SGRM: Stackelberg Game-Based Resource Management for Edge Computing Systems

Antonis Karteris*, Manolis Katsaragakis*†, Dimosthenis Masouros*, Dimitrios Soudris*
*Microprocessors and Digital Systems Laboratory, ECE , National Technical University of Athens, Greece
†Katholieke Universiteit Leuven, Kasteelpark Arenberg 10, 3001 Heverlee, Belgium
*{akarteris, mkatsaragakis, dmasouros, dsoudris}@microlab.ntua.gr

***Abstract*—The incessant technological advancements of recent Internet of Things (IoT) networks have led to a rapidly increasing number of connected devices and workloads. Resource management is a key technique for such systems to operate efficiently. In this paper, we present SGRM, a game theory-based framework for dynamic resource management of IoT networks under CPU, memory, bandwidth and latency constraints. SGRM combines a novel execution time prediction mechanism along with Stackelberg games and Vickrey auctions in order to tackle the multi-objective problem of task offloading in a competitive Edge Computing system. We design, implement and evaluate our novel game theory-based framework over a real IoT system for a diverse set of interference scenarios and varying devices, showing that i) the proposed prediction mechanism can provide accurate predictions, achieving 2.3% absolute percentage error on average, ii) SGRM achieves near-optimal results and outperforms alternative solutions by up to 66.6% and iii) SGRM provides scalable, real-time and lightweight performance characteristics.**

***Index Terms*—IoT, Edge Computing, Resource Management, Task Offloading, Game Theory, Stackelberg Game, Vickrey Auction**

I. Introduction

Over the last years, the connection and networking of alternative heterogeneous devices that need to interact and cooperate with each other in order to offer more efficient and advanced control and monitoring services to everyday users composes a multidisciplinary and self-configuring network, namely Internet of Things (IoT). Traditionally, a significant subset of computation, storage and processing has been offloaded to the cloud infrastructure for the information inference [1].

However, the explosive growth of IoT devices and their increasing computing power have resulted in unprecedented volumes of data, computational, storage and energy resources [2]. Moreover, machine learning-enabled applications deployed at the edge of IoT networks and the rise of time-sensitive and Quality of Service(QoS)-critical requirements [3] impose numerous challenges for efficiently operating IoT networks.

Geographically centralized Cloud servers and datacenters cannot guarantee real-time and latency-sensitive computation services due to large round-trip delays, network congestion and service quality degradation [4]. In order to overcome the aforementioned limitations, Edge Computing provides an alternative and promising platform paradigm to process tasks by pushing data inference to the Edge of the IoT network and alleviating the dependency on Cloud infrastructure.

This work has been partially funded by EU Horizon 2020 program under grant agreement No 825061 EVOLVE (https://www.EVOLVE-h2020.eu/).

In this work we target the well-established Edge Computing architecture, consisting of interconnected IoT nodes and Gateways. Both nodes and Gateways possess limited computational resources (CPU), limited memory capabilities and bounded network bandwidths. Substantial data workloads produced on the IoT nodes require processing, either locally or offloaded to the Gateways under throughput constraints. Thus, an efficient resource management to dynamically allocate the shared Gateway resources and define the operating behavior of IoT nodes is essential. This composes an NP-complete problem, which corresponds to the Multidimensional Multiple-Choice Knapsack Problem (MMKP), thus computationally intractable [5].

The main goal of this research is to take advantage of the inherently competitive and distributed nature of edge computing architectures, in order to tackle the multi-objective problem of task offloading, under resource and time constraints. **The contributions of this work are:**

- **We present an Edge computing system architecture,** where IoT nodes can either execute their tasks locally or offload them to Edge Gateways. Our model captures the limited resources and constraints of the system and formulates the problem optimization objective as a deadline miss minimization problem.
- **We formulate and evaluate a novel Execution Time Prediction Mechanism** for heterogeneous applications and devices by identifying the special characteristics of the application, the available resources of the corresponding device and alternative interference levels.
- **We design and implement SGRM**, a distributed, scalable game-theoretic resource management framework that models the resource management problem as a Stackelberg game, utilizing Vickrey auctions.
- **We conduct an extensive experimental evaluation of the SGRM framework** in order to investigate the validity of our approach. We compare SGRM to a set of baseline algorithms and state-of-the-art resource management approaches over real hardware and network setups, showing that our proposed framework outperforms the compared approaches by up to 66.6%, while highlighting its distributed, scalable and lightweight nature.

