using proposed method

Fig. 14 Accuracy values for path extraction using proposed and existing methods



of steganography and network coding result in a small amount of delay and power consumption.

5.3 Optimal path finding

The performance of optimal path finding module is presented in this section. The test configurations used are: Google Cloud VM (4 $vCPU$, 15$GB\ memory$), TensorFlow platform. A region of 10.8km^2 with 16×10^3 nodes is considered for evaluating the framework over the underlying road-network. For fair comparisons, we have simulated several exigency scenarios. Our framework finds out the path when these scenarios occur. Figure 13 presents the precision, recall, and F-measures of the path finding module with a high precision (≥ 0.90) and recall (≥ 0.912) values with 100 and 5000 nodes in the road-graph, respectively. It is observed that in exigency situations Mobi-Sense is capable of extracting optimal paths to reach the destination avoiding the blockage or affected regions.

Figure 14 illustrates accuracy of the path finding module with varied number of nodes ranging from 100 to 5000. We have compared the accuracy result with two well-known deep-learning baselines, namely, *ST-RNN* [33] and *CNN* [31]. The accuracy is computed based on the percentage of correct path extracted to reach the destination in minimal time. It shows high accuracy in the range of $0.979 - 0.9375$. In the same set-up, $ST - RNN$ and CNN provide $0.98 - 0.887$ and $0.978 - 0.878$ range of accuracy, respectively. Since Mobi-Sense models the study-region in an opportunistic network, and deploys the path finding algorithm using deep learning architecture to capture the impact of overall environment (victim region, blockage of roads etc.), it can efficiently extract optimal path and outperforms the existing baselines.

5.4 Summary of the inferences from results

As observed from the simulation and experimental results the proposed framework achieves:

- Reduction in delay than the cloud-only framework (sensor-cloud framework without fog computing) as observed from the iFogSim results.
- Data is hidden in another medium to provide data confidentiality during transmission, and no visible change in the transmitted media in which the data is hidden. This is observed from the results obtained using MATLAB.
- Reduction in delay and power consumption than existing cloud-only framework (sensor-cloud framework without fog computing) as observed from the results obtained using MATLAB.
- The proposed optimal path selection method has high accuracy and precision as observed from the experimental results.

Thus, we can state that the proposed framework Mobi-Sense provides data confidentiality during transmission, achieves lower delay and lower power consumption, and finds path to the victim region with high accuracy and precision in emergency situation.

6 Conclusions and future work

In this paper, we have proposed a mobility-aware sensor-fog paradigm for mission-critical applications using network coding and modified LSB-based image steganography, referred to as Mobi-Sense. The data collected by the sensor nodes is processed inside the fog device to predict the current status of a region. For confidentiality during data transmission to the fog node and cloud, modified LSB-based image steganography is used. In the modified LSB-based image steganography, we encode the sensor data and then embed into the image using LSB-based steganography. If any disaster is detected, an optimal path to the affected region is found using opportunistic network and deep learning. The sensor-fog framework has been implemented in iFogSim, and the results illustrate that Mobi-Sense has reduced the delay by $\sim (18 - 40)\%$ than the cloud-only framework (sensor-cloud framework) for mission-critical applications. The theoretical analysis illustrates that Mobi-Sense has $\sim (40 - 80)\%$ lower delay and power consumption than the cloud-only framework (sensor-cloud framework). The experimental analysis shows that the proposed optimal path finding method has achieved precision and accuracy above 90%. The experimental results also demonstrate that the proposed method outperforms the existing path finding modules with respect to the accuracy. As part of the future work we plan to extend our framework for secure monitoring of highly sensitive unmanned regions.

Author contributions AM: Conceptualization, Formal Analysis, Methodology, Writing original draft. SG: Conceptualization, Methodology, Data curation, Writing original draft. SKG: Resources, Supervision, Writing review & editing. RB: Investigation, Supervision, Writing review & editing.

Funding The authors did not receive support from any organization for the submitted work.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval Not applicable.

Availability of data and materials The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

References

1. Mishra M, Roy SK, Mukherjee A, De D, Ghosh SK, Buyya R (2019) An energy-aware multi-sensor geo-fog paradigm for mission critical applications. J Amb Intell Human Comput. 1–19
2. Misra S, Singh A, Chatterjee S, Obaidat MS (2014) Mils-cloud: a sensor-cloud-based architecture for the integration of military tri-services operations and decision making. IEEE Syst J 10(2):628–636
3. Ghosh S, Mukherjee A, Ghosh SK, Buyya R (2022) Stoppage: Spatio-temporal data driven cloud-fog-edge computing framework for pandemic monitoring and management. Softw Pract Exper 52(12):2700–2726
4. Nage T, Yu FR, St-Hilaire M (2010) Adaptive control of packet overhead in xor network coding. In: 2010 IEEE International Conference on Communications 1–5. IEEE
5. Chen J, Lee VCS, Liu K, Ali MGGN, Chan E (2013) Efficient processing of requests with network coding in on-demand data broadcast environments. Inf Sci 232:27–43
6. Provos N, Honeyman P (2003) Hide and seek: an introduction to steganography. IEEE Secur Privacy 1(3):32–44
7. Sarmah DK, Kulkarni AJ (2018) Jpeg based steganography methods using cohort intelligence with cognitive computing and modified multi random start local search optimization algorithms. Inf Sci 430:378–396
8. Yuan HD (2014) Secret sharing with multi-cover adaptive steganography. Inf Sci 254:197–212
9. Mukherjee A, De D, Ghosh SK (2020) Fogioht: a weighted majority game theory based energy-efficient delay-sensitive fog network for internet of health things. Inter Things 100181
10. Ghosh S, Mukherjee A, Ghosh SK, Buyya R (2019) Mobi-iost: mobility-aware cloud-fog-edge-iot collaborative framework for time-critical applications. IEEE Trans Netw Sci Eng
11. Das J, Ghosh S, Mukherjee A, Ghosh SK, Buyya R (2022) Rescue: enabling green healthcare services using integrated iot-edge-fog-cloud computing environments. Softw Pract Exp 52(7):1615–1642

