A Query Processing Framework for Efficient Network Resource Utilization in Shared Sensor Networks

RAHUL KUMAR VERMA and K. K. PATTANAIK, Wireless Sensor Networks Laboratory, ABV-Indian Institute of Information Technology and Management, Gwalior, India

SOURABH BHARTI, Department of Information Technology, Indira Gandhi Delhi Technical University for Women, Delhi, India

DIVYA SAXENA and JIANNONG CAO, Department of Computing, The Hong Kong Polytechnic University, Hong Kong

Shared Sensor Network (SSN) refers to a scenario where the same sensing and communication resources are shared and queried by multiple Internet applications. Due to the burgeoning growth in Internet applications, multiple application queries can exhibit overlapping in their functional requirements, such as the region of interest, sensing attributes, and sensing time duration. This overlapping results in redundant sensing tasks generation leading to the increased overall network traffic and energy consumption. Existing approaches operate on data sharing among various tasks to minimize the upstream traffic. However, no existing work attempts to prevent the redundant task generation to reduce the downstream traffic. Moreover, the allocation of suitable sensor nodes to meet the Quality of Service (QoS) requirements of the queries is still an open issue. This paper proposes an end-to-end query processing framework (named, QueryPM) that first, calculates the functional requirements similarity among queries to prevent the redundant task generation. Then, it takes the QoS and functional requirements into account while allocating the tasks on the sensor nodes. Extensive simulations on the proposed approach show that downstream traffic, upstream traffic, and energy consumption reduced to 60%, 20%-40% and 40%, respectively, as compared to state-of-the-art mechanisms.

CCS Concepts: • **Networks** → **Sensor networks**; **Network performance analysis**; *Network management*;

Additional Key Words and Phrases: Shared sensor networks, Query pre-processing, Task allocation, Network Traffic

ACM Reference format:
Rahul Kumar Verma, K. K. Pattanaik, Sourabh Bharti, Divya Saxena, and Jiannong Cao. 2020. A Query Processing Framework for Efficient Network Resource Utilization in Shared Sensor Networks. 1, 1, Article 1 (March 2020), 29 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Traditionally, wireless sensor networks (WSNs) have been deployed for supporting single applications, which can lead to inefficient use of sensor nodes and low cost-benefit results [14]. Therefore, recent research efforts in the WSN domain are focusing on sharing its sensing and communication infrastructure for a multitude of applications towards the realization of the Internet of Things (IoT) paradigm and referred to as shared sensor networks (SSNs) [29][8][5]. It significantly reduces the

deployment cost and improves network resource utilization for large-scale sensing applications that are geographically co-located and therefore can be supported by the same infrastructure [9][21][11].

SSN provides sensed data corresponding to the queries of different IoT applications based on the queries' functional and quality of service (QoS) requirements. Functional requirements of a query include region of interest (RoI), sensing attributes (temperature, humidity, etc.), and sensing time-slot whereas QoS requirements include minimum delay, information accuracy, etc. Generally, query processing [2] in sensor network consists of four steps: (1) sensing tasks generation from queries and their allocation to the appropriate sensor nodes, (2) sampling of sensor data, (3) performing in-network processing on the sampled data, and (4) returning the results to the gateway. As a part of step (1), a number of independent sensing tasks for each query is generated and allocated to the relevant sensor nodes to measure the required sensing attributes. For example, *two different applications, namely "fire detection" and "air quality monitoring", are deployed over an SSN that provides sensed data concerning to the queries of these applications. To infer the occurrence of fire, queries from fire detection application generates three sensing tasks to measure the temperature, humidity, and smoke data from their RoI over a specific time-slot. On the other hand, air quality monitoring application sends the query to observe the presence of different types of gases (Carbon Dioxide (CO_2), Carbon Monoxide (CO), Nitrogen Dioxide (NO_2), etc.) and the presence of smoke to measure the air quality index (AQI).* For the successful execution of queries, their sensing tasks have to be allocated to the appropriate sensor nodes providing data in a timely manner [30]. In this example scenario, suppose queries arrive at the gateway to gather the information from the same RoI for the same time-slot. Consequently, sensing tasks generated by these queries will have one redundant sensing task (to gather smoke data) with overlapping RoI and time-slot. In a large-scale SSN where data-intensive applications are deployed, a huge number of redundant sensing tasks are generated. Apart from transmission of data to the gateway (upstream traffic) being one of the major factors of energy consumption [4], the execution of such queries also leads to high network traffic and energy consumption due to dissemination of redundant sensing tasks and redundant data sampling. Although various data sharing mechanisms [28][9][20] have been proposed to reduce the data transmissions but the issue of redundant task dissemination is not addressed yet. Since queries can have overlapping functional requirements, thus sending all of their sensing tasks into the network is not an efficient approach. To overcome this, we propose a query pre-processing mechanism to eliminate the redundant sensing tasks generation and dissemination to conserve the network resources.

As illustrated in Fig. 1, queries Q_1 and Q_2 from two different applications have overlapping spatial, temporal, and sensing attribute requirements. Suppose Q_1 generates three sensing tasks T_{11}, T_{13}, and T_{14} to sense three distinct attributes a_1, a_3, and a_4, respectively from RoI *R1* (dotted circles represent the RoI of respective queries) over a time-slot $[t_s^1, t_e^1]$. T_{22} and T_{23} are the sensing tasks corresponding to query Q_2 to sense attributes a_2 and a_3, respectively from RoI *R2* over a time-slot $[t_s^2, t_e^2]$. In this scenario, tasks T_{13} and T_{23} are said to be redundant since they correspond to the overlapping requirements of the queries. Therefore, similar or partially similar queries, first, leads to redundant sensing task generation that increases downstream traffic. Second, redundant sensing tasks generate redundant sensed data that results in increased upstream traffic and energy consumption. Apart from this, satisfying the functional and QoS requirements of the sensing tasks is essential for the successful execution of the corresponding queries.

In attempt to enhance the resource utilization in SSN, we identify the following research problems.