The remainder of this paper is organized as follows: Section II presents related work. In section III we present the system model and the formulation of the target optimization problem. The execution time prediction mechanism and the proposed Stackelberg-based framework are detailed in Sec-

tion IV. Section V provides the experimental and comparative evaluation of the proposed resource management scheme, while Section VI concludes this work.

II. Related Work

Several works have been conducted in order to address the resource management challenges in Edge Computing systems. Authors of [5] address the problem of QoS management for IoT devices under bandwidth, battery and processing constraints, by proposing a centralized, pseudo-polynomial solution based on a dynamic programming approach, which enables reusing pre-computed solutions. In a similar setup, authors of [6], solve the task assignment problem using stochastic optimization, taking also into account the mobility of devices in the IoT network. Even though these approaches provide near-optimal solutions, they are not lightweight and lack in terms of scalability.

In order to overcome the inherent limitations of centralized approaches, market and game theory based approaches have been employed. Authors of [7] present a distributed market-based approach for resource management in Edge Computing systems, however they target only single-Gateway systems. In [8], authors design a game-theoretic task allocation algorithm combined with reinforcement learning, however this approach avoids the critical parameter of real-time requirements.

Regarding Stackelberg games (SG), these have also been widely utilized to tackle the resource management problem in Edge computing systems. Authors of [9] tackle a resource management problem utilizing a reverse Stackelberg model, aiming to achieve maximum revenue of the target system. Authors in [10] combine an SG approach with a many-to-many matching game to design a framework for resource allocation in three-tier Edge networks. Both solutions cannot be applied in real-time systems as they exhibit increased latency overhead.

Although research has illuminated the potential impact of efficient resource management in Edge Computing systems, no study to date, according to our knowledge, has incorporated Stackelberg games and auctions in order to produce an efficient, decentralized solution for allocating latency-sensitive tasks and managing the limited resources of Edge Computing systems.

III. System and Problem Formulation

Our target system architecture consists of X IoT Nodes (IoT Nodes) and Y Edge Gateways. The former are resource-constrained as far as CPU, memory and network capabilities are concerned and can process locally or offload tasks for execution to the latter.

Each IoT Node is assigned a unique identifying value $x \in \{1, ..., X\}$, and described by a tuple $D_x = \{\mathcal{C}_x, \mathcal{M}_x, \mathcal{N}_x, \mathcal{T}_x\}$. Variables $\mathcal{C}_x$, $\mathcal{M}_x$ and $\mathcal{N}_x$ denote the available processor, memory and network resources of node x respectively, while $\mathcal{T}_x$ indicates the set containing all the tasks τ_z^x generated on IoT node x. Similarly, every Gateway is defined by an identifying value $y \in \{1, ..., Y\}$, and described by a tuple $G_y = \{\mathcal{C}_y, \mathcal{M}_y, \mathcal{N}_y, \mathcal{T}_y\}$, where $\mathcal{C}_y$, $\mathcal{M}_y$ and $\mathcal{N}_y$ denote the available CPU, memory and network resources of Gateway y respectively. $\mathcal{T}_y$ indicates the set containing all the tasks τ_z^y offloaded on Gateway y. Every task of device x is uniquely defined by an identifying value z, and described by a tuple :

$$\tau_z^x = \{\mathcal{CL}_z, \mathcal{ML}_z, \mathcal{NL}_z, \mathcal{CG}_z^y, \mathcal{MG}_z^y, \mathcal{NG}_z^y, \Omega_z, ct_z, et_z, y_z\} \quad (1)$$

where $\mathcal{CL}_z$, $\mathcal{ML}_z$ and $\mathcal{NL}_z$ signify the amount of CPU, memory and network bandwidth required for the task z to be executed locally. Similarly, $\mathcal{CG}_z^y$, $\mathcal{MG}_z^y$ and $\mathcal{NG}_z^y$ signify the amount of CPU, memory and network bandwidth required for the task z to be executed on Gateway y. Ω_z signifies the deadline of the task z, while ct_z denotes the termination time of task z. Variable et_z denotes the estimated execution time of task z, as derived by the *Execution Time Prediction Mechanism*(Sec. IV-B). We define o_x^z, a binary variable indicating whether task z was offloaded or not, and variable l_x^z, which is its complement. Finally, variable y_z denotes the Gateway where the task was offloaded to. For tasks that are not offloaded, y_z is set to zero. All key parameters are summarized in Table I.

