Cache-Enabled Sensor-Cloud: The Economic Facet

Aishwariya Chakraborty, Ayan Mondal, and Sudip Misra
Department of Computer Science and Engineering,
Indian Institute of Technology Kharagpur
Kharagpur-721302, India
Email: {aishwariya.chakraborty, ayanmondal, smisra}@iitkgp.ac.in

***Abstract*—In this work, we propose a dynamic cache-based pricing scheme, named CASH, for service-oriented sensor-cloud. In sensor-cloud, the Sensor-Cloud Service Provider (SCSP) provisions Sensors-as-a-Service (Se-aaS) to multiple end-users based on a *pay-per-use* model. The service-requests of the end-users have heterogeneous data-rate requirements. In the cache-enabled architecture of sensor-cloud, these service-requests are served by the SCSP using either the Internal or the External cache, which incurs different costs to the SCSP. Thereby, the SCSP tries to maximize its own profit by distributing these service-requests, optimally, among the caches. Additionally, the SCSP ensures that the end-users are minimally charged. Existing literature fails to propose any pricing scheme for service-oriented sensor-cloud, while considering the cost incurred for data caching. In CASH, we propose a dynamic pricing model for sensor-cloud using dynamic coalition formation game with transferable utility. Using CASH, based on the preference relation of the partitions, we determine the optimal internal cache refresh rate, while maximizing the coalition value. Through simulation, we observe that the cost incurred by the SCSP reduces by 34.32-51.15% and the price paid by the end-users decreases by 9.60-17.47% as compared to the existing schemes. Additionally, CASH ensures 9.60-21.85% increase in the profit of the SCSP.**

***Index Terms*—Cache-based Service Oriented Architecture, Dynamic Pricing, Dynamic Coalition Formation Game, Merge-and-Split Algorithm, Sensor-Cloud**

I. Introduction

Sensor-cloud, an enhancement of traditional wireless sensor networks (WSNs), is designed by merging WSNs with the concepts of cloud computing. The main aim of sensor-cloud is to utilize the concept of virtualization of cloud for rendering Sensors-as-a-Service (Se-aaS). A centralized Sensor-Cloud Service Provider (SCSP), possessing the necessary cloud infrastructure, obtains physical WSNs on rent from their respective owners and provisions these resources as units of Se-aaS to the end-users. In lieu of these services, the SCSP receives revenue from the end-users as per their usage. Thus, the end-users remain abstracted from the various hardware-related complexities and their expenditures are significantly reduced.

In the existing literature, Chatterjee *et al.* envisioned a cache-enabled architecture [1] for sensor-cloud. In this architecture, the traditional sensor-cloud is equipped with two caches to store sensed data — Internal Cache (IC), which is present inside the cloud data-centers, and External Cache (EC), which is present in the sensor network gateways. The cache-based architecture of sensor-cloud improves the network performance and increases the efficiency of the system. Therefore, in this work, we explore the various service-oriented aspects of the cache-enabled sensor-cloud.

Similar to any cloud-based service-oriented architecture, sensor-cloud also follows a *pay-per-use* model for Se-aaS. Thus, pricing is an important aspect in sensor-cloud. There exist few works in the existing literature which propose pricing schemes for sensor-cloud, viz., [2], [3]. However, none of these schemes are suitable for the cache-enabled architecture of sensor-cloud. Additionally, the pricing schemes designed for cache-enabled cloud-based systems and WSN-based systems are not suitable for sensor-cloud as the architecture of sensor-cloud is based on heterogeneous SOA (combination of both hardware and infrastructure-as-a-service) [2]. Hence, there is a need for a dynamic pricing scheme suitable for the cache-enabled service-oriented architecture of sensor-cloud.

In this work, we propose a dynamic pricing scheme, named CASH, for sensor-cloud, while taking into consideration the cost incurred by SCSP for data caching. The proposed scheme leverages the dynamic caching mechanism of sensor-cloud to ensure maximum profit of the SCSP. Here, we consider that the internal and external caches have dynamic refresh rates, which are determined depending on the service requirements of the end-users at a particular time. The end-user applications, which require high data-rate services, are served directly from EC, whereas those requiring low data-rate are served from IC. The main contributions of this work are summarized as follows:

1) In this work, a dynamic pricing scheme, CASH, is proposed for sensor-cloud. Here, the price charged by SCSP depends on which cache is used to serve the request.

2) Using a dynamic coalition-formation game-theoretic approach, the optimal partition of the set of service-requests among IC and EC is obtained, while taking into account the data-rate requirements of the end-users at that time instant.