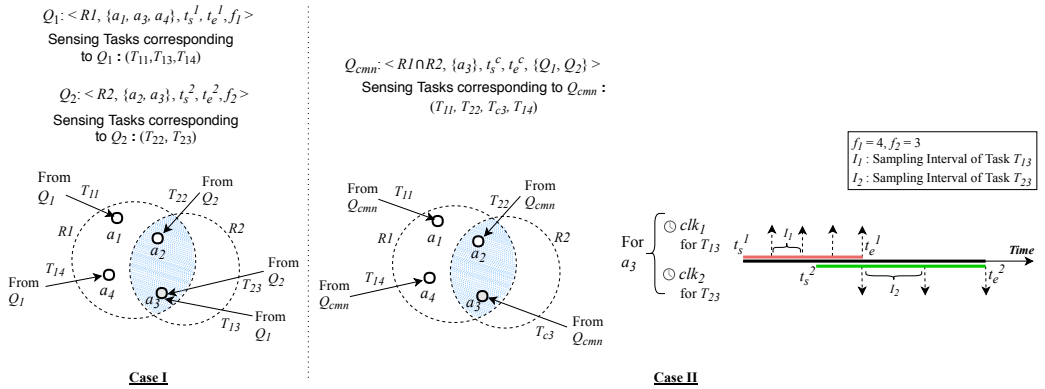


Fig. 1. Illustrating overlapped functional requirements in queries Q_1 and Q_2. A common query Q_{cmn} is created that serves the requirement of both the queries.

(1) Queries arriving at the gateway from different applications have overlapping functional requirements and therefore, result in redundant sensing tasks. This problem is named as ***min-RST problem*** (refer Section 4.2).
(2) Allocation of sensing tasks to the sensor nodes that can satisfy their functional and QoS requirements. This problem is named as ***Task allocation problem*** (see Section 4.2).

A naive method of the query processing is generating the sensing tasks individually and disseminating them into the network to sample the data independently, as shown in Fig. 1 (Case I). On the other hand, a common query Q_{cmn} can be created from Q_1 and Q_2 as they have overlapping functional requirements. Henceforth, Q_{cmn} generates a lesser number of sensing tasks than the total sensing tasks generated by Q_1 and Q_2 collectively. Q_{cmn} generates 4 sensing tasks T_{11}, T_{22}, T_{c3}, and T_{14} on the behalf of both Q_1 and Q_2, in which T_{c3} corresponds to the overlapping requirements of Q_1 and Q_2. Thus, Q_{cmn} avoids the generation of redundant sensing tasks i.e., T_{13} and T_{23} as illustrated in Fig. 1 (Case II).

T_{c3} is allocated to a sensor node in the overlapping RoI (shaded region in Fig. 1) of Q_1 and Q_2 to sense the common attribute a_3. The data is sent periodically at the desired frequencies of T_{13} and T_{23} using respective clock timers clk_1 and clk_2 maintained at the common node and updated based on their sampling intervals (see Eq. 1).

$$\begin{aligned} clk_1 &= clk_1 + I_1 \quad \blacktriangleright \textit{transmit sampled data at each } clk_1 \\ clk_2 &= clk_1 + I_2 \quad \blacktriangleright \textit{transmit sampled data at each } clk_2 \end{aligned} \tag{1}$$

An instance is shown in Fig. 1 where sampling intervals of tasks T_{13} and T_{23} are $I_1 = \frac{[t_s^1, t_e^1]}{f_1}$ and $I_2 = \frac{[t_s^2, t_e^2]}{f_2}$, and time-slots are $[t_s^1, t_e^1]$ and $[t_s^2, t_e^2]$, respectively. Sensor node samples the data during specified sampling intervals of T_{13} and T_{23}, and send the aggregated data to the gateway at each updated value of the respective clocks for the duration specified by the corresponding queries/tasks.

Considering the aforementioned scenario, we propose an end-to-end query processing framework, named as *QueryPM*, which minimizes the downstream traffic by reducing the number of sensing tasks generating from queries and allocate the generated tasks to appropriate sensor nodes. *QueryPM* comprises of two main modules: query pre-processing ($Query_p$) and task allocation ($Task_a$) where $Query_p$ calculates similarity among distinct pairs of queries based on their functional requirements and $Task_a$ allocates generated tasks to the sensor nodes fulfilling the

functional and QoS requirements of tasks. Queries having the similar functional requirements are pre-processed and *common queries* are created. On the other hand, remaining queries exhibiting no similarity to others get executed individually and termed as *individual queries*. Common queries generate non-redundant sensing tasks corresponding to the overlapping requirements of the parent queries, which minimizes the total number of sensing tasks and downstream traffic. $Task_a$ estimates the end-to-end delay between sensor nodes and the gateway before allocating sensing tasks to ensure their temporal deadlines. In summary, we have the following contributions in this paper.

(1) We design a query processing framework (*QueryPM*) that comprises of two main components: *query pre-processing* ($Query_p$) to pre-process the queries for minimizing redundant sensing tasks and *task allocation* ($Task_a$) for allocating the generated sensing tasks to most suitable sensor nodes fulfilling their functional requirements and QoS requirements, i.e., temporal deadline.
(2) We propose a novel query similarity scheme for $Query_p$ to identify similar queries for pre-processing. Similar queries result in *common queries* generating non-redundant sensing tasks. For $Task_a$, we further propose an *end-to-end delay estimation model* to identify most appropriate sensor nodes to allocate the generated sensing tasks.
(3) Extensive simulations show that *QueryPM* performs better than the other state-of-the-art mechanisms in terms of network traffic, energy consumption, and number of queries meeting their QoS requirements.

Rest of the paper is organized as follows. The comparison and analysis of related research efforts are discussed in Section 2. Section 3 defines the model of SSN, application queries, and sensing tasks. Section 4 discusses the proposed query processing framework and formulates the problem. The description of each module of *QueryPM* is discussed in Section 5. Performance evaluation of *QueryPM* is shown in Section 6 and finally Section 7 concludes this work.