TABLE I: Key System Model Parameters

Denotation	Description
$\mathcal{C}_x$,$\mathcal{M}_x$,$\mathcal{N}_x$	Available CPU, Memory and Bandwidth of device x
$\mathcal{C}_y$, $\mathcal{M}_y$, $\mathcal{N}_y$	Available CPU, Memory and Bandwidth of Gateway y
T_x, T_y	Set of tasks on device x and Gateway y
τ_z^x	Task z of device x
$\mathcal{CL}_z$, $\mathcal{ML}_z$, $\mathcal{NL}_z$	CPU, Memory and Bandwidth required to execute task z locally
$\mathcal{CG}_z^y$, $\mathcal{MG}_z^y$, $\mathcal{NG}_z^y$	CPU, Memory and Bandwidth required to execute task z on Gateway y
ct_z, et_z, Ω_z	Completion time, Estimated time & Deadline of task z of device x
y_z	Id of Gateway y that received task z, 0 otherwise
o_j^i	1 if task j of device i is offloaded, 0 otherwise
l_j^i	1 if task j of device i is executed locally, 0 otherwise

$$\text{minimize} \quad N_{missed} = \sum_{\forall x}(\sum_{\forall z} miss(\Omega_z^x)) \quad (2)$$

$$\text{where} \quad miss(\Omega_z^x) = \begin{cases} 0 & if\ ct_z^x \leq \Omega_z^x \\ 1 & otherwise \end{cases} \quad (3)$$

Gateway y:

$$\sum_{\forall x}(\sum_{\forall z} o_z^x \cdot \mathcal{CG}_z^x) \leq \mathcal{C}_y \quad \textit{CPU}$$

$$\sum_{\forall x}(\sum_{\forall z} o_z^x \cdot \mathcal{MG}_z^x) \leq \mathcal{M}_y \quad \textit{Memory} \quad (4)$$

$$\sum_{\forall x}(\sum_{\forall z} o_z^x \cdot \mathcal{NG}_z^x) \leq \mathcal{N}_y \quad \textit{Network} \quad (5)$$

IoT Node x:

$$\sum_{\forall z} l_z^x \cdot \mathcal{CL}_z^x \leq \mathcal{C}_x \quad \textit{CPU} \quad (6)$$

$$\sum_{\forall z} l_z^x \cdot \mathcal{ML}_z^x \leq \mathcal{M}_x \quad \textit{Memory} \quad (7)$$

$$\sum_{\forall z} l_z^x \cdot \mathcal{NL}_z^x \leq \mathcal{N}_x \quad \textit{Network} \quad (8)$$

Unique execution

$$o_j^k + l_j^k \leq 1, \forall j \in \mathcal{T}_k \quad (9)$$

The optimization objective of the target problem as expressed in Eq. 2 is to minimize the number of tasks that exceed their

deadline. The task arrival trend is unknown, since IoT Nodes can dynamically spawn new tasks. The solution is constrained by the available CPU, memory and network resources both on IoT Nodes and Gateways (Eq. 4-8). Moreover, each individual task has the requirement of *Unique Execution* as shown in Eq. 9, i.e., a task is prohibited to be executed both on an IoT Node and a Gateway. Our proposed solution aims at solving the optimization problem by dynamically designating where each task of the IoT nodes will be executed.

IV. SGRM: A Stackelberg Game-Based Resource Management Framework

A. Background and Main Proposed Concepts

The key idea of the proposed approach is to take advantage of the inherently competitive nature of IoT networks and implement a non-cooperative strategic game among IoT Nodes and Gateways. In order to model our approach, we utilize the following fundamental game-theory concepts:

- **Stackelberg games** [11], which are non-cooperative games where the participants are segregated into two groups and the game is played sequentially.
- **Vickrey auctions** [12] or sealed-bid, second-price auctions, where the highest bidder wins the item in question. The second-price property forces the agents participating in the auction to bid truthfully on the value of the auctioned item.