3) The proposed scheme also takes into account the cost incurred by the SCSP for serving the end-users from either of the caches and maximizes the overall profit of the SCSP.

4) Performance evaluation of CASH and comparative

978-1-5386-1734-2/18/$31.00 ©2018 IEEE

analysis of the proposed scheme with existing schemes is also presented in this work.

II. Related Works

In past few years, several works are done which focus on various aspects of sensor-cloud. The basic concept, architecture and theoretical modeling of sensor-cloud is presented by Yuriama *et al.* [4] and Misra *et al.* [5]. Sen *et al.* [6] developed a prototype of the sensor-cloud middle-ware, which is capable of provisioning virtual sensors. Thereafter, the technical aspects of sensor-cloud are studied by the researchers [7]–[10], viz., selection of optimal gateway node for transmitting data to the cloud, virtual sensor composition using resource-constrained nodes, optimal data-center selection for Se-aaS provisioning and risk assessment of sensor-cloud architecture. Few works also focused on the economic aspects of sensor-cloud. Chatterjee *et al.* [2] presented a dynamic pricing scheme for traditional sensor-cloud, while considering the satisfaction factor of the end-users, whereas Zhu *et al.* [3] proposed several pricing schemes for sensor-cloud, while taking into account various service parameters. However, these pricing schemes do not consider the cost incurred due to caching.

On the other hand, in existing literature, several research works focused on pricing. Awad *et al.* [11] presented a dynamic pricing scheme for device-to-device communication-enabled HetNets. A double auction-based dynamic pricing scheme for cloud in the presence of heterogeneous resources is studied by Zhang *et al.* [12]. Several other pricing schemes, viz., [13], [14], are proposed for efficient utilization of resources in cloud. However, these schemes neither consider caching cost, nor are applicable to sensor-cloud. Additionally, few cache-based schemes are proposed in existing literature. Araldo *et al.* [15] proposed a cost-aware caching mechanism for information centric networks. A contract theory based pricing scheme for commercial caching systems was proposed by Le *et al.* [16]. Feng *et al.* [17] explored the optimal placement of data caches among mobile nodes in mobile cloud environment. However, since sensor-cloud follows a heterogeneous service-oriented architecture, none of the proposed schemes in existing literature are suitable for sensor-cloud. Therefore, we infer that, there is a need to design dynamic pricing schemes suitable for cache-enabled sensor-cloud.

III. System Model

Sensor-cloud encompasses the integration of WSNs and cloud using the concept of virtualization for provisioning Se-aaS. In this work, we consider a sensor-cloud having a single SCSP with multiple registered end-users and sensor owners, as shown in Figure 1. Several heterogeneous physical sensor nodes, which are owned by the sensor owners, are deployed over varied geographic regions and used to serve the end-user service-requests. We consider that the sensor-cloud comprises of several base-stations or gateway nodes, which act as the sink nodes. These nodes collect the aggregated sensed data from the deployed sensor nodes and transfer it to the cloud. Each gateway node has an EC, which gets updated periodically with a time-interval decided by the SCSP. Additionally, we consider that, in order to efficiently serve the end-users belonging to different geographical regions, the sensor-cloud includes several IC-equipped data-centers. These data-centers are synchronized together for avoiding any discrepancy.

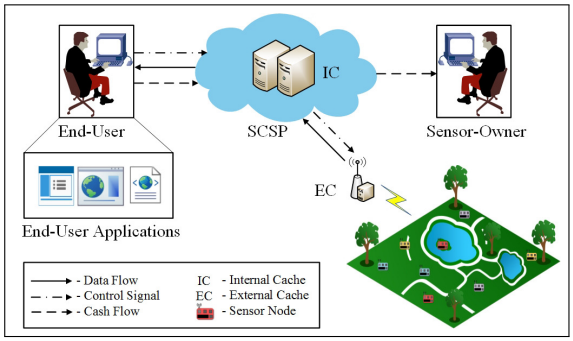


Fig. 1: Schematic Diagram of Cache-Enabled Sensor-Cloud

We consider that each end-user $j \in \mathcal{N}(t)$, where $\mathcal{N}(t)$ is the set of registered end-users at time instant t, request the SCSP to provide sensed data with data-rate r_j. Accordingly, the SCSP allocates a suitable data-center $d \in \mathcal{D}$, where $\mathcal{D}$ is the set of data-centers possessed by the SCSP, and gateway node $g \in \mathcal{G}$, where $\mathcal{G}$ is the set of gateway nodes registered with the SCSP. Thereby, we observe that there is a many-to-many relation between the data-centers and the gateway nodes, i.e., the internal and external caches. In this scenario, the optimal cache refresh rate for IC and EC needs to be decided in order to ensure the profit of the SCSP as well as QoS for the end-users. Based on the service requirements of the end-users, the SCSP decides the EC refresh rate R_E. We argue that $R_E = \max\{r_j | \forall j \in \mathcal{N}(t)\}$. However, the SCSP may vary IC refresh rate R_I for maximizing its own profit. Thereafter, the SCSP serves each end-user $j \in \mathcal{N}(t)$ with data-rate requirement $r_j \leq R_I$ from IC. On the other hand, the end-users having data-rate requirements $R_I < \forall r_j \leq R_E$ are served directly from EC.