12. Gavrilović N, Mishra A (2021) Software architecture of the internet of things (iot) for smart city, healthcare and agriculture: analysis and improvement directions. J Amb Intell Humaniz Comput 12(1):1315–1336
13. Almurisi N, Tadisetty S (2022) Cloud-based virtualization environment for iot-based wsn: solutions, approaches and challenges. J Amb Intell Humaniz Comput 1–23
14. Burmaoglu S, Saritas O, Yalcin H (2019) Defense 4.0: internet of things in military. In *Emerg Technol Econ Develop* 303–320. Springer
15. Mukherjee A, Ghosh S, De D, Ghosh SK (2022) Mcg: Mobility-aware computation offloading in edge using weighted majority game. IEEE Trans Netw Sci Eng
16. Memari P, Mohammadi SS, Jolai F, Tavakkoli-Moghaddam R (2022) A latency-aware task scheduling algorithm for allocating virtual machines in a cost-effective and time-sensitive fog-cloud architecture. J Supercomput 78(1):93–122
17. Ahmad M, Amin MB, Hussain S, Kang BH, Cheong T, Lee S (2016) Health fog: a novel framework for health and wellness applications. J Supercomput 72(10):3677–3695
18. Mukherjee A, Ghosh S, Behere A, Ghosh SK, Buyya R (2020) Internet of health things (ioht) for personalized health care using integrated edge-fog-cloud network. J Amb Intell Humaniz Comput
19. Sulyman AI, Henggeler C (2022) Physical layer security for military iot links using mimo-beamforming at 60 ghz. Information 13(2):100
20. Campioni L, Poltronieri F, Stefanelli C, Suri N, Tortonesi M, Wrona K (2023) Enabling civil-military collaboration for disaster relief operations in smart city environments. Fut Generat Comput Syst 139:181–195
21. Bichi BY, Islam SU, Kademi AM, Ahmad I (2022) An energy-aware application module for the fog-based internet of military things. Discov Intern Things 2(1):4
22. Limkar SV, Jha RK (2019) A novel method for parallel indexing of real time geospatial big data generated by iot devices. Fut Generat Comput Syst 97:433–452
23. MacEachren AM, Robinson A, Hopper S, Gardner S, Murray R, Gahegan M, Hetzler E (2005) Visualizing geospatial information uncertainty: What we know and what we need to know. Cartogr Geogr Inf Sci 32(3):139–160
24. Yang C, Yu M, Hu F, Jiang Y, Li Y (2017) Utilizing cloud computing to address big geospatial data challenges. Comput Environ Urban Syst 61:120–128
25. Dulík M, Junior MD (2016) Security in military cloud computing applications. Sci Milit J. 11(1):26
26. Chakravarthy M Hemanth, Kannan E, Belinda MJ Carmel Mary (2019) A hybrid routing protocol towards secure and smart military applications in cloud environments. In *Novel Practices and Trends in Grid and Cloud Computing*, pages 234–248. IGI Global
27. Thangadurai K, Devi G Sudha (2014) An analysis of lsb based image steganography techniques. In *2014 International Conference on Computer Communication and Informatics*, pages 1–4. IEEE
28. Hemanth H, Hrutish Ram VS, Raghavendran S Guru, Subhashini N (2022) Modified lsb algorithm using xor for audio steganography. In: Sustainable Advanced Computing: Select Proceedings of ICSAC 2021, pages 369–379. Springer
29. Rhoads GB (July 24 2001) Audio or video steganography. US Patent 6,266,430
30. Huang CM, Lan KC, Tsai CZ (2008) A survey of opportunistic networks. In :22nd International Conference on Advanced Information Networking and Applications-Workshops (aina workshops 2008), pages 1672–1677. IEEE
31. Karatzoglou A, Schnell N, Beigl M (2018) A convolutional neural network approach for modeling semantic trajectories and predicting future locations. In: International Conference on Artificial Neural Networks, 61–72. Springer
32. Liu W, Wang Z, Liu X, Zeng N, Liu YY, Alsaadi Fuad E (2017) A survey of deep neural network architectures and their applications. Neurocomputing 234:11–26
33. Liu Q, Wu S, Wang L, Tan T (2016) Predicting the next location: a recurrent model with spatial and temporal contexts. In: Thirtieth AAAI conference on artificial intelligence
34. Ghosh S, Mukherjee A (2022) Strove: Spatial data infrastructure enabled cloud–fog–edge computing framework for combating covid-19 pandemic. Innov Syst Softw Eng 1–17
35. Gupta H, Vahid DA, Ghosh SK, Buyya R (2017) ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. Softw Pract Exp 47(9):1275–1296

Authors and Affiliations

Anwesha Mukherjee[1] · Shreya Ghosh[2] · Soumya K. Ghosh[3] · Rajkumar Buyya[4]

Anwesha Mukherjee
anweshamukherjee2011@gmail.com

Soumya K. Ghosh
skg@cse.iitkgp.ac.in

Rajkumar Buyya
rbuyya@unimelb.edu.au

[1] Department of Computer Science, Mahishadal Raj College, Mahishadal, West Bengal, India

[2] College of Information Sciences and Technology, The Pennsylvania State University, State College, USA

[3] Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, Kharagpur, India

[4] Cloud Computing and Distributed Systems (CLOUDS) Lab, School of Computing and Information Systems, The University of Melbourne, Melbourne, Australia