In our approach, we model the task allocation decision making process of a node after a Stackelberg game played between the corresponding node and the Edge Gateways. The IoT node acts as a *leader* in the Stackelberg game, evaluating the task and deciding whether it should be offloaded to one of the available Gateways. Afterwards, the Gateways participating in the Edge computing system act as *followers* in the game and evaluate the incoming task, informing the corresponding IoT node. Finally, the IoT node conducts a Vickrey auction utilizing the task evaluations as bids, and offloads the task to the highest bidder.

Towards this direction, we present SGRM, a framework composed of three discrete modules divided among IoT nodes and Edge Gateways: (a) a *Task Evaluation Module*, which is executed by every individual device (IoT nodes and Gateways) on the network and consists of the *Task Profiling Mechanism*, the *Execution Time Prediction Mechanism* and the *Utility Function*, (b) a *IoT Node Module*, which is executed on IoT Nodes and consists of the *Auction* and *Offloading mechanisms* and (c) a *Gateway Module*, which is executed on the Gateways, containing the *Bidding Mechanism*. The efficient combination of these mechanisms via data exchange between nodes and Gateways completes SGRM, leading to its outcome, i.e. the decision of where each individual task of the IoT nodes will be executed, either on the Gateway or locally. Figure 1 illustrates the operational overview of the SGRM framework steps.

B. Task Evaluation Module

Both IoT nodes and Gateways are equipped with a *Task Evaluation Module*, consisting of the *Task Profiling Mechanism*, the *Execution Time Prediction Mechanism* and the *Utility Function*. This module is responsible for a device to appraise an incoming task, estimate its execution time and quantify the computational and temporal cost of carrying it out.

Prior to a device's joining our target Edge Computing system, a pre-processing step of profiling the different types of tasks present in the system needs to be carried out on the device, and the effects of multiple tasks being executed in tandem need to be analyzed and quantified. To that end, the *Task Profiling Mechanism* (❶, ❻) executes a set of dedicated combinations of tasks to define alternative interference levels on the target device and records their execution times. The goal is to define a unique function for each individual device that will be utilized for predicting the execution time.

The process for evaluating a given task z on a device x is as follows: (1) *Chunk Gauging*: Task z is executed on the device, and its execution time is logged and quantized by a factor q, i.e. split into entities, namely $chunks_{z,x}$. Variable q is defined throughout experimentation. (2) *Execution Time Estimation*: A bundle of combinations of the different tasks present in the system is carried out in tandem, and the execution time $et_{z,x}$ for task z on device x is estimated as shown in Eq. 10 each time.

$$et_{z,x} = chunks_{z,x} \times n_x \tag{10}$$

where n_x denotes the number of tasks being executed on device x. (3) *Error compensation*: In order to reduce the error produced by the *Execution Time Prediction* (❷,❼) steps, we generate a logarithmic trendline of the estimation error percentage as a function of the current number of tasks being executed on the device x (Eq. 11).

$$compf_{z,x}(r_x) = \alpha_{z,x} \times \ln n_x + \beta_{z,x} \tag{11}$$

Constants $\alpha_{z,x}$ and $\beta_{z,x}$ are defined through extensive experiments. As a result of the aforementioned process, a tuple $(chunks_{z,x}, \alpha_{z,x}, \beta_{z,x})$ is generated for every task z and device x combination. The aforementioned tuple is passed on as input to the *Execution Time Prediction Mechanism*, which predicts for each incoming task z the execution time on device x as a function of the number of tasks currently running on the device (Eq. 12). The process remains the same for Gateway y.

$$et_{z,x}(n_x) = chunks_{z,x} \times n_x - compf_{z,x}(n_x) \tag{12}$$

Fundamental to the decision making of the framework is the *Utility Function* (❸,❽), which attempts to capture and quantify the cost of undertaking a particular task in terms of time and computational resources into a single value that acts as a bid in the sealed-bid auction orchestrated by the corresponding IoT node.

In order to successfully capture this value, every spawned task is assigned an initial reward r_z, set to 1 by default. Subsequently, two penalty functions are estimated and applied to this reward value, decreasing it in accordance with the workload being already executed on the device and the estimated execution time of the task. The first penalty function penalizes the final utility according to the estimated execution time of the task z, similarly to [13]. It becomes negative when the task is estimated to miss its deadline (Eq. 13).