IV. Dynamic Pricing Framework for CASH

A. Pricing Model of CASH

In CASH, we propose that the cache-based pricing model comprises of two types of pricing policies – *base-price* and *variable-price*. The pricing policies are discussed in the following sections.

1) Base-Price Policy: The base-price $p_B(t)$ is the definite amount charged at time instant t by the SCSP for serving each unit of data to the end-users. At time instant t, it is fixed for both IC and EC and is proportional to the ratio of the minimum and the maximum data-rate requirements of the end-users at that time. It is calculated as follows:

$$p_B(t) = \eta \tan^{-1}\left(\frac{\min\{r_j | \forall j \in \mathcal{N}(t)\}}{\max\{r_j | \forall j \in \mathcal{N}(t)\}}\right) \quad (1)$$

where η is a constant. In CASH, $p_B(t)$ depends on the range of data-rates for different services served by the SCSP.

2) Variable-Price Policy: The variable-price component signifies the extra price charged by the SCSP depending on the service requirements of the end-users. It is different for IC and EC and denoted as $p_V^I(R_I,t)$ and $p_V^E(R_I,t)$, respectively. In order to define the variable price component, we define the average normalized data-rate as follows.

Definition 1. *We define the average normalized data-rate $\zeta(t)$ as the ratio of the cumulative sum of the normalized data-rates demanded and the cardinality of the set of service-requests by the end-users at time instant t.*

$$\zeta(t) = \left[\frac{\sum_{j=1}^{|\mathcal{N}(t)|} \frac{r_j}{\max\{r_j | \forall j \in \mathcal{N}(t)\}}}{|\mathcal{N}(t)|} \right] \tag{2}$$

From Definition 1, we specify the boundary values of $\zeta(t)$ using Theorem 1.

Theorem 1. *Given that the set of service requests is non-empty, irrespective of the cardinality value of the set of service requests at time instant t, the average normalized data-rate $\zeta(t)$ always follows the following inequality:*

$$0 < \zeta(t) \leq 1 \tag{3}$$

Proof: We consider that the set of service requests $\mathcal{N}(t)$ is non-empty, i.e., $\mathcal{N}(t) \neq \{\emptyset\}$. Therefore, there exists at least one service-request having data-rate r_j, such that $r_j > 0$. Hence, we infer that —

$$0 < \zeta(t) < \infty \tag{4}$$

On the other hand, since $r_j \leq \max\{r_j | \forall j \in \mathcal{N}(t)\}$, the maximum value of the numerator is $|\mathcal{N}(t)|$. Thus, we conclude that $\zeta(t) \leq 1$. ■

a) Variable-Price for IC: The variable price $p_V^I(R_I,t)$ for IC varies linearly with the average normalized data-rate requirement of the end-user applications at time instant t. Mathematically,

$$\frac{\partial p_V^I(R_I,t)}{\partial \zeta(t)} > 0 \tag{5}$$

Thereby, $p_V^I(R_I,t)$ is computed as follows:

$$p_V^I(R_I,t) = \alpha \zeta(t) \tag{6}$$

where α is a constant.

b) Variable-Price for EC: We consider that, the variable-price of EC $p_V^E(R_I,t)$ varies *polynomially* having degree *two* with $\zeta(t)$. Mathematically,

$$\frac{\partial p_V^E(R_I,t)}{\partial \zeta(t)} > 0 \quad \text{and} \quad \frac{\partial^2 p_V^E(R_I,t)}{\partial \zeta(t)^2} > 0 \tag{7}$$

Therefore, $p_V^E(R_I,t)$ is expressed as follows:

$$p_V^E(R_I,t) = \beta \zeta(t)^2 + \gamma \zeta(t) \tag{8}$$

where β and γ are constants.

B. Cost Model of CASH

Similar to the pricing model, we model the cost incurred by the SCSP for provisioning Se-aaS, as a combination of two different cost policies — *caching cost* and *virtualization cost*, discussed as follows.

1) Caching Cost: Caching cost is the combination of cache storage and retrieval cost [15]. Cache storage cost is fixed for each data unit. However, cache retrieval cost depends on the data-rate requirements of the end-users. Therefore, we consider cache retrieval cost as caching cost in the rest of the paper. W_I and W_E denote the caching cost for IC and EC, respectively. Intuitively, we get that $W_I < W_E$.

2) Virtualization Cost: Virtualization cost Ψ_{vm} is the cost associated with the instantiation and maintenance of virtual sensor(s) and virtual machine(s) for serving a particular end-user application per unit time. We consider that Ψ_{vm} is fixed for all services.

C. Pricing Game Formulation

Based on the aforementioned pricing model, we formulate a *dynamic coalition-formation cooperative game with transferable utility* [18] to obtain the optimal IC refresh rate R_I such that the incoming service-requests received by the SCSP are distributed profitably among IC and EC. Here, the incoming service-requests of the end-users are considered to be the players of the game. The set of of players or service-requests is partitioned into two disjoint coalition sets, $\mathcal{N}_I(t)$ and $\mathcal{N}_E(t)$, which are to be served by the IC and EC, respectively. Subsequently, the SCSP tries to maximize the cumulative payoff of the utility functions, i.e., its profit, obtained from the IC and EC coalitions, as mentioned in Sections IV-C1 and IV-C2.

1) Utility Function for Internal Cache Coalition: The utility function $\mathcal{U}_I(R_I,t)$ for IC coalition is the cumulative profit obtained by the SCSP on serving the coalition-set $\mathcal{N}_I(t)$ of service-requests from IC at time instant t. It is a function of optimal IC refresh rate, R_I. $\mathcal{U}_I(R_I,t)$ comprises of two components — price function $P_j^I(R_I,t)$ and cost function $C_j^I(R_I,t)$, discussed as follows.

a) Price Function for IC: The price function $P_j^I(R_I,t)$ determines the price charged, at time instant t, by the SCSP for providing a service having data-rate requirement r_j for unit time. It is expressed as follows:

$$P_j^I(R_I,t) = \left[p_B(t) + p_V^I(R_I,t)\right] r_j, \quad \forall j \in \mathcal{N}_I(t) \tag{9}$$

b) Cost Function for IC: The cost function $C_j^I(R_I,t)$ depicts the cost incurred by the SCSP at time instant t for provisioning a service from IC for unit time to end-user j having data-rate requirement r_j. It is calculated as follows:

$$C_j^I(R_I,t) = \Psi_{vm} + r_j W_I, \quad \forall j \in \mathcal{N}_I(t) \tag{10}$$

The utility function $U_I(R_I,t)$ has a linear relation with the price and cost functions of the individual service-

requests, and follows the following inequalities:

$$\frac{\partial U_I(R_I,t)}{\partial P_j^I(R_I,t)} > 0 \quad \text{and} \quad \frac{\partial U_I(R_I,t)}{\partial C_j^I(R_I,t)} < 0 \tag{11}$$

Thus, based on Equations (9) and (10), we express the utility function $U_I(R_I,t)$ at time instant t as follows:

$$U_I(R_I,t) = \sum_{j=1}^{|\mathcal{N}_I(t)|} \left[P_j^I(R_I,t) - C_j^I(R_I,t)\right] \tag{12}$$

2) Utility Function for External Cache Coalition: We define the utility function $U_E(R_I,t)$ for EC coalition at time instant t as the cumulative profit of the SCSP for serving the set of service requests $\mathcal{N}_E(t)$ from EC. $U_E(R_I,t)$ also has two components — price function $P_j^E(R_I,t)$ and cost function $C_j^E(R_I,t)$, which are defined as follows.

a) Price Function for EC: The price charged by the SCSP per unit time for serving the service-requests of the end-users from EC determines the price component of the utility function for EC. $P_j^E(R_I,t)$ is expressed as follows:

$$P_j^E(R_I,t) = \left[p_B(t) + p_V^E(R_I,t)\right] r_j, \ \forall j \in \mathcal{N}_E(t) \tag{13}$$

b) Cost Function for EC: The cost incurred by the SCSP at time instant t for provisioning service having data rate requirement r_j for unit time from the EC is denoted by the cost function for EC $C_j^E(R_I,t)$. We express $C_j^E(R_I,t)$ as follows:

$$C_j^E(R_I,t) = \Psi_{vm} + r_j W_E \quad \forall j \in \mathcal{N}_E(t) \tag{14}$$

The utility function for EC $U_E(R_I,t)$ varies linearly with the individual price and cost functions of each service-request served from the EC. Thus, we have:

$$\frac{\partial U_E(R_I,t)}{\partial P_j^E(R_I,t)} > 0 \quad \text{and} \quad \frac{\partial U_E(R_I,t)}{\partial C_j^E(R_I,t)} < 0 \tag{15}$$

Hence, we express $U_E(R_I,t)$ mathematically as follows:

$$U_E(R_I,t) = \sum_{j=1}^{|\mathcal{N}_E(t)|} \left[P_j^E(R_I,t) - C_j^E(R_I,t)\right] \tag{16}$$

In order to maximize the overall profit, the SCSP needs to decide the optimal value of R_I, while making a trade-off between the coalition values of the IC and EC coalitions. Additionally, we consider that in CASH, the coalition value obtained from IC and EC coalitions need to be maximized. Therefore, the objective function for CASH is as follows:

$$\arg\max_{R_I} U_I(R_I,t)U_E(R_I,t) \tag{17}$$

Additionally, in CASH, the SCSP needs to satisfy the following constraints:

$$\min\{r_j\} \le R_I \le \max\{r_j\}, \quad \forall j \in \mathcal{N}(t) \tag{18}$$

$$\mathcal{N}_I(t) \cup \mathcal{N}_E(t) = \mathcal{N}(t) \tag{19}$$

D. Equilibrium in CASH

In order to reach the Pareto optimal partition in CASH, the preference relation among the elements of the super set of all possible partitions of $\mathcal{N}(t)$ needs to be evaluated, as per Definition 2.

Algorithm 1 Cache Refresh Rate Determination Algorithm

INPUTS:
1: $\mathcal{N}(t)$ ▷ Set of service-requests received by SCSP at time t
2: $r_j \forall j \in \mathcal{N}(t)$ ▷ Date-rate requirement of each service request at time t
3: Ψ_{vm} ▷ Virtualization Cost per unit time
4: W_I, W_E ▷ Caching cost per unit data for IC and EC
5: $\eta, \alpha, \beta, \gamma$ ▷ Constants decided by SCSP
OUTPUT:
1: R_I ▷ Cache refresh rate for IC
PROCEDURE:
1: Calculate $p_B(t)$ and $\zeta(t)$ using Equations (1) and (2), respectively;
2: Initialize $\mathcal{N}_I(t) \leftarrow \{\emptyset\}$ and $\mathcal{N}_E(t) \leftarrow \mathcal{N}(t)$;
3: **do**
4: $R_I \leftarrow \min\{r_j | \forall j \in \mathcal{N}_E(t)\}$;
5: $\mathcal{N}_I(t) \leftarrow \{j | \forall j \in \mathcal{N}(t)\ \&\ \forall r_j \le R_I\}$;
6: **for** $\forall j \in \mathcal{N}_I(t)$ **do**
7: Calculate $P_j^I(R_I,t)$ and $C_j^I(R_I,t)$ using Equations (9) and (10);
8: **end for**
9: Calculate $U_I(R_I,t)$ using Equation (12);
10: Set $\mathcal{N}_E(t) \leftarrow \mathcal{N}(t)/\mathcal{N}_I(t)$
11: **for** $\forall j \in \mathcal{N}_E(t)$ **do**;
12: Calculate $P_j^E(R_I,t)$ and $C_j^E(R_I,t)$ using Equations (13) and (14);
13: **end for**
14: Calculate $U_E(R_I,t)$ using Equation (16);
15: **while** $(\Delta U_I(R_I,t)U_E(R_I,t) \ge 0)$;
16: Return R_I;

Definition 2. *Given a set of end-users $\mathcal{N}(t)$ and the corresponding service-request rates $\{r_j | \forall j \in \mathcal{N}(t)\}$, the preference relation among two possible partitions $\mathcal{A}$ and $\mathcal{B}$ follows $\mathcal{A} \triangleright \mathcal{B}$, if and only if the following inequality holds:*

$$U_I(R_I^1,t)U_E(R_I^1,t) \ge U_I(R_I^2,t)U_E(R_I^2,t) \tag{20}$$

where $\mathcal{A} = \{A_I, A_E\}$ and $\mathcal{B} = \{B_I, B_E\}$. $A_I = \{r_j | \forall j \in \mathcal{N}_I^1(t)$ and $r_j \le R_I^1\}$, $A_E = \{r_j | \forall j \in \mathcal{N}_E^1(t)$ and $r_j > R_I^1\}$, $B_I = \{\{r_j | \forall j \in \mathcal{N}_I^2(t)$ and $r_j \le R_I^2\}$, and $B_E = \{r_j | \forall j \in \mathcal{N}_E^2(t)$ and $r_j > R_I^2\}\}$.

Based on Definition 2, the SCSP obtains the generalized Nash equilibrium (GNE) given a set of service-request rates, as defined in Definition 3.

Definition 3. *The SCSP ensures GNE of CASH by satisfying the following inequality:*

$$U_I(R_I^*,t)U_E(R_I^*,t) \ge U_I(R_I,t)U_E(R_I,t) \tag{21}$$

where $[\min\{r_j | \forall j \in \mathcal{N}(t)\} \le R_I^*, R_I \le \max\{r_j | \forall j \in \mathcal{N}(t)\}]$, R_I^* is the optimum cache refresh rate of IC and R_I represents all possible cache refresh rates, such that $(R_I \ne R_I^*)$. Hence, given a set of service-request rates $\{r_j | \forall j \in \mathcal{N}(t)\}$, CASH ensures that there must be an optimum cache refresh rate R_I^* of IC.

E. Proposed Algorithm

In CASH, the SCSP needs to determine the optimal IC refresh rate R_I in order to ensure profit by serving the requests of the end-users. To achieve the aforementioned objective, we use a *dynamic coalition formation game*

with transferable utility to form two coalitions of service-requests for IC and EC, dynamically, and maximize the coalition values. Motivated by the *Merge-and-Split Algorithm* [18], we determine the optimum IC refresh rate R_I^* based on the *preference relation* of the *collections of coalitions* as discussed in Algorithm 1.

V. Performance Evaluation

A. Simulation Parameters

In this section, we present the performance evaluation of the proposed scheme, CASH, in the cache-enabled architecture of sensor-cloud. For simulation, we consider a sensor-cloud environment in MATLAB consisting of a single SCSP and variable number of end-users. The number of end-users requesting the SCSP for Se-aaS at time instant t is varied from 100 to 10000, as mentioned in the Table I. Additionally, we also vary the maximum data-rate requested by the set of end-users from 100 to 1000 *packets/sec*. Thereby, we analyze the effect of these parameters on the percentage of end-users served from IC, total price charged from the end-users, the total cost incurred by the SCSP and the profit of the SCSP.

TABLE I: Simulation Parameters

Parameter	Value
Simulation Time	30 simulation hours
Number of Internal Cache	1
Number of External Cache	1
Number of service requests	100, 1000, 10000
Maximum requested data-rate	100, 250, 500, 750, 1000 packets/sec
Cost for single VM maintenance	100 units per unit simulation time
Caching cost for IC	10 units
Caching cost for EC	100 units

B. Benchmarks

We compare the proposed scheme, CASH, with two benchmark schemes — DADCM [1] and ED. In DADCM, Chatterjee *et al.* proposed a dynamic and adaptive caching scheme for sensor-cloud. The authors considered that the refresh rate for EC is dependent on the rate of change of the environment and that for IC varies with the rate of change of EC. Thus, DADCM is mostly suitable for event-driven applications. In case of highly dynamic environment, EC and IC are updated at almost same rate. Therefore, DADCM serves majority of the service-requests from EC and the corresponding end-users are charged according to the EC pricing. On the other hand, ED is a static scheme which considers equal division of the service requests among IC and EC. It is suitable for both event-driven and periodic applications. In ED, half of the end-users are charged based on IC pricing and the other half based on EC pricing.

C. Performance Metrics

In this work, we consider the following metrics to evaluate the performance of the proposed scheme, CASH.

Cost incurred by SCSP: We evaluate the total cost incurred by the SCSP for provisioning hardware and computational resources to the end-users for unit service-time.

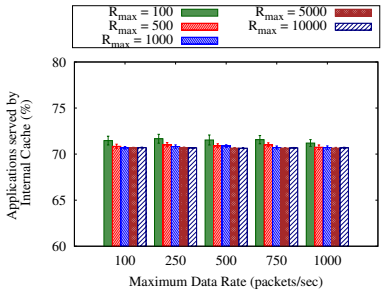


Fig. 2: Percentage of end-users served using Internal Cache

Since data is retrieved more frequently from EC than from IC, the cost incurred by the SCSP to provision services from EC is higher than that in case of IC.

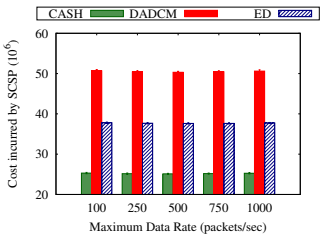
Price paid by end-users: The price per unit service-time, which is to be paid by the end-users to the SCSP for Se-aaS, depends on the cost incurred by the SCSP to provision the services requested by the end-users. Therefore, as evident from the Equations (9) and (13), the end-users served from EC have to pay higher price per unit service-time as compared to those served from IC.