2 RELATED WORK

Zhou et al. [29] proposed a cooperative caching-based mechanism to address the challenge of hosting multiple application in a shared infrastructure of WSN and perform query execution in an energy-efficient manner. It facilitate the reuse of sensory data for answering concurrent and forthcoming queries by storing the sensed data in caches at sink node and intermediate nodes. Binary strings are adopted to represent the query requests that helps to avoid redundant processing of similar (sub)query requests. Data sharing among multiple applications is another approach to reduce the communication cost and energy consumption of the WSN. Fang et al. [7] and Gao et al. [9] proposed a data sharing approach to minimize the overall length of sampling intervals by exploiting the time overlapping nature of the sensing tasks. In this way, data sampled in overlapping time duration can be shared among multiple application queries, thus reduces redundant data transmissions. However, authors assume that all the sampling intervals have same length and every sensing task overlaps with each other, which is too ideal.

Zhao et al. [28] proposed a task allocation mechanism with the objective of identifying a continuous sub-interval (sampling interval) within the overlapping time window of the allocated tasks to enable data sharing among them. Both Zhao et al. [28] and Fang et al. [7] addressed it as an *interval data sharing* problem which aims at the scheduling of sampling intervals to ensure maximum data sharing across various allocated sensing tasks on a sensor node. It identifies a continuous sub-interval (sampling interval) within the overlapping time region of each allocated task so that the data transmission is as less as possible in the network as well as satisfy the requirements of all the application queries. However, it requires dissemination of all the sensing tasks into the network followed by computing their overlapping sampling intervals at the node. Moreover, they did not

Table 1. Analysis of related research on various parameters

Mechanism	Application	Data sampling	Task requirement aware selection of sensor nodes	Dissemination of sensing tasks	Data Sharing
CATS [28]	-	Continuous	No	All	Yes
CAQO [29]	-	Continuous	No	-	Yes
TRAPS [3]	-	Discrete	Yes	One leader task from each group	Yes
2-Factor approximation [9]	Multi-application	Continuous	No	-	Yes
Task-Cruncher [20]	Multi-application	Continuous	No	All	Yes
Query optimization [22]	-	Discrete	No	-	Yes
QueryPM (Proposed)	Multi-application	Continuous	Yes	Non-redundant tasks	Yes

consider the the QoS requirements of the sensing tasks while allocating them. Directing the queries or sensing tasks always to the WSN can overload the network and poses several challenges to meet their QoS requirements. Mitici et al. [17] proposed an optimal query assignment scheme based on Discrete Time Markov Decision process to overcome this situation.

In contrast, we observe that different application *queries* can overlap not only in the time region but also in RoI and sensing attribute requirements. Our mechanism leverages this aspect to create common queries for providing appropriate services to the different queries having overlapping requirements. The overall objective of the proposed mechanism is to minimize the number of sensing tasks to be injected into the network corresponding to the queries from different applications deployed over the SSN. However, existing interval data sharing problem addresses the issue of communication redundancy only in upstream traffic. Since SSN is responsible to respond the queries arriving concurrently from different applications, individual execution of each query leads to high amount of energy consumption in sensing tasks dissemination and corresponding data transmission. On the other hand, we are targeting on the query pre-processing to analyze the requirements of the application queries to avoid generating redundant sensing tasks so that downstream traffic can be reduced. Our proposed mechanism eventually generates common sensing tasks serving the requirements of other similar queries by sharing the data, thus reducing network traffic and energy consumption significantly. To reduce the energy consumption associated with communication, various in-network processing [24][27][23] and data sharing [28][25] mechanisms have been proposed. However, the issues related to the task dissemination is not addressed yet and this paper focuses on reducing the number of sensing tasks to go inside the network by exploiting the functional requirement similarity among application queries. Table 1 shows the comparison of recent research efforts made in this direction.

Bestehorn et al. [2] have affirmative views on the issues of query dissemination and its routing towards appropriate sensors. To address this issue, they used probabilistic dissemination of the queries as an alternative of the flooding. In this approach, each sensor node rebroadcasts the query with a probability so that the number of nodes involved in query dissemination can be reduced. If we map their solution to our problem, probabilistic dissemination of the sensing tasks can minimize the involvement of the number of nodes in rebroadcasting but it can not eliminate the dissemination of the redundant sensing tasks, which is the main focus of this paper.

Bharti and Pattanaik [3] proposed a task pre-processing mechanism, called TRAPS, that clusters the similar sensing tasks based on their service type (attribute requirement) and spatial requirement. In addition to these parameters, tolerable delay of the sensing tasks is considered as their temporal requirements to choose one leader task from each cluster. The leader task has strict delay requirement and its execution ensures the fulfillment of other tasks in its cluster. The leader task(s) from each group are scheduled inside WSN and the fetched data is shared among their respective group members. The task clustering mechanism and leader task selection criteria proposed in [3] can not be applied in our problem scenario as they focus on task pre-processing and their assumptions of

sensing tasks are different. Moreover, TRAPS is not designed for continuous sampling tasks and thus, not applicable for our problem scenario. On the other hand, our focus is on pre-processing of the queries arriving from different applications to minimize the number of sensing tasks and their allocation.

Tavakoli et al. [20] presented an approach to reduce the communication and computation redundancy while providing the sensed data to the gateway. Communication redundancy minimizer is implemented on each sensor node to identify the common set of time instances at which samples are required from multiple sensing tasks allocated to that node. Since, all sensing tasks have to be sent on the designated nodes, it increases the downstream traffic in the sensor field. The communication redundancy is addressed only for the upstream traffic, while our work considers both upstream and downstream traffic minimization by pre-processing the queries. The computational redundancy is taken care by the central server or gateway after receiving the data from relevant sensors.

An SSN comprises of a gateway and several sensor nodes where gateway is a resourceful device with abundant processing power, storage, and energy. Whereas, sensor nodes are resource-constrained devices with limited processing capacity, storage, and energy. The main aim of the researchers is to consider the resources of sensor nodes as well as ensuring the successful execution of the queries. Existing mechanisms allocate all the sensing tasks of application queries individually to the relevant sensor nodes and schedule the sampling interval to maximize data sharing among allocated tasks [28][25]. However, many sensing tasks can have similar functional requirements, thus sending all the sensing task(s) into the network is not an efficient approach. Further, allocation of individual sensing tasks considering their functional and QoS requirements is another challenge. In [26], a data collection mechanism is proposed that aims to find the least-cost path from a source to sink such that the selected path meet the delay constraint as well as prolonging the network lifetime. In this work, we focus on minimizing the communication cost by eliminating the redundant sensing task dissemination and allocating the sensing tasks to the appropriate sensor nodes to fulfil their functional and QoS requirements.