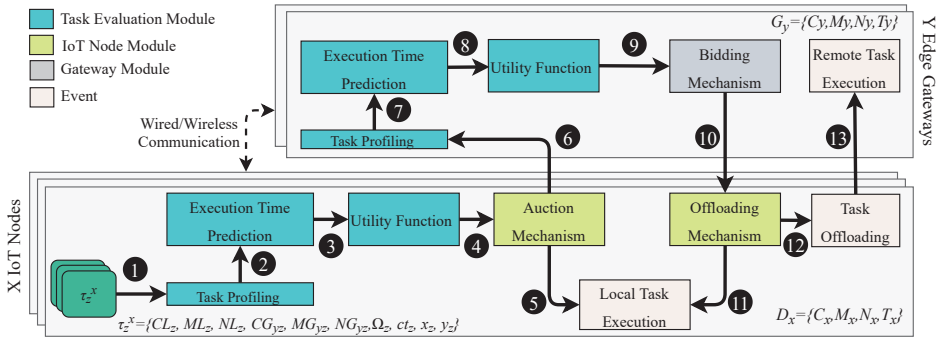


Fig. 1: Overview of SGRM Operation Flow among IoT Nodes and Edge Gateways

$$f_{z,x|z,x,y} = \frac{\Omega_z}{\Omega_z - et_{z,x|z,y}(n_{x|y}) - wt_{z,x|z,y} - tt_{z,x,y}} \quad (13)$$

where $wt_{z,x|z,y}$ denotes the waiting time for task z to begin execution on device x or on Gateway y. Variable $tt_{z,x,y}$ denotes the time required to transfer task z from IoT node x to Gateway y. In case that a task is not offloaded, $tt_{z,x,y}$ is set to zero. The second penalty function (Eq. 14) penalizes the utility further, according to the number of currently executing tasks.

$$p_{x|y} = \begin{cases} c^{n_{x|y}} & \text{if } f_{z,x|z,x,y} < 0 \\ \frac{1}{c^{n_{x|y}}} & \text{if } f_{z,x|z,x,y} > 0 \end{cases} \quad (14)$$

where $c > 1$ constitutes a weight factor selected through extensive experimentation. Finally, the utility function is extracted on the IoT Node x or on Gateway y, as shown in Eq. 15.

$$u_{z,x|z,x,y} = \frac{r_z}{f_{z,x|z,x,y} \times p_{x|y}} \quad (15)$$

C. IoT Node Module

The portion of SGRM executed on an IoT Node is composed of two fundamental parts, i.e. the *Auction Mechanism* (❹) and the *Offloading Mechanism* (❿). The former defines whether a task will be executed locally (❺) or whether it will participate as a candidate for potential offloading (❻). Each independent task is assigned a utility value, as described in IV-B. We define $threshold$ as the utility value over which the task should not be selected for offloading and is defined throughout extensive experimentation. The *Auction Mechanism* (❹) advertises all the corresponding tasks participating in the auction to the Edge Gateways. Each Gateway returns a bid for the corresponding task, as described in Section IV-D. Once all bids are collected by the IoT Node, the auction is performed. If the corresponding IoT node has made the highest bid then the task is executed locally (⓫), otherwise the Gateway with the highest bid is the winner of the auction. Next, the *Offloading Mechanism* (❿) is enabled and the input data is sent to the Gateway that won the auction (⓬, ⓭). Algorithm 1 summarizes the functionality of the *IoT Node Module*.

Algorithm 1: SGRM functionality on IoT Node Module

```
Data: IoT Node, Gateways Tuple, Set of Tasks
IoT Node-Algorithm(IoT Node, GatewaysTuple, Tasks):
1   for task in Tasks do
2       u = utility(task) /* estimate utility */
3       if u > threshold then
4           execute(task) /* local execution */
5       else
6           initiateAuction(task,GatewaysTuple) /* start auction */
7           waitForBids(GatewaysTuple) /* wait for bids */
8           bids = receiveBids(GatewaysTuple) /* receive bids */
9           winner = auctionWinner(bids) /* determine winner */
10          offload(task, winner) /* offload task to winner */
```

D. Gateway Module

The core functionality of the SGRM on the Gateways is dominated by the *Bidding Mechanism* (❾). For every individual task offloading request received in the corresponding Gateway, a utility value is calculated, as described in Section IV-B. The utility value is given as bid to the auction conducted on the corresponding IoT Node. The Gateway that wins the auction conducted on the IoT Node receives the corresponding task and input for execution. The functionality of the *Gateway Module* is summarized in algorithm 2.

Algorithm 2: SGRM functionality on Gateway Module

```
Data: Gateway, IoT Nodes Tuple, Set of Tasks
Gateway-Algorithm(Gateway, IoTNodesTuple, Tasks):
1   /* Repeat for all received tasks*/
2   for task in Tasks do
3       u = utility(task) /* estimate utility */
4       if u > threshold then
5           /* If utility over threshold: */
6           sendBid(task,IoTNodesTuple,u) /* send bid */
7           winner = receiveAuctionWinner(task) /* learn winner of
              auction */
8           if winner is Gateway then
9               /* if Gateway is the winner */
10              receive(task) /* receive task data */
11              execute(task) /* start execution */
```

E. Implementation

The SGRM framework consists of the core functionality logic, as described previously, and is responsible for monitoring, coordinating and deploying the target system and the

applications. The SGRM framework has been implemented using the Python programming language, and the wireless communication between the IoT Nodes and Gateways is achieved through utilizing the MQTT communication protocol. Moreover, aiming to avoid the latency overhead added by an application at startup, each application's process is launched in the background prior to data arrival, and remains idle until the reception of the corresponding input, i.e. the startup latency overhead is tackled once. Moreover, the components of SGRM are containerized, utilizing the Docker platform. The SGRM framework is publicly available with an open source licence[1].

V. Experimental Evaluation

A. Experimental Setup

To evaluate SGRM, we implement an IoT network that consists of heterogeneous embedded devices. We utilize three Raspberry Pi 4 Model B's (Quad-core Cortex-A72 @ 1.5GHz, ARMv7 32-bit Architecture), one Jetson TX1 (Quad-core Cortex-A57 @ 1.9 GHz, ARMv8 64-bit Architecture), one Jetson Nano (Quad-core Cortex-A57 @ 1.43 GHz, ARMv8 64-bit Architecture) and two powerful virtual machines (Octa-core Intel Xeon @ 2 GHz, x86 64-bit Architecture). All devices are equipped with 4GB of RAM.

We utilize four real-life machine learning applications with alternative resource requirements. The focus is on real-world benchmarks rather than synthetic, thus making the evaluation of our approach representative of performances of applications running on real production. More specifically, we utilize *Deepspeech* [14], *Facenet* [15], *Lanenet* [16] and *Retain* [17]. For each application the corresponding input data is offloaded from the IoT node to the target Gateway, respectively.

In order to properly evaluate our approach, we conduct a set of varying experiments in terms of arrival time and density of the incoming tasks. Specifically, we consider two distinct cases: In the first one, created using the Normal distribution, tasks are arriving almost simultaneously, thus creating a huge demand for resources. The second scenario involves cases,where simultaneous high peaks of demand are avoided and the Poisson distribution is selected for this goal. The tasks are randomly distributed among the devices and their deadlines are defined through extensive experimentation.

B. Experimental Results

Comparative Analysis: We evaluate SGRM against various resource management algorithms. First, in order to quantify the performance of SGRM against the optimal solution, we augment the comparison by adding an Oracle prediction mechanism, which possesses all the incoming task information a priori and provides the optimal decision making through an exhaustive search method. Moreover, we compare against two naive approaches: (1) Offload None, which executes all tasks without offloading any computation to the Gateways and (2) Offload All, on which all input workload is offloaded. Last but not least, we develop from scratch and compare against a state-of-the-art market-based resource management algorithm presented in [7], namely DMRM. Since DMRM is designed for systems with one Gateway, we examine its efficiency only on single-Gateway experiments.

[1]https://github.com/UphillD/edgebench

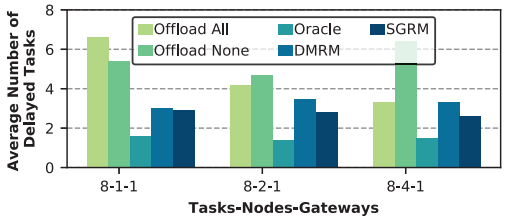
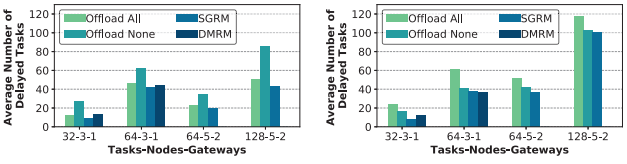


Fig. 2: #Tasks with missed deadlines (small-scale workload)

Small-scale Workload Analysis: In the first comparative experiment, we evaluate SGRM against all other alternatives over a single Gateway. However, due to the exhaustive behaviour of *Oracle* prediction, the experiment is conducted only for inputs with limited number of tasks and devices. Fig. 2 illustrates the results of the comparison. The x-axis denotes the triplets of the number of incoming tasks, the number of IoT nodes and the number of Edge Gateways of the examined scenario. The y-axis represents the number of tasks that exceed their deadline. As shown, SGRM clearly reduces the number of the delayed tasks compared to the naive approaches of Offload All by 42.5% and Offload None by 50.9% on average, and it also outperforms DMRM by 15.3% on average. Moreover, our approach leads to near-optimal results, as the proposed approach approximates the Oracle prediction results, closely. This initial evaluation illustrates that SGRM provides near-optimal results.

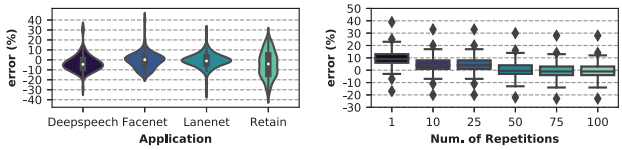
Larger-scale Workload Analysis: However, further investigation is required in order to evaluate the behaviour of our framework under increased workload, IoT Node and Gateway conditions. Fig. 3 depicts this evaluation, maintaining the same format for the X and Y axes as the previous experiment. In the first scenario, tasks are generated according to the Normal distribution, while in the second scenario we utilize the Poisson distribution. Our proposed framework achieves up to 25% fewer deadline misses compared to Offload All and up to 66.6% compared to Offload None for Normal distribution workloads, as shown in Fig. 3a. Similarly, for Poisson distribution, SGRM achieves 36.9% fewer deadline misses compared to Offload All and 17.7% compared to Offload None on average, as illustrated in Fig. 3b. In both scenarios, for single-Gateway experiments, SGRM achieves up to 30.7% fewer missed deadlines compared to the DMRM approach.

Execution Time Prediction Analysis: The functionality of our approach is directly linked to the efficiency of our proposed average time prediction mechanism. We evaluate the



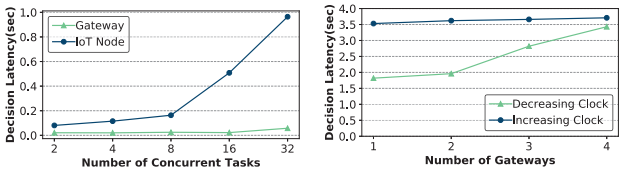
(a) Normal Distribution (b) Poisson Distribution

Fig. 3: #Tasks with missed deadlines (larger-scale workload)



(a) Error Percentage (b) Rate of Convergence

Fig. 4: Execution time prediction mechanism analysis



(a) Tasks Scalability (b) Gateways Scalability

Fig. 5: SGRM's tasks and Gateways scalability

prediction mechanism in terms of percentage error and the rate of convergence of our learning approach, i.e. how fast our system learns, as depicted in Fig. 4. The violin plots shown in Fig. 4a present the average execution time prediction error percentage for each application. We observe an absolute error rate of 2.3% on average across all applications. Moreover, in Fig. 4b, we evaluate the number of executions required over alternative interference levels for a single application in order to train our model and reach near-zero average error. We observe that our proposed mechanism converges for 50 repetitions.

Task Scalability Analysis: Moreover, we evaluate the scalability of SGRM, in terms of the number of concurrent tasks and Gateways, as illustrated in Fig. 5. Figure 5a highlights the decision latency of the SGRM framework on a single IoT Node and a single Gateway for up to 32 tasks generated concurrently. X-axis denotes the number of concurrent tasks generated on the IoT Node, while Y-axis denotes the decision latency in seconds. An efficient dynamic resource management scheme should be able to make decisions in an online manner. We observe that Gateway's decision time remains almost constant as tasks increase. Additionally, the IoT Node processing latency rises exponentially, which is caused by the increased number of auctions being carried out in the system sequentially. However, the decision latency still satisfies real-time requirements. Moreover, for more than 32 consecutive tasks, network bandwidth starts to be congested, i.e. the scalability depends on the specifications of the underlying network.

Gateway Scalability Analysis: Lastly, we evaluate the scalability of our framework over an increasing number of Gateway devices, as shown in Fig. 5b. In this experiment we dynamically add Gateways to the system. In the first scenario we add Gateways in decreasing CPU clock frequencies, while in the second scenario in increasing CPU clock frequencies. The decision latency of the latter gradually converges to the latency of the former, as we increase the number of Gateways, i.e., the decision time of the Gateways only depends to the computational capabilities of the most CPU-constrained device and no communication overhead is added by SGRM.

VI. Conclusion

In this paper we presented SGRM, a novel resource management and task allocation framework for Edge Computing systems. SGRM combines a novel execution time prediction mechanism along with Stackelberg games and Vickrey auctions to tackle the problem of efficient, real-time task offloading. We evaluate our framework over a real IoT network, showing that i) the proposed prediction mechanism can provide accurate predictions, achieving 2.3% absolute percentage error on average, ii) SGRM achieves near-optimal results and outperforms alternative solutions by up to 66.6% and iii) SGRM provides scalable, real-time and lightweight performance characteristics.

References

[1] F. Samie, L. Bauer, and J. Henkel, "IoT technologies for embedded computing: A survey," in *2016 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ ISSS)*, pp. 1–10, IEEE, 2016.

[2] A. Chowdhury and S. A. Raut, "A survey study on internet of things resource management," *Journal of Network and Computer Applications*, vol. 120, pp. 42–60, 2018.

[3] G. Tanganelli, C. Vallati, and E. Mingozzi, *Ensuring Quality of Service in the Internet of Things*, pp. 139–163. 06 2018.

[4] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, 2016.

[5] F. Samie, V. Tsoutsouras, S. Xydis, L. Bauer, D. Soudris, and J. Henkel, "Distributed QoS management for Internet of Things under resource constraints," in *International Conference on Hardware/Software Codesign and System Synthesis*, 2016.

[6] K. Cao, G. Xu, J. Zhou, T. Wei, M. Chen, and S. Hu, "QoS-Adaptive Approximate Real-Time Computation for Mobility-Aware IoT Lifetime Optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018.

[7] M. Katsaragakis, D. Masouros, V. Tsoutsouras, F. Samie, L. Bauer, J. Henkel, and D. Soudris, "DMRM: Distributed Market-Based Resource Management of Edge Computing Systems," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1391–1396, IEEE, 2019.

[8] S. Li, X. Hu, and Y. Du, "Deep reinforcement learning and game theory for computation offloading in dynamic edge computing markets," *IEEE Access*, 2021.

[9] Y. Chen, Z. Li, B. Yang, K. Nai, and K. Li, "A Stackelberg game approach to multiple resources allocation and pricing in mobile edge computing," *Future Generation Computer Systems*, vol. 108, pp. 273–287, 2020.

[10] H. Zhang, Y. Xiao, S. Bu, D. Niyato, F. R. Yu, and Z. Han, "Computing resource allocation in three-tier IoT fog networks: A joint optimization approach combining Stackelberg game and matching," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1204–1215, 2017.

[11] Z. Han, D. Niyato, W. Saad, T. Başar, and A. Hjørungnes, *Game theory in wireless and communication networks: theory, models, and applications*. Cambridge university press, 2012.

[12] L. M. Ausubel and P. Milgrom, "The Lovely but Lonely Vickrey Auction," in *Combinatorial Auctions*, ch. 1, pp. 17–40, MIT Press, 2006.

[13] D. Y. Zhang and D. Wang, "An integrated top-down and bottom-up task allocation approach in social sensing based edge computing systems," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pp. 766–774, IEEE, 2019.

[14] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Y. Ng, "Deep Speech: Scaling up end-to-end speech recognition," 2014.

[15] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2015.

[16] D. Neven, B. D. Brabandere, S. Georgoulis, M. Proesmans, and L. V. Gool, "Towards End-to-End Lane Detection: an Instance Segmentation Approach," 2018.

[17] E. Choi, M. T. Bahadori, J. Sun, J. Kulas, A. Schuetz, and W. Stewart, "RETAIN: An Interpretable Predictive Model for Healthcare using Reverse Time Attention Mechanism," in *Advances in Neural Information Processing Systems* (D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, eds.), vol. 29, Curran Associates, Inc., 2016.