Profit of SCSP: The difference of the price charged by the SCSP from its end-users and the actual cost incurred by the SCSP for provisioning Se-aaS to the end-users is the profit of the SCSP.

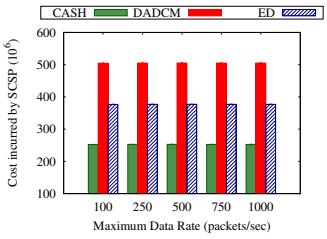
D. Results and Discussions

For simulation, we consider that at time instant t, SCSP receives heterogeneous service-requests having varied data-rate requirements from multiple end-users. We assume that the regions of interest have a highly dynamic environment and the SCSP is capable of serving every incoming service-request either using IC or EC. Therefore, in case of DADCM, every service-request is served from the EC, whereas, in case of ED, the percentage of users served from either of the caches remains fixed at 50%. However, we observe that, using CASH, the percentage of end-users served using IC (and hence, EC) varies with the overall distribution of the service-request rates. Figure 2 depicts that 70.5-72.15$\%$ of incoming service-requests at each time instant are served from IC using CASH.

From Figure 3, we observe that the cost incurred by the SCSP to serve the requests of the end-users is improved by 49.95-51.15$\%$ using CASH than using DADCM. This is because, in a highly dynamic environment, DADCM serves majority of requests from EC, resulting in increased service provisioning cost. On the other hand, using CASH, the cost incurred by SCSP decreases by 32.97-34.32$\%$ compared to using ED. Additionally, Figure 4 depicts that using CASH, the price charged by the SCSP from the end-users decreases by 16.71-17.47$\%$ and 9.08-9.6$\%$, than using DADCM and ED, respectively. This can be attributed to the fact that, among the three schemes, CASH ensures optimal distribution of the service-requests among IC and EC. Moreover, from Figure 5, we infer that the profit

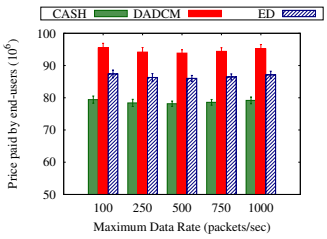


(a) # Service Requests = 1000

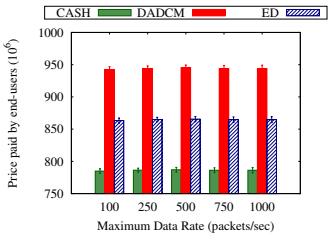


(b) # Service Requests = 10000

Fig. 3: Cost incurred by the SCSP

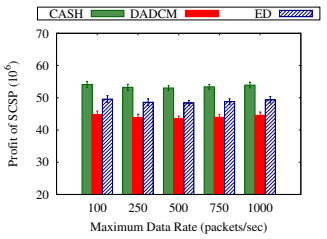


(a) # Service Requests = 1000

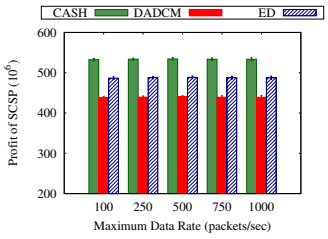


(b) # Service Requests = 10000

Fig. 4: Price paid by the end-users



(a) # Service Requests = 1000



(b) # Service Requests = 10000

Fig. 5: Profit of the SCSP

earned by the SCSP also increases by 20.83-21.85% and $9.43 - 9.60\%$ using CASH, as compared to using DADCM and ED, respectively. Hence, we conclude that CASH is economically more preferable to the SCSP as well as the end-users than the other two existing schemes.

VI. Conclusion

In this work, a dynamic pricing scheme, named CASH, for cache-enabled sensor-cloud using dynamic coalition formation game is presented. Using CASH, the SCSP ensures its maximum profit, while serving the end-user service-requests. Additionally, the proposed scheme, CASH, adds dynamism to the pay-per-use model of sensor-cloud, while making Se-aaS more economic for the end-users. Simulation results depict that the profit of the SCSP increases and the price paid by the end-users decreases using the proposed dynamic pricing scheme as compared to the existing schemes.

This work can be extended by exploring Se-aaS pricing scheme in the presence of heterogeneous virtual sensors. Additionally, as an extension of CASH, the variable maintenance cost incurred by the SCSP in the presence of heterogeneous physical sensor nodes can be incorporated.

References

[1] S. Chatterjee and S. Misra, "Dynamic and Adaptive Data Caching Mechanism for Virtualization within Sensor-Cloud," in *Proc. of IEEE Int. Conf. on ANTS*, Dec 2014, pp. 1–6.

[2] S. Chatterjee, R. Ladia, and S. Misra, "Dynamic Optimal Pricing for Heterogeneous Service-Oriented Architecture of Sensor-cloud Infrastructure," *IEEE Trans. on Serv. Comp.*, vol. 10, no. 2, pp. 203–216, 2017.

[3] C. Zhu, X. Li, V. C. M. Leung, L. T. Yang, E. C. H. Ngai, and L. Shu, "Towards Pricing for Sensor-Cloud," *IEEE Trans. on Cloud Comp.*, 2017, DOI: 10.1109/TCC.2017.2649525.

[4] M. Yuriyama, T. Kushida, and M. Itakura, "A New Model of Accelerating Service Innovation with Sensor-Cloud Infrastructure," in *Annual SRII Global Conf.*, Mar 2011, pp. 308–314.

[5] S. Misra, S. Chatterjee, and M. S. Obaidat, "On Theoretical Modeling of Sensor Cloud: A Paradigm Shift From Wireless Sensor Network," *IEEE Syst. J.*, vol. 11, no. 2, pp. 1084–1093, Dec 2017.

[6] A. Sen, V. P. Modekurthy, R. Dalvi, and S. Madria, "A Sensor Cloud Test-bed For Multi-model and Multi-user Sensor Applications," in *Proc. of IEEE WCNC*, Apr 2016, pp. 1–7.

[7] T. Ojha, S. Bera, S. Misra, and N. S. Raghuwanshi, "Dynamic Duty Scheduling for Green Sensor-Cloud Applications," in *Proc. of IEEE CloudCom*, Dec 2014, pp. 841–846.

[8] S. Chatterjee and S. Misra, "Optimal Composition of a Virtual Sensor for Efficient Virtualization within Sensor-Cloud," in *Proc. of IEEE ICC*, Jun 2015, pp. 448–453.

[9] S. Chatterjee, S. Misra, and S. Khan, "Optimal Data Center Scheduling for Quality of Service Management in Sensor-cloud," *IEEE Trans. on Cloud Comp.*, 2015, DOI: 10.1109/TCC.2015.2487973 .

[10] A. Sen and S. Madria, "Risk Assessment in a Sensor Cloud Framework Using Attack Graphs," *IEEE Transactions on Services Computing*, vol. 10, no. 6, pp. 942–955, Nov 2017.

[11] A. Awad, A. Mohamed, C. F. Chiasserini, and T. Elfouly, "Network Association with Dynamic Pricing over D2D-Enabled Heterogeneous Networks," in *Proc. of IEEE WCNC*, Mar 2017, pp. 1–6.

[12] Y. Zhang, K. Xu, X. Shi, H. Wang, J. Liu, and Y. Wang, "Continuous double auction for cloud market: Pricing and bidding analysis," in *Proc. of IEEE WCNC*, Apr 2016, pp. 1–6.

[13] D. M. Divakaran and M. Gurusamy, "Towards Flexible Guarantees in Clouds: Adaptive Bandwidth Allocation and Pricing," *IEEE Trans. on Para. and Dist. Syst.*, vol. 26, no. 6, pp. 1754–1764, Jun 2015.

[14] Y. Chi, X. Li, X. Wang, V. C. M. Leung, and A. Shami, "A Fairness-Aware Pricing Methodology for Revenue Enhancement in Service Cloud Infrastructure," *IEEE Syst. J.*, vol. 11, no. 2, pp. 1006–1017, Jun 2017.

[15] A. Araldo, M. Mangili, F. Martignon, and D. Rossi, "Cost-aware caching: Optimizing cache provisioning and object placement in ICN," in *Proc. of IEEE GLOBECOM*, Dec 2014, pp. 1108–1113.

[16] T. H. T. Le, N. H. Tran, P. L. Vo, Z. Han, M. Bennis, and C. S. Hong, "Contract-Based Cache Partitioning and Pricing Mechanism in Wireless Network Slicing," in *Proc. of IEEE GLOBECOM*, Dec 2017, pp. 1–6.

[17] Y. Feng, R. Stoleru, C. A. Chen, and G. G. Xie, "A Routing-Protocol-Independent Caching Framework for Mobile Clouds," *IEEE Transactions on Emerging Topics in Computing*, vol. 5, no. 3, pp. 353–366, July 2017.

[18] Z. Han, D. Niyato, W. Saad, T. Baar, and A. Hjrungnes, *Game Theory in Wireless and Communication Networks: Theory, Models, and Applications*, 1st ed. NY, USA: Cambridge University Press, 2012.