3 BASIC CONCEPTS AND MODELS

This section presents relevant concepts, definitions and models used in the proposed mechanism. Table 2 outlines the symbols frequently used throughout this paper.

3.1 Shared sensor networks (SSN)

A shared sensor network (SSN) comprises of resource-constrained sensor nodes sensing different physical parameters and a resourceful gateway. The gateway bridges the gap between sensor network and the Internet through which users can send the queries to the SSN. An SSN can be defined as a tuple $\{\mathcal{G}, \mathcal{N}, \mathcal{A}\}$, where $\mathcal{G}$ is the gateway, $\mathcal{N}$ is a set of resource-constrained sensor nodes deployed in the sensor field, and $\mathcal{A}$ is a set of physical attributes that can be sensed from the sensor field.

In this paper, sensor nodes in SSN are assumed to be deployed in 2D plane divided into $r \times c$ rectangular grid cells, where r and c are the number of rows and columns, respectively. Fig. 2 depicts a sensor field divided into 12 grid cells, where symbols ▲, o, ★, and ♦ represent sensor nodes sensing $|\mathcal{A}| = 4$ different physical attributes a_1, a_2, a_3, and a_4, respectively. It is assumed that applications are aware of grid cell configuration and physical attributes that can be sensed from the sensor field. The gateway has the logical view of network topology that gets updated every time a sensor node dies. Sensor nodes are considered to be static in this paper. Sensor device definition (SDD) at the gateway contains information about the sensor nodes, such as *sensor_id, location, remaining_energy, sensing_attribute, grid_cell*, etc., which is used to identify the relevant sensor nodes for allocating the sensing tasks. However, identification of the relevant sensor nodes

Table 2. Summary of important symbols

Notations	Description
Q_i	Query i
T_{ij}	A sensing task from Q_i to sense an attribute a_j
Q_{cmn}	Common query
$\mathcal{T}_{cmn}$	Set of sensing tasks corresponding to Q_{cmn}
Q_{ind}	Individual query
$\mathcal{T}_{ind}$	Set of sensing tasks corresponding to Q_{ind}
$\mathcal{N}$	Set of deployed sensor nodes
N_j^k	A sensor node belonging to grid cell gc_k and senses attribute a_j
$\mathcal{A}$	Set of attributes that can be sensed from sensor field
atr_i ($\subseteq \mathcal{A}$)	Set of attributes required by Q_i
a_j ($\in atr_i$)	A physical attribute j to be sensed
reg_i	Region of Interest (RoI) of Q_i
t_s^i, t_e^i	Start and end timestamp of sensing duration of Q_i
f_i	Frequency of sensing the data for Q_i
I	Sampling interval
m	Number of queries arriving in a decision-window
ρ	Number of applications targeted on SSN

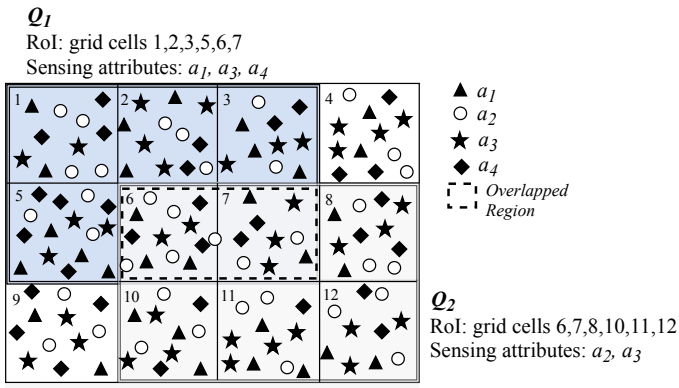


Fig. 2. Representation of RoI and sensing attribute requirements of queries Q_1 and Q_2 in a SSN.

for sensing tasks based on their functional requirements is not sufficient. The temporal deadline (QoS requirement) of each sensing task should be taken into consideration while selecting the sensor nodes for task allocation.

3.2 Application query

An application query Q_i is represented as a tuple $\langle A_i, reg_i, atr_i, t_s^i, t_e^i, f_i \rangle$, where A_i is the application ID, reg_i represents the RoI of Q_i, atr_i represents the sensing attribute to be retrieved from SSN for serving the query. t_s^i and t_e^i are the starting and ending time-stamps, respectively, and f_i is the sampling frequency. In addition to these functional requirements, each query contains a QoS requirement, i.e., temporal deadline before which the required data has to be provided in response to the queries.

We use binary strings [29] to represent RoI and sensing attribute requirements of an application query. If a sensor field is divided into $r \times c$ grid cells and can sense $|\mathcal{A}|$ types of sensing attributes then the length of binary strings corresponding to the RoI and sensing attribute requirements will have rc bits and $|\mathcal{A}|$ bits, respectively.

(1) Each grid cell (p, q), i.e., p^{th} row and q^{th} column, is represented by an identifier gc_k, where $k = ((p-1) \times c + q)$; c is the total number of columns in the sensor field. The length of the binary string of RoI in an application query is equal to the number of grid cells in the sensor field. The status of grid cells regarding their involvement in RoI of a query is indicated by binary values 0 or 1 at the bit locations same as gc_k in the binary string.

$$\text{bit value corresponding to } gc_k = \begin{cases} 1, & if\ gc_k \vDash reg \\ 0, & otherwise \end{cases}$$

Here, $gc_k \vDash reg$ represents that a grid cell gc_k lies in the RoI. For an instance, binary string representation of RoI of an arbitrary query Q_i can be represented as Eq. 2 if it includes grid cells 1, 2, 3, 5, 6, and 7.

$$reg_i : 111011100000 \ \blacktriangleright\ |reg_i| = 6 \tag{2}$$

(2) Application queries specify their sensing attribute requirements by $|\mathcal{A}|$ bit binary string. Let $\mathcal{A} = \{a_1, a_2, a_3, ..., a_{10}\}$ is a set of 10 different physical sensing attributes that can be sensed from an SSN. An application query can require all or subset of these attributes. The requirement of an attribute in an application query is indicated by value 1 at the bit location of concerned sensing attribute, otherwise it is 0. An example is illustrated by Eq. 3, representing three attributes a_1, a_3, and a_6 required by query Q_i.

$$atr_i : 1010010000 \ \blacktriangleright\ |atr_i| = 3 \tag{3}$$

Based on the aforementioned nomenclature, application query Q_1 (see example case in Fig. 2) can be represented as $\langle A_1, 111011100000, 1011, 1, 9, 4\rangle$ that demonstrates 3 different attributes a_1, a_3, and a_4 out of total 4 are to be sensed from RoI of 6 different grid cells out of 12.

– **Common query** (Q_{cmn}) is represented as a tuple $\langle reg_o, atr_c, ts_s^c, ts_e^c, \mathbb{Q}_p\rangle$, where $\mathbb{Q}_p = \{Q_i \mid 1 \leq i \leq m\}$ is a set of parent queries having similarity in their functional requirements. reg_o and atr_c represent the overlapping RoI and common sensing attribute requirements respectively among queries in $\mathbb{Q}_p$. t_s^c and t_e^c are respectively the start and end time of Q_{cmn} as shown in Table 3. Common queries contain the information of time-slots and frequencies of their parent queries so that corresponding sensing tasks can serve them at proper clock time (refer Eq. 1).

3.3 Sensing tasks

Each application query Q_i is decomposed into a set of sensing tasks $\mathcal{T}_i = \{T_{ij}^k \mid \forall j\ a_j \vDash atr_i, \forall k\ gc_k \vDash reg_i\}$ such that $|\mathcal{T}_i| = |reg_i| \times |atr_i|$ where $|reg_i|$ and $|atr_i|$ are the number of grid cells in RoI and required sensing attributes, respectively. A sensing task $T_{ij}^k \in \mathcal{T}_i$ is characterized as tuple $T_{ij}^k = \langle gc_k, a_j, t_s^i, t_e^i, f\rangle$, which implies that it corresponds to the query Q_i and requesting for an attribute $a_j \vDash atr_i$ from a grid cell $gc_k \vDash reg_i$. Rest of the parameters are inherited from its corresponding query. With reference to the Fig. 2, Q_1 is decomposed into ($|reg| \times |atr|$) 18 distinct sensing tasks as follows:

$$Q_1 = \langle A_1, 111011100000, 1011, 3, 12, 2\rangle$$
$$T_{11}^1 = \langle A_1, gc_1, a_1, 3, 12, 2\rangle, \quad T_{13}^1 = \langle A_1, gc_1, a_3, 3, 12, 2\rangle, \quad T_{14}^1 = \langle A_1, gc_1, a_4, 3, 12, 2\rangle$$
$$T_{11}^2 = \langle A_1, gc_2, a_1, 3, 12, 2\rangle, \quad T_{13}^2 = \langle A_1, gc_2, a_3, 3, 12, 2\rangle, \quad T_{14}^2 = \langle A_1, gc_2, a_4, 3, 12, 2\rangle$$

$$T_{11}^3 = \langle A_1, gc_3, a_1, 3, 12, 2\rangle, \quad T_{13}^3 = \langle A_1, gc_3, a_3, 3, 12, 2\rangle, \quad T_{14}^3 = \langle A_1, gc_3, a_4, 3, 12, 2\rangle$$
$$T_{11}^5 = \langle A_1, gc_5, a_1, 3, 12, 2\rangle, \quad T_{13}^5 = \langle A_1, gc_5, a_3, 3, 12, 2\rangle, \quad T_{14}^5 = \langle A_1, gc_5, a_4, 3, 12, 2\rangle$$
$$T_{11}^6 = \langle A_1, gc_6, a_1, 3, 12, 2\rangle, \quad T_{13}^6 = \langle A_1, gc_6, a_3, 3, 12, 2\rangle, \quad T_{14}^6 = \langle A_1, gc_6, a_4, 3, 12, 2\rangle$$
$$T_{11}^7 = \langle A_1, gc_7, a_1, 3, 12, 2\rangle, \quad T_{13}^7 = \langle A_1, gc_7, a_3, 3, 12, 2\rangle, \quad T_{14}^7 = \langle A_1, gc_7, a_4, 3, 12, 2\rangle$$

Similarly, Q_2 is decomposed into total 12 sensing tasks. Q_1 and Q_2 have overlapping RoI (grid cells gc_6 and gc_7), and a common attribute (a_3). If it is considered that their time-slots are also overlapping, there will be ($|reg_o| \times |atr_c| = 2 \times 1$) 2 identical sensing tasks from Q_1 and Q_2.

– A set of sensing tasks corresponding to a common query is represented as $\mathcal{T}_{cmn} = \{T_{ij} | 1 \leq i \leq |reg_o|$ and $1 \leq j \leq |atr_c|\} \cup \{T_p\}$. Sensing tasks in $\mathcal{T}_{cmn}$ provide data corresponding to the overlapping as well as non-overlapping requirements of their parent queries. First term of $\mathcal{T}_{cmn}$ represents a set of sensing tasks to sense the attributes specified in atr_c from overlapping RoI and formally termed as *common sensing tasks*. The second term ($\{T_p\}$) represents a set of sensing tasks for the non-overlapping requirements of the corresponding set of parent queries, and termed as *parent sensing tasks*.

3.4 Network model

We considered SSN as an undirected connected graph $G = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ represents a set of sensor nodes and $\mathcal{E}$ represents a set of all possible communication links among the sensor nodes. All sensors are equipped with the same radio interface as well as communication and sensing ranges. Moreover, sensors can detect all events of interest occurred within their sensing range. Let $u, v \in \mathcal{N}$ are two arbitrary nodes in the network and an edge $uv \in \mathcal{E}$ exists between them if $dis(u, v) \leq R$, where R is the communication range of u, and $dis(u, v)$ is the euclidean distance between u and v. The sensing tasks and sensed data are transmitted through multi-hop communication between gateway and sensor nodes. We assume that all the sensors in the network have a valid communication path to receive the sensing tasks and transmit their data to the gateway. There are two types of traffic considered in this paper: *downstream traffic*, that occurs in disseminating the sensing tasks from gateway to the relevant sensor nodes, and *upstream traffic*, that corresponds to the sensed data transmitted by sensor nodes to the gateway.

The instances of data transmission from sensor nodes to the gateway are considered as tree-based communication where sensor nodes are leaf nodes rooted towards the gateway and transmit their data through shortest path. The incurred delay in transmitting the sampled data from a sensor node $v \in \mathcal{N}$ to the gateway is estimated (see Section 5.2) before allocating the sensing tasks to the sensor nodes so that temporal requirements of the sensing tasks can be ensured.

4 PROPOSED FRAMEWORK AND PROBLEM STATEMENT

4.1 Proposed Framework

Fig. 3 shows framework of *QueryPM* that comprises of two key modules, query pre-processing module ($Query_p$) and task allocation module ($Task_a$). Application queries arriving from different applications are processed at a query pre-processing module ($Query_p$) that produces common queries (Q_{cmn}) and/or individual queries (Q_{ind}) based on the similarities in their functional requirements. Subsequently, sensing tasks are generated from these queries. This completes the pre-processing phase. The sensing tasks are now scheduled on the designated sensor nodes in the network through the task allocation module ($Task_a$). $Query_p$ interacts with $Task_a$ module and provide the details of generated sensing tasks. The functionality of $Query_p$ and $Task_a$ is implemented on the gateway node. Thus, the gateway node is fully responsible for pre-processing the incoming queries as well as sensing task generation and allocation to the suitable sensor nodes. On the other hand, a sensor

node's functionality is restricted to sensing and forwarding the sensed data towards the gateway. We also implement an end-to-end delay estimation model at the gateway for $Task_a$ to select the relevant sensor nodes ensuring to meet the temporal deadlines as well as functional requirements of the sensing tasks. $Query_p$ also interacts with *service provisioning* process at the gateway and provide the details of common queries and individual queries so that the sensed data received from tasked sensor nodes can be collaborated as per the requirements of the queries to provide them appropriate responses.

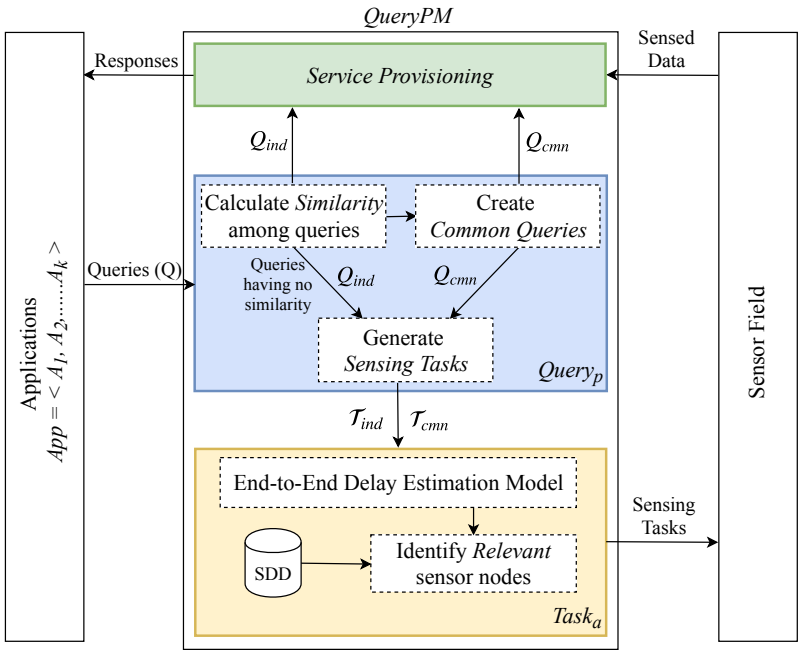


Fig. 3. Interaction among various modules of *QueryPM*

4.2 Problem Statement

Since SSN consists of energy-constrained sensor nodes, the cost of disseminating sensing task into the network can be estimated in terms of total energy consumption involved in forwarding sensing tasks to the relevant sensor nodes. Considering the network as a connected graph where the source nodes are the leaf nodes of the tree rooted at the gateway, the cost of tasks dissemination (C_{diss}) can be characterized by Eq. 4

$$C_{diss} = \sum_{\tau} \left[\sum_{n=1}^{d} (C_T^n + C_R^n) + C_R^n \right] \tag{4}$$

Where, τ is the total number of sensing tasks generated from a query, and d is the number of intermediate nodes between gateway and the source node where a sensing task is to be allocated. C_T and C_R are the energy cost for transmitting and receiving a task, respectively [12]. According to the Eq. 4, C_{diss} can be minimized by reducing the number of sensing tasks (τ) entering into the network subject to maximising the network lifetime. Since sensing tasks are responsible for the execution of the queries they are originated from, simply reducing the number of sensing tasks may

lead to unsuccessful query executions. However, if the common queries are created beforehand at the gateway, the number of sensing tasks entering into the network can be reduced, which further helps in minimizing C_{diss}. Thus, we define this problem as follows.

Problem 1 (*min-RST* problem): *The dissemination cost minimization problem is to minimize the **R**edundant **S**ensing **T**asks exposed to the SSN, under the constraint of successful execution of each query.*

In addition to the Problem 1, allocation of sensing tasks resulting from the common queries and individual queries to the "relevant" sensor nodes by considering their functional and QoS requirements is another challenge for the successful execution of the queries. The relevancy (rel_n) of a sensor node for a task corresponding to its functional requirements can be characterized by Eq. 5.

$$rel_n = f(\mathbb{S}_n, T_{ij}^k), \quad \forall n \in \mathcal{N} \tag{5}$$

Where, $\mathbb{S}_n = \{gc_k, a_j, E_{res}\}$ represents the node's specification. A sensor node is said to be relevant for a sensing task $T_{ij}^k \in \mathcal{T}_i$, if sensor node belongs to the grid cell gc_k and sense the data corresponding to an attribute $a_j \vDash atr_i$.

However, satisfying the functional requirements of the application queries is not sufficient in today's SSN scenario where real-time applications are often deployed in the realization of IoT. Thus, next challenge for the successful execution of queries is meeting their QoS requirements, i.e., temporal deadline ($R_{T_{ij}^k}^{temp}$) before which the required sensed data should reach to the gateway. To this end, this paper deals with another problem of task allocation defined as follows.

Problem 2 (*Task allocation* problem): *The problem of allocating sensing tasks considering their functional and QoS requirements is defined as a mapping function $\Gamma : T_{ij}^k \rightarrow N_j^k$ where $T_{ij}^k \in \mathcal{T}_i$, and $N_j^k \in \mathcal{N}$ is a relevant sensor node for T_{ij}^k such that $delay_{e2e} \leq R_{T_{ij}^k}^{temp}$.*

Where $delay_{e2e}$ is the end-to-end delay incurred in reaching the data from sensor node N_j^k to the gateway. $delay_{e2e}$ corresponding to each sensor node is estimated beforehand at the gateway for delay-aware decision making of allocating sensing tasks. It ensures the timeliness of the reception of the sensed data.

Lemma 1: *Given a set of queries $\mathbb{Q} = \{Q_1, Q_2, ..., Q_m\}$, the identification of a subset of queries ($\mathbb{Q} \subseteq \mathbb{Q}$) from which common queries can be created, such that total number of sensing tasks to be generated are minimum, is an NP-complete problem.*

Proof: Assume that there is a set of queries $\mathbb{Q} \subseteq \mathbb{Q}$ such that the overlapping in functional requirements among all its queries is maximal, i.e., $|reg_o|$ and $|atr_c|$ are maximal and $t_1 \oplus t_2 \oplus t_3 \oplus ..... \oplus t_m \neq 0$.

First, if $\mathbb{Q}$ is an optimal subset, none of its queries can be removed from $\mathbb{Q}$ otherwise it would cause an increment of $|\mathcal{T}_{cmn}|$ in total number of sensing tasks (refer **Lemma 2**). Then, we check each query and determine whether it can be removed and $\mathbb{Q}$ results in same number of sensing tasks. If any query in $\mathbb{Q}$ can not be removed, the subset $\mathbb{Q}$ can be proved to be optimal. The process of verifying can be completed in polynomial time. Thus, identification of a subset of $\mathbb{Q}$ for minimum number of sensing task generation is an NP problem.

Second, the "longest common sub-sequence" problem, which has been proved to be an NP-complete problem, is transformed to our problem. A longest common sub-sequence is a string x such that x is a sub-sequence of s_i for $i = 1, 2, ..., n$. Since the longest common sub-sequences of strings provide information about their similarity, this problem can be mapped to our problem with

the constraint that starting and ending index of the sub-sequence should be same in all the strings and only one contagious sub-sequence is possible.

The spatial and sensing attribute requirements of each query $Q_i \in \mathbb{Q}$ is represented as binary strings of length rc and $|\mathcal{A}|$, respectively. Thus, identification of a subset of queries, such that they have largest common sub-sequences in spatial and sensing attribute requirements with aforementioned constraint as well as satisfy the condition of time-slot overlapping (see **Definition 3**), is a hard or NP-complete problem. This finishes the proof.

5 QUERY PROCESSING FRAMEWORK (QUERYPM)

We propose two modules in *QueryPM* to solve the problems mentioned in Section 4.2. The detail description of each module is as follows.

5.1 Query Pre-processing Module ($Query_p$)

$Query_p$ eliminates the generation of redundant sensing tasks by pre-processing the similar or partially similar queries. The functioning of $Query_p$ can be divided broadly into three steps: (1) calculate functional requirement similarity (FRS_Q) for each distinct pair of queries, (2) create common query corresponding to the pair having maximum FRS_Q, (3) generate sensing tasks from Q_{cmn} and Q_{ind}. FRS_Q between two arbitrary queries Q_i and Q_j is calculated by Eq. 6.

$$FRS_Q(Q_i, Q_j) = \left(\frac{R_i \cdot R_j}{||R_i||\ ||R_j||}\right)\left(\frac{A_i \cdot A_j}{||A_i||\ ||A_j||}\right)(t_i \oplus t_j) \tag{6}$$

Where, $||R_i||$ and $||A_i||$ are the Euclidean norms of p-bit binary string representing RoI (reg_i) and q-bit binary string representing sensing attribute requirement (atr_i) of the query Q_i, respectively. $||R_i||$ and $||A_i||$ are calculated as $\sqrt{\sum_p (b_p^2)_i}$ and $\sqrt{\sum_q (b_q^2)_i}$, respectively, where b_p and b_q are the bit values (0 or 1) at different bit locations in the corresponding binary strings. $R_i \cdot R_j$ and $A_i \cdot A_j$ are calculated as $\sum_p ((b_p)_i \times (b_p)_j)$ and $\sum_q ((b_q)_i \times (b_q)_j)$, respectively. $t_i \oplus t_j$ represents time-slot overlapping between queries (see Table 3).

Algorithm 1 demonstrates the underlying steps of $Query_p$, which are applied on the queries arriving at the gateway within a specified *decision-window* $[dw_s, dw_e]$. Algorithm 1 results in one or more common and individual queries after pre-processing the queries. A common query constitutes overlapping functional requirements of the corresponding parent queries, which are identified as follows.

(a) **Overlapping RoI:** Given the binary strings of RoI of different queries arrived in a decision-window, the overlapped RoI is identified through *logical AND* operation. Eq. (7) shows the 12-bit binary strings for the RoIs of queries Q_i and Q_j arriving in a sensor field of 4×3 grid. Binary string reg_o represents the overlapping RoI constituting grid cells such that $\forall k\ gc_k \vDash reg_i\ \&\&\ gc_k \vDash reg_j$.

$$\begin{aligned} reg_i &: 111011100000 \\ reg_j &: 000001110111 \\ reg_o &: 000001100000 \end{aligned} \tag{7}$$

Definition 1: *The RoI reg_i and reg_j of two queries Q_i and Q_j respectively are said to be overlapping if,*

$$reg_i \wedge reg_j \neq \phi$$

Algorithm 1 Query Pre-processing

Initialization: $\mathbb{Q} = \{Q_1, Q_2, .., Q_m\}$ ► Set of m queries arrived in a decision-window
procedure $Query_p(\mathbb{Q})$
  Calculate FRS_Q for each pair of queries in $\mathbb{Q}$
  if $FRS_Q(Q_i, Q_j)$, $i \neq j$ is maximum **then**
    Identify t_s^c and t_e^c ► refer Table 3
    Identify reg_o ► refer Eq. 7
    Identify atr_c ► refer Eq. 8
    if Q_i is a common query **then**
      $\mathbb{Q}_p = \mathbb{Q}_p^i \cup Q_j$
      Create $Q_{cmn} = \langle reg_o, atr_c, t_s^c, t_e^c, \mathbb{Q}_p \rangle$
    else if Q_j is a common query **then**
      $\mathbb{Q}_p = \mathbb{Q}_p^j \cup Q_i$
      Create $Q_{cmn} = \langle reg_o, atr_c, t_s^c, t_e^c, \mathbb{Q}_p \rangle$
    else
      $\mathbb{Q}_p = Q_i \cup Q_j$
      Create $Q_{cmn} = \langle reg_o, atr_c, t_s^c, t_e^c, \mathbb{Q}_p \rangle$
    end if
  else
    Return $\mathbb{Q}$
  end if
  Remove Q_i and Q_j from $\mathbb{Q}$, and insert Q_{cmn}
  Go to step 2.
  $Task\_Generation(\mathbb{Q})$
end procedure
procedure $Task\_Generation(\mathbb{Q})$
  for each query $Q_i \in \mathbb{Q}$ **do**
    if Q_i is a common query **then**
      $\mathcal{T}_{cmn} = \{T_{ij}^k | \forall j\ a_j \vDash atr_c \text{ and } \forall k\ gc_k \vDash reg_o\} \cup \{T_p\}$
      $Task_a(\mathcal{T}_{cmn})$ ► **Algorithm 2**
    else
      $\mathcal{T}_{ind}^i = \{T_{ij}^k | \forall j\ a_j \vDash atr_i \text{ and } \forall k\ gc_k \vDash reg_i\}$
      $Task_a(\mathcal{T}_{ind})$ ► **Algorithm 2**
    end if
  end for
end procedure

(b) **Common sensing attributes:** Given the binary strings representing sensing attributes required by the queries, the common attribute requirements can be identified through a *logical AND* operation. Eq. (8) shows 10-bit binary strings atr_i and atr_j corresponding to the queries Q_i and Q_j, respectively. The binary string atr_c represents the common attributes which are requested by both the queries.

$$\begin{aligned} atr_i &: 1011000000 \\ atr_j &: 0110000000 \\ atr_c &: 0010000000 \end{aligned} \tag{8}$$

Definition 2: *Two queries Q_1 and Q_2 have common sensing attribute requirement(s) if,*

$$atr_i \wedge atr_j \neq \phi$$

(c) **Overlapping time-slots:** The sampling intervals $t_i = [t_s^i, t_e^i]$ and $t_j = [t_s^j, t_e^j]$ of queries Q_i and Q_j, respectively can exhibit partial overlapping, full overlapping, or no overlapping at all. The effective time-slot $[t_s^c, t_e^c]$ of a common query is identified as shown in Table 3.

Table 3. Cases of time-slot overlapping

Sn.	Condition	Degree of overlapping	$t_i \oplus t_j$	t_s^c	t_e^c
1.	$(t_s^i < t_s^j$ *and* $t_e^i < t_e^j$ *and* $t_s^j < t_e^i)$ or $(t_s^j < t_s^i$ *and* $t_e^j < t_e^i$ *and* $t_s^i < t_e^j)$	*Partial overlapping*	1	$min\{t_s^i, t_s^j\}$	$max\{t_e^i, t_e^j\}$
2.	$(t_e^i \leq t_s^j)$ *or* $(t_e^j \leq t_s^i)$	*Non-overlapping*	0	-	-
3.	$if\ (t_s^i \geq t_s^j$ *and* $t_e^i \leq t_e^j)$ ► *query j overlaps i* or $(t_s^j \geq t_s^i$ *and* $t_e^j \leq t_e^i)$ ► *query i overlaps j*	*Full overlapping*	1	$min\{t_s^i, t_s^j\}$	$min\{t_e^i, t_e^j\}$

Definition 3: *The time-slots t_i and t_j of two queries Q_i and Q_j respectively are said to be partially or fully overlapped if,*

$$t_i \oplus t_j \neq 0$$

where, $\oplus$ represents the overlapping operator that results 0 or 1 in case of non-overlapping and partial/fully overlapping of the time-slots, respectively.

Finally, $Query_p$ generates a set of *common sensing tasks* and *parent sensing tasks* corresponding to each $Q_{cmn} \in Q$ to provide the data corresponding to overlapping and non-overlapping requirements of its parent queries ($\mathbb{Q}$), respectively (see Section 3.3). Likewise, a set of *individual sensing tasks* ($\mathcal{T}_{ind}$) for each individual queries residing in Q is generated. In order to fulfil the QoS requirement, i.e., temporal deadline, of the generated sensing tasks, $Task_a$ allocates them to the appropriate sensor nodes (refer Section 5.2).

Lemma 2: *Pre-processing of similar or partially similar queries reduces generation of total number of sensing tasks.*

Proof: Let there are m number of queries that are pre-processed and as a result k ($< m$) number of common queries ($Q_{cmn}^1, Q_{cmn}^2, Q_{cmn}^3, \ldots\ldots, Q_{cmn}^k$) are created. Suppose, number of distinct parent queries corresponding to each common query are $p_1, p_2, p_3, \ldots\ldots, p_k$, respectively such that $\sum_k p_k \leq m$. Since a common query possesses the overlapping functional requirements as well as non-overlapping requirements of the corresponding parent queries, therefore, total number of sensing tasks generated by a common query Q_{cmn}^k is:

$$|\mathcal{T}^k_{cmn}| = \underbrace{\Big\lfloor |reg_c| \times |atr_c| \Big\rfloor}_{\text{for overlapping requirements}} + \underbrace{\Big\lfloor (|reg_1| \times |atr_1|) - (|reg_c| