 doi: 10.1109/WCNC.2019.8885626

OPTIVE: Optimal Configuration of Virtual Sensor in Mobile Sensor-Cloud

Arijit Roy[†], *Student Member, IEEE*, Sudip Misra[‡], *Senior Member, IEEE*, and Lakshya[§]
[†]Advanced Technology Development Centre, [‡§]Department of Computer Science and Engineering,
[†‡§]Indian Institute of Technology Kharagpur, India,
{[†]arijitroy, [‡]sudipm}@iitkgp.ac.in, [§]lakshya0459.iitkgp@gmail.com

***Abstract*—In this paper, we propose a scheme, *OPTIVE*, for obtaining the optimal configuration of a virtual sensor in the mobile sensor-cloud (MSC) architecture. The proposed scheme is capable of selecting the physical sensor nodes to form a virtual sensor (VS), based on the sensing area coverage for an application region. We use *Markov Decision Process* (*MDP*) to select the optimal mobile sensor nodes among the available ones, for configuring the VS in the application area. The MSC architecture is a new paradigm in which physical sensor nodes attain mobility by the virtue of mobile devices such as, laptops, cell phones, and vehicles. In MSC, a mobile device may move or exit from the application region at any time instant. Consequently, sensing hole arises in the application area, resulting in undesirable interruption in the end-user services. As multiple sensor nodes may be present in the application region, it is not suitable to allocate any available sensor node, randomly, to re-configure the VS for covering the sensing hole. In such a situation, OPTIVE selects the optimal physical sensor node to allocate in the VS for ensuring uninterrupted services to the end-users. Simulation results show that OPTIVE is capable of providing at least $80 - 90\%$ coverage in the application area. Additionally, in the presence of 2 to 7 sensor nodes, the number of iterations in MDP change by 19.56%.**

***Keywords*—Mobile Sensor-Cloud, Virtual Sensor, Optimal Sensing Coverage, Markov Decision Process, Percentage of coverage.**

I. Introduction

The sensor-cloud architecture is based on the concept of virtualization of physical sensor nodes [1], which allows multiple end-users to receive the services from a single sensor node simultaneously. The sensor-cloud architecture depletes the traditional single user-centric view of Wireless Sensor Networks (WSNs). Typically, in sensor-cloud, a set of static sensor nodes combine to form a VS for provisioning *Sensor-as-a-Service* (*Se-aaS*) to the end-users. On the other hand, the Mobile Sensor-Cloud (MSC) explores a new dimension of cloud computing where mobile sensor nodes are virtualized to serve end-user applications. In MSC, the sensor nodes are attached to certain mobile devices, such as laptop, cell phone, and vehicles. Thus, the physical sensor nodes attain mobility due to the movement of the devices, to which these are attached. The MSC architecture comprises of four actors– sensor owner, device owner, end-user, and sensor-cloud service provider (SCSP). The sensor owners procure and deploy the sensor nodes on the mobile devices, which are owned by the respective device owners. A SCSP manages the entire MSC architecture using certain algorithms or by manual intervention and provides Se-aaS to multiple end-users. On the other hand, the end-users enjoy the requested services through a Web portal on payment-basis. Further, SCSP pays rent to the device owners and the sensor owners using the payment earned from the end-users. Additionally, SCSP makes his/her profit and earns the maintenance cost from the payment of the end-users. Typically, the payment mechanism is maintained with the help of some pre-defined pricing schemes. An architecture of MSC is depicted in the Fig. 1.

In an MSC, the sensor nodes become mobile with the movement of the devices to which they are attached. In order to provision Se-aaS for an end-user application, mobile sensor nodes are required to be allocated to a VS. Also, the coverage in the application area depends on the location of mobile devices, which are equipped with the sensor nodes. Therefore, the movement of a device, which serves a certain application area, from one location to another gives rise to sensing holes in the application area. This work primarily focuses on the dynamic allocation of physical sensor node, optimally, for configuring a VS to cover the sensing holes in an application region.

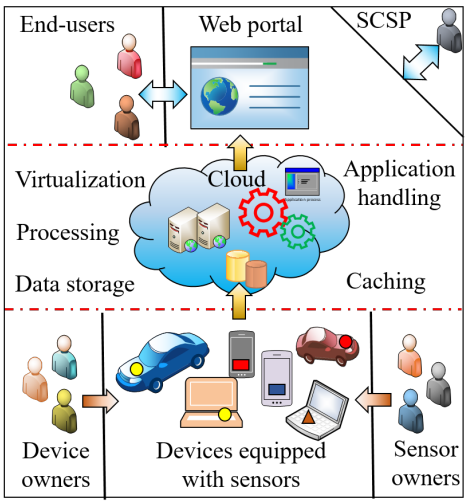


Fig. 1: Architecture of MSC

A. Motivation

In an MSC architecture, a VS comprises of multiple physical sensor nodes, which are attached to different mobile devices. Thus, due to the dynamic movement of these mobile devices, the application region may become uncovered. Consequently, sensing hole arises in the application region and

the end-users service interrupts. To resume the uninterrupted end-user services, it is essential to allocate other sensor nodes in the VS. In such a situation, multiple mobile devices, equipped with different sensor nodes, may be present in the application region. Thus, these sensor nodes can be used for reconfiguring the VS to cover the sensing hole. However, the selection of any random sensor node among the available ones for reconfiguring the VS is not pertinent. The traditional sensor-cloud architecture comprises of static sensor nodes, and therefore, the existing schemes [2] and [3] for reconfiguring the VS is not suitable for MSC. Thus, we propose a scheme, named OPTIVE, which is capable of reconfiguring the VS by selecting the best possible sensor node, among the available ones in the MSC architecture.

B. Contribution

In this work, we consider the case of distortion in the composition of a VS in MSC, due to the dynamic movement of the mobile devices from an application area. This situation gives rise to sensing holes in the application area. To address this problem, we propose a scheme, OPTIVE, which optimally selects a physical sensor node among the available ones to re-configure the VS. In Fig. 2, we depict a few possible cases of the movement of mobile devices from the application area. Further, we use MDP in the solution for allocating the best possible available sensor nodes in a VS to remove the sensing holes from the application area. Finally, to evaluate the proposed scheme, OPTIVE, we performed rigorous simulations and discussed the results.

II. Background

In this section, we discuss different works on sensor-cloud architecture. Yuriama and Kushida. [4] introduced the concept of sensor-cloud with a basic architecture. Further, Misra *et al.* [1] proposed the theoretical model of senor-cloud in which the authors elaborately discussed about the different actors associated with it. Typically, in the sensor-cloud architecture, end-users pay the price based on the usage of the services. The payment from the end-users is needed to be distributed in a fair way among the different actors, such as the SCSP and the sensor owner. Thus, in order to handle the pricing issues of sensor-cloud, Chatterjee *et al.* [5] proposed an optimal pricing scheme for sensor-cloud architecture. Additionally, a trust enforcing pricing scheme, *DETER*, for sensor-cloud is proposed by Chakraborty *et al.* [6]. DETER enforces trust among the sensor owners, while maintaining the quality of Se-aaS. The authors used *Single-Leader-Multiple-Follower Stackelberg Game* for deciding the price to be paid to the sensor owners. Bose *et al.* [7] proposed the concept of using virtual sensors for environment monitoring. Further, Madria *et al.* [8] explored the real implementation of sensor-cloud. An adaptive data caching scheme for sensor-cloud is proposed by Chatterjee *et al.* [9], which is capable of minimizing the service delay to the end-users. Additionally, Roy *et al.* [10] came up with a unique scheme of data caching for sensor-cloud, which is able to cache the data of the destroyed virtual machines. In another work, Chatterjee and Misra [2] proposed a scheme for composing virtual sensors dynamically in order to provide efficient services to the end users. In this work [2], the authors consider the presence of non-overlapping sensor deployment region. Further, considering overlapping sensor node deployment region, Roy *et al.* [3] designed a scheme for forming VS with the physical sensor node.

Synthesis: In the existing literature, the authors studied different problems in sensor-cloud architecture and provided corresponding suitable solutions for addressing these problems. These works consider the presence of static sensor nodes in the sensor-cloud architecture. Moreover, the authors in [2] and [3] proposed their respective schemes for forming a VS optimally considering only static sensor nodes in the sensor-cloud architecture. In this work, we consider that the VS is composed of mobile sensor nodes, which are typically attached to the mobile devices. These mobile devices may exit the application area at any time instant, which result in distortion in the composition of the VS. Moreover, the aforementioned problem itself is different from the problems of configuring the VS, discussed in the existing literature [2] and [3]. Consequently, the existing solutions are not suitable for addressing the problem of formation of VS in MSC, identified in this work.

III. Problem Description

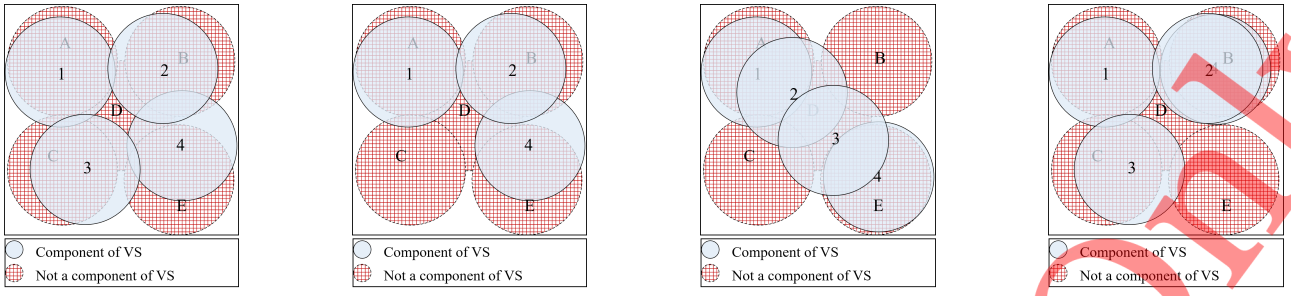
An MSC consists of mobile devices, which are equipped with different physical sensor nodes. Thus, physical sensor nodes attain mobility by the motion of mobile devices. However, the motion of the devices is controlled by the device owner.

A. Problem Scenario

We consider a mobile sensor-cloud platform, which consists of heterogeneous physical sensor nodes. In order to provide sensing coverage to a particular application area, multiple physical sensor nodes are required. Consequently, multiple devices are essential to be present inside the application area. In MSC platform, multiple physical sensor nodes combine to form a VS. However, due to the movement of mobile devices, physical sensor nodes may exit from the VS at any time instant. Thus, in such a scenario, it is essential to re-configure the VS, in order to provide an uninterrupted service to the end users.

***Definition* 1.** *Threshold Percentage of Sensing Coverage (P_S) is defined as the minimum amount of sensing coverage, expressed in percentage, in an application area, provided by a SCSP to an end-user.*

Fig. 2 depicts the possible cases for which re-allocation of physical sensor node in a VS is required. In each of the cases, there are four nodes-$1, 2, 3$, and 4, which are already covering the application region and are the part of a VS. However, other sensor nodes, A,B,C,D, and E are present in the application area, which are not part of the VS. The possible cases those arise due to the mobility of physical sensor nodes are as follows:
Case 1: Physical sensor nodes, $1, 2, 3, 4$, cover the application region more than a threshold value, P_S, defined by SCSP.
Case 2: Initially, the physical sensor nodes cover the application region more than the threshold value. However, due to the mobility of the device, node 3 exits the application region. Consequently, sensing hole arises in the application region and percentage of coverage drops below P_S, defined by the SCSP,



(a) Case 1 (b) Case 2 (c) Case 3 (d) Case 4

Fig. 2: Few possible cases, where re-allocation of sensor node is required

as mentioned in Definition 1.

Case 3: Due to mobility, the sensor nodes, $1, 2, 3, 4$, are aligned in such a fashion that they are unable to cover the application area, equal to or above P_S value, resulting in sensing hole.

Case 4: In this case, node 4 move in such a fashion that the sensing range of node 2 and 4 overlaps. Consequently, sensing hole arises in the application region and percentage of coverage drops below P_S.

We consider Case 1 to be a normal condition, where the percentage of coverage is above or equal to P_S. For Case 2-4, we require to re-allocate the physical sensor nodes to the VS, in order to cover the application area. However, there may exist other cases also, for which re-allocation of the physical sensor nodes are required.

B. Problem Formulation

Let the set of physical sensor nodes present in the system be denoted as $\mathbb{S} = \{s_1, s_2, s_3, \cdots, s_n\}$. The set of mobile devices present in the MSC platform are represented by $\mathcal{D}$, where each $d_i \in \mathcal{D}$ be any device, which is equipped with one or multiple physical sensor nodes. An application requested by an end-user is depicted as $A = \langle A_{id}, A_{loc}, A_{type} \rangle$, where $A_{id}, A_{loc}, A_{type}$ are application id, application location, and application type respectively [1]. A physical sensor node, s, is defined as a tuple, $s = \langle s_{id}, s_{loc}, s_{type}, s_{device}, s_{velocity}, s_{st}, s_{energy}, s_r \rangle$, where s_{id} and s_{type} denote the id and type of the physical sensor node, s. Additionally, s_{device}, s_{loc}, and $s_{velocity}$ denotes the device id to which the sensor node is attached, current location of the sensor node in the application area, and the current velocity of the device, respectively. s_{energy} and s_r represents the current residual energy and the sensing range of the sensor node, s. The state of a sensor node is denoted as s_{st}. The possible state of the sensor node is either active or inactive. If a physical sensor node serves at least one application area, then we consider the state of the sensor node as active. Thus, the state of a sensor node, s_{st}, is represented as:

$$s_{st} = \begin{cases} 1, & \text{if node is active} \\ 0, & \text{if node is inactive} \end{cases} \quad (1)$$

The set of applications is denoted by $\mathbb{A}$. We define $\mathbb{S}'(t)$ as a set of physical sensor nodes, which are suitable to be used for serving the application, A, given as:

$$\mathbb{S}'(t) = \{s | s.s_{type} = A.A_{type} \text{ and s is located inside } A.A_{loc}, \forall s\}, \quad (2)$$

At any time instant t, $VS(t)$ represents a virtual sensor, such that $VS(t) \subseteq \mathbb{S}'(t)$.

***Definition* 2.** *Percentage of coverage ($\mathcal{PC}$) is the total percentage of application area covered, by the sensing area of a set of physical sensor nodes, S, such that $S \subset \mathbb{S}$ and $s_i \in S$ $\forall i$.*

Let $s_i \in S$ denotes any sensor node. The total number of nodes present in S is represented by k. The sensing area of any physical sensor node, s_i, is denoted by $Area(s_i) = \pi {r_i}^2$, where r_i is the sensing radius of s_i. Let γ_t denotes the set of locations at time instant t, such that $\gamma_t = \{l_1, l_2, l_3, \cdots, l_k\}$ and l_i is the location of s_i. The *percentage of coverage*, $\mathcal{PC}(S, \gamma_t)$, is mathematically represented as:

$$\mathcal{PC}(S, \gamma_t) = \frac{Area(S, \gamma_t)}{Area_{app}} \times 100\% \quad (3)$$

and

$$Area(S, \gamma_t) = \bigcup_{i=1}^{k} Area(s_i) \quad (4)$$

where $Area_{app}$ and $Area(s_i)$ denote the total application area and the area covered by any physical sensor node, $s_i \in \mathbb{S}$, respectively. Let at time instant, $t = 0$, the computation for the reconfiguration of a VS is started. At t = $-\epsilon$, $\mathcal{PC}(S, \gamma_\epsilon) > P_S$, and at t = 0, $\mathcal{PC}(S, \gamma_0) < P_S$, where, $\epsilon \to 0$, γ_0 is the corresponding set of locations for S at $t = 0$, and γ_ϵ is the corresponding set of locations for S at $t = -\epsilon$.

Therefore, we re-configure the VS, at time instant τ, in such a way that the percentage coverage of VS is greater than P_S. Mathematically, at $t = \tau$, we have, $\mathcal{PC}(S, \gamma_\tau) > P_S$, where γ_τ is the corresponding location set for S.

IV. Solution Approach

Let P denotes the probability of the achieved location by a sensor node after time, τ.

$$P(s, \theta, \tau, r) = \text{real value between (0,1)} \quad (5)$$

where θ is the angle of deviation of the location of mobile device with respect to the current velocity of sensor node, with range, $(0, \pi) \cup (-\pi, 0)$.

$$\theta = \begin{cases} > 0, & \text{if } \theta \text{ is clockwise} \\ < 0, & \text{if } \theta \text{ is anticlockwise} \end{cases} \quad (6)$$

where r is the distance between the current location of the mobile device and the location to which the device reaches after time, τ.

A. Approximation of device motion

The possible values of θ and r, in Equation (5), are infinite for a particular value of s and τ. Therefore, the process of computing the probability, $P(s, \theta, \tau, r)$, of a sensor node, s,

being at any given position (r, θ) after given time, τ, is very challenging. We can approximate (r, θ), by a set of expected locations[1] of the device owner. We compute the expected locations of a sensor node, in terms of expected trajectories and possible distance, d, traveled by the sensor node in time τ. The locations of the sensor node after traveling the distance d in time τ on the expected trajectories are the expected locations. We use linear regression [11] for calculating the value of d as $d = a + b \cdot t$, where a and b are computed using the observed motion of the device, such that $d - (a + b \cdot t)$ is minimum for each pair (d_i, t_i). We use *least square error* to penalize the learning, using Equations (7) and (8), as follows:

$$\delta = \sum_{i=1}^{i=n} (d_i - (a + b \cdot t_i))^2 \tag{7}$$

$$a, b : min_{a,b}(\delta) \tag{8}$$

where n is the total number of observations recorded for the device. In order to calculate minimum value of δ, we calculate the partial derivatives of Equation (7) w.r.t. a and b, and equate them to zero. Thus, we get Equations (9) and (10) respectively.

$$\frac{\partial \delta}{\partial b} = -2 \sum_{i=1}^{i=n} t_i \cdot (d_i - (a + b \cdot t_i)) = 0 \tag{9}$$

$$\frac{\partial \delta}{\partial a} = -2 \sum_{i=1}^{i=n} (d_i - (a + b \cdot t_i)) = 0 \tag{10}$$

Therefore,

$$\begin{bmatrix} b \\ a \end{bmatrix} = \begin{bmatrix} \sum_i d_i \\ \sum_i (t_i \cdot d_i) \end{bmatrix} \begin{bmatrix} n & \sum_i (t_i \cdot d_i) \\ \sum_i t_i & \sum_i t_i^2 \end{bmatrix}^{-1} \tag{11}$$

$P'(s, \tau, l)$ denotes the probability of a sensor node, s, for being at the expected location, l, after time, τ. Let $\mathbb{L}_i = \{l_1, l_2, l_3, \cdots, l_n\}$ denote the set of all expected locations of sensor node, s_i. Let $\mathbb{L}$ be the set of expected location sets of corresponding sensor nodes. Thus, we have,

$$\mathbb{L}(\{s_1, s_2, s_3, ...s_z\}) = \{\{l_1, l_2, l_3.....l_z\}, |l_i \in \mathbb{L}_i\} \tag{12}$$

B. Markov Decision Process-based Optimal Reconfiguration

In MSC, the devices in which the sensor nodes are attached are mobile and their motion is unpredictable. In such a scenario, configuring the VS with mobile sensor nodes is a stochastic process. Therefore, we use the *MDP* [12], in order to reconfigure the VS with available physical sensor nodes in the application region. Typically, MDP has four major components–*state*, *action*, *state transition probability*, and *reward*, which are defined as follows:

State: In this work, we represent state, Ψ, as a 3-tuple, $\langle \alpha, M, \beta \rangle$, where $\alpha = \mathcal{PC}(M, \beta)$, $M \subseteq \mathbb{S}'(t = 0)$ and $\beta \in \mathbb{L}(M)$. Subsequently, we define state space=$\{\Psi, \forall M$ and $\forall \beta\}$.

[1]Example: A sensor node is attached to cell phone of employee of a company. The motion of the cell phone will depend on the motion of the employee (as the employee keeps his/her phone with himself/herself). The possible destinations of the employee are his desk, his friend's desk, to a toilet, to a boss office etc. Moreover, there are some finite paths that he will take (through cabins, stairs, and lift), to reach a destination. A statistical analysis can be done to compute the probability of the employee going to a particular destination through the particular path.

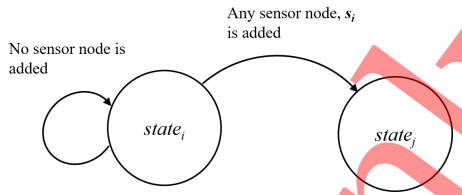


Fig. 3: Possible state transition based on action

***Proposition* 1.** *The size of state space is* $\prod_{i=1}^{|\mathbb{S}'(t=0)|}(q_i + 1)$, *where* q_i=$|\mathbb{L}_i|$

Justification: Consider β of any state for a particular sensor node, s_i, there are $(q_i + 1)$ independent choices with respect to β as:

(i) $s_i \in M$ and it exists in one of the expected locations in $\mathbb{L}_i$. Thus, it generates q_i choices.
(ii) $s_i \notin M$

Each sensor node, s_i has $(q_i + 1)$ independent choices, and therefore, the size of state space is $\prod_{i=1}^{|\mathbb{S}'(t=0)|}(q_i + 1)$
Action: An action is one of the following:

- Inclusion of a sensor node , $s_i \in (\mathbb{S} - M_i)$, with expected location, l_i, which results in change of state from $State_i$ to $State_j$, such that $(M_j - M_i) = s_i$ and $(\beta_j - \beta_i) = l_i$.
- Inclusion of a sensor node, $s_i \in M_i$, which results in no change of state.

Mathematically:

$$\mathcal{A}(State_i) = \begin{cases} State_i, & \text{if included sensor node } s_i \in M \\ State_j, & \text{such that } M_j - M_i = s_i \end{cases} \tag{13}$$

where $s_i \in (\mathbb{S} - M)$.

Let us consider that there are two states, $State_i$ and $State_j$. The possible state transitions based on the possible actions are depicted in the Fig. 3.

Lemma 1. *If the Markov chain is considered to be a directed graph, then there exists a directed path between two states,* ψ_1 *and* ψ_n, *such that* $M_1 \subset M_n$ *and* $\beta_1 \subset \beta_n$.

Proof: We use mathematical induction for justifying this lemma. In *base* case, we consider two states, ψ_1 and ψ_2, such that $(M_2 - M_1) = s_1$ and $(\beta_2 - \beta_1) = l_1$, where $l_1 \in \mathbb{L}_1$, then there exists a path between ψ_1 and ψ_2, consisting of action, $\mathcal{A}$, on inclusion of a sensor node, s_1, with expected location, l_1. We assume that there exists a path between two states, ψ_1 and ψ_{n-1}, such that $(M_{(n-1)} - M_1) = \{s_1, s_2, s_3, \cdots, s_{n-2}\}$ and $(\beta_{n-1} - \beta_1) = \{l_1, l_2, l_3, \cdots, l_{n-2}\}$, where $l_k \in \mathbb{L}_k, \forall k \in [1, (n-2)]$.

If we consider state, ψ_n, such that $(M_n - M_1) = \{s_1, s_2, s_3, \cdots, s_{n-1}\}$ and $(\beta_n - \beta_1) = \{l_1, l_2, l_3, \cdots, l_{n-1}\}$, where $l_k \in \mathbb{L}_k, \forall k \in [1, (n-1)]$, then there exists a path between ψ_n and ψ_{n-1}, consisting of action, $\mathcal{A}$, of inclusion of a sensor node, s_{n-1}, with expected location, l_{n-1}. Moreover, there exists a path between states, ψ_1 and ψ_{n-1} as per the assumption. Therefore, we conclude that there exists a path between states, ψ_1 and ψ_n, given $M_1 \subset M_n$ and $\beta_1 \subset \beta_n$. ■

Reward: Reward is defined as

$$R : \Psi \rightarrow \mathbb{R} \tag{14}$$

such that

$$R(\Psi) = \begin{cases} \lambda, & \text{if } \alpha < P_S \\ \dfrac{\alpha}{|M|}, & \text{if } \alpha \geq P_S \end{cases} \tag{15}$$

where λ is a negative constant. Moreover, the magnitude of λ has no impact on the optimal policy. If $\alpha \geq P_S$, the reward must be positive and proportional to α, and inversely proportional to $|M|$.

State Transition Probability: The state transition probability associated with the action, $\mathcal{A}$, of inclusion of a sensor node, s_i, with expected location, l_i, is depicted in Equation (16).

$$\mathcal{A}(State_i) = \begin{cases} \mathbb{P}(\Psi_i|\Psi_i, \mathcal{A}) = 1, & s_i \in M \\ \mathbb{P}(\Psi_j|\Psi_i, \mathcal{A}) = P'(s_i, \tau, l_i), & s_i \in \mathbb{S} - M \end{cases} \tag{16}$$

C. Computation of Optimal Policy

For calculation of optimal policy in the proposed problem, we use value iterations over infinite horizon [13]. Thus, we compute policy, such that expected sum of all future rewards are maximum. Mathematically:

$$max\left\{\mathbf{E}\left[\sum_{t=0}^{t=\infty} \mu^t R_t\right]\right\} \tag{17}$$

where μ is the scaling factor to provide an upper bound over the expected sum of all future rewards, such that $0 < \mu < 1$. The value function with respect to policy, π, is given as:

$$V^{\pi}(\Psi) = \mathbf{E}^{\pi}\left[\sum_{t=0}^{t=\infty} \mu^t R_t\right] \tag{18}$$

The optimal policy corresponds to the optimal value function, $V^*(\Psi)$. In order to determine $V^*(\Psi)$, we use *value iteration method*. In this process, we start with $V_t(\Psi) = 0, \forall \Psi$. Subsequently, we calculate $V_{t+1}(\Psi) = 0$ using:

$$V_{t+1}(\Psi) \leftarrow R(\Psi) + \mu \cdot max_{\mathcal{A}}\left\{\sum_{\Psi'} \mathbb{P}(\Psi'|\Psi, \mathcal{A}) \cdot V_t(\Psi')\right\} \tag{19}$$

According to *Bellman's theorem*, the value function converges to $V^*(\Psi)$ in finite number of iterations. Thus, we get the optimal policy, π^* as follows:

$$\pi^* = argmax_{\alpha}\left\{\sum_{\S'} Pr(\S'|\S, \alpha) \cdot V^*(\S')\right\} \tag{20}$$

After achieving the optimal policy, we start the initial state as $\Psi_{initial} = \langle 0, \phi, \phi \rangle$ and follow the policy until we reach to a state from which further action results in no change in the state. Let the final state be $\Psi_{final} = \langle \alpha_{final}, M_{final}, \beta_{final} \rangle$, then the reconfigured VS is equal to M_{final}.

Theorem 1. *Re-configured VS always provides higher coverage than P_S, if there exist a state, ψ_k such that $\alpha_k \geq P_S$.*

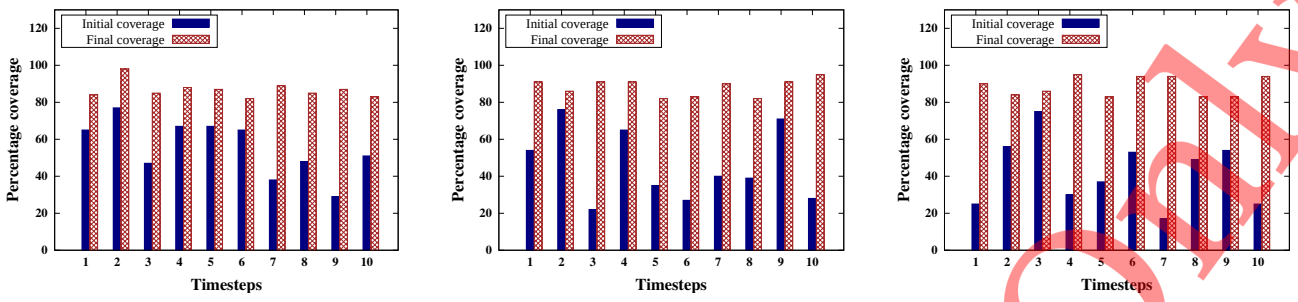
Proof: Let there exist a state, ψ_k, such that $\alpha_k \geq P_S$. The policy in MDP tries to optimize Equation (15). Thus, any state, ψ_i, in MDP state space, must have optimal action, which leads to a state, ψ_j, such that $R(psi_j) \geq R(psi_i)$. Moreover, by Lemma 1, there exists a path between $\psi_{initial}$ and ψ_k. The optimal action in $\psi_{initial}$ must lead to a state ψ_{final}, such that $R(\psi_{final}) \geq R(\psi_k)$. Thus, $R(\psi_k) > P_S \implies R(\psi_{final}) > P_S$. Therefore, we conclude that re-configured VS always provide coverage higher than P_S. ■

Algorithm 1 OPTIVE

INPUTS:
1: $Optimal\_action(\psi)$: Returns optimal action for the input state ψ w.r.t. optimal policy
2: $Apply(\psi, \mathcal{A})$: Returns the state, after state transition from ψ due to $\mathcal{A}$
OUTPUT:
3: ψ_{final} : The final state reached by applying optimal policy
PROCEDURE:
1: $\psi_1 =< 0, \phi, \phi >$
2: **while** true **do**
3: $\mathcal{A} = Optimal\_action(\Psi_1)$
4: $\psi_2 = Apply(\psi_1, \mathcal{A})$
5: **if** $(\phi_1 == \phi 2)$ **then**
6: $\psi_{final} = \psi_1$
7: $break$
8: **else**
9: $\psi_1 = \psi_2$
10: **end if**
11: **end while**
12: **return** ψ_{final}

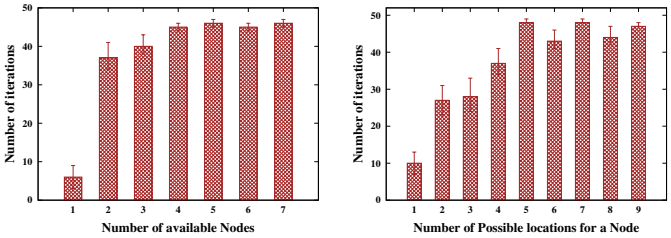
V. Performance Evaluation

To evaluate the performance of our proposed scheme, we consider the presence of $40-50$ sensor nodes, with the sensing range $90 - 95m$ over a simulation area of $500m \times 500m$. These sensor nodes have equal number of possible expected locations. We also consider the presence of 3 user applications for the simulation. Mathematically, $\forall i, |\mathbb{L}_i| = k$, where k is a constant. The algorithm for simulating OPTIVE is represented in Algorithm 1. Fig. 4 depicts the percentage of coverage in an application region in the presence of total number nodes $4, 5$, and 6. We observed that OPTIVE provides at least 80% coverage, which is significantly higher as compared to the initial percentage of coverage. Fig. 5 depicts the variation in the number of iterations required by the MDP process, considering the number of nodes available and number of possible expected locations of a sensor node. In Fig. 5a, we observe that in the presence of one sensor node, the number of iterations is less than 10. The possible reason for such a value is that, to find the optimal policy, number of actions considered by MDP are only 2. However, in the presence of $2 - 7$ sensor nodes, the number of iterations is significantly higher as compared to the case with one sensor node. Moreover, there is no significant change in the number iterations, when the total number of sensor nodes varies from $2-7$. Similarly, in Fig. 5b, we observe that the number of iterations is 10 when the expected possible number of locations is one. The possible reason behind this patten is that, when the number of expected location is one, the process becomes non-stochastic. Consequently, the number of iterations is less as compared to the other cases. In the presence of $2 - 4$ expected possible locations, the number of iterations vary between $25 - 35$. However, in the presence of $5 - 9$ expected possible locations, the variations in the number of iterations is significantly less, which is between 45 and 50. Additionally, we observe from Figs. 5a and 5b that the number of iterations get saturated after reaching a certain value of the total number of nodes and possible locations. Fig. 6 shows the change in number of states with the variation in number of sensor nodes and number of expected possible locations. We observe exponential increase in the number of



(a) # available sensor nodes=4 (b) # available sensor nodes=5 (c) # available sensor nodes=6

Fig. 4: Percentage of coverage



(a) Sensor nodes (b) Possible locations

Fig. 5: Change in iterations required for MDP

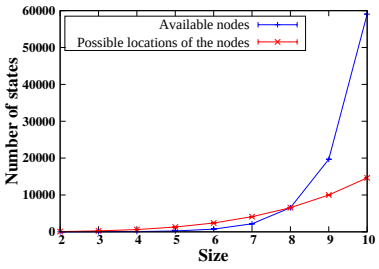


Fig. 6: Change in states with number of sensor nodes and number of expected possible locations

states in MDP with the increasing number of sensor nodes. We also observe that the number of states increases polynomially, with the increment in expected possible locations. Both the trends are in accordance with the state space size defined in the Proposition 1.

VI. Conclusion

In this work, we focused on the MSC architecture and proposed a scheme, OPTIVE, for configuring a VS. In MSC, the sensor nodes are attached with the mobile devices and attain mobility. These mobile devices may move dynamically at any time instant, which causes the distortion in the configuration of the VS. Consequently, such situation gives rise to the sensing hole in the application area. However, multiple device, equipped with sensor nodes, may be present in the application region. Thus, in such a scenario, our proposed scheme, OPTIVE, selects the sensor nodes optimally to allocate in the VS and cover the sensing hole. In future, we plan to extend this work, considering the Quality-of-Service of Se-aaS. Additionally, we plan to provide a data caching scheme for the MSC which can significantly reduce the end-user service time.

VII. Acknowledgment

The first author of this work is partially funded by project file no. 9/81(1293)/17 sponsored by the Council of Scientific and Industrial Research (CSIR), Govt. of India.

References

[1] S. Misra, S. Chatterjee, and M. S. Obaidat, "On Theoretical Modeling of Sensor Cloud: A Paradigm Shift From Wireless Sensor Network," *IEEE Systems Journal*, no. 99, pp. 1–10, 2014.

[2] S. Chatterjee, S. Misra, and S. Khan, "Optimal Data Center Scheduling for Quality of Service Management in Sensor-cloud," *IEEE Transactions on Cloud Computing*, no. 99, 2015.

[3] C. Roy, A. Roy, and S. Misra, "DIVISOR: Dynamic virtual sensor formation for overlapping region in IoT-based sensor-cloud," in *IEEE Wireless Communications and Networking Conference (WCNC)*, April 2018.

[4] M. Yuriyama and T. Kushida, "Sensor-Cloud Infrastructure - Physical Sensor Management with Virtualized Sensors on Cloud Computing," in *Proceedings of the* 13^{th} *International Conference on Network-Based Information Systems*, Sept 2010, pp. 1–8.

[5] S. Chatterjee, R. Ladia, and S. Misra, "Dynamic optimal pricing for heterogeneous service-oriented architecture of sensor-cloud infrastructure," *IEEE Transactions on Services Computing*, vol. 10, no. 2, pp. 203–216, March 2017.

[6] A. Chakraborty, A. Mondal, A. Roy, and S. Misra, "Dynamic Trust Enforcing Pricing Scheme for Sensors-as-a-Service in Sensor-Cloud Infrastructure," *IEEE Transactions on Services Computing*, 2018.

[7] N. M. S. Bose and S. Mistry, "Environment Monitoring in Smart Cities Using Virtual Sensors," in *the* 4^{th} *IEEE Int. Conf. on Future Int. of Things and Cloud*, 2016, pp. 399–404.

[8] S. Madria, V. Kumar, and R. Dalvi, "Sensor Cloud: A Cloud of Virtual Sensors," *IEEE Software*, vol. 31, no. 2, pp. 70–77, Mar 2014.

[9] S. Chatterjee and S. Misra, "Dynamic and Adaptive Data Caching Mechanism for Virtualization within Sensor-cloud," in *IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS)*, Dec 2014, pp. 1–6.

[10] A. Roy, S. Misra, and S. Ghosh, "QoS-Aware Dynamic Caching for Destroyed Virtual Machines in Sensor-Cloud Architecture," in *Proceedings of the* 19^{th} *International Conference on Distributed Computing and Networking*, ser. ICDCN '18. New York, NY, USA: ACM, 2018, pp. 28:1–28:7.

[11] D. T. Larose, *Regression Modeling*. Wiley-IEEE Press, 2006, pp. 33–92.

[12] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1st ed. New York, NY, USA: John Wiley & Sons, Inc., 1994.

[13] S. Bhatnagar and M. S. Abdulla, "A Reinforcement Learning Based Algorithm for Finite Horizon Markov Decision Processes," in *Proceedings of the* 45^{th} *IEEE Conference on Decision and Control*, Dec 2006, pp. 5519–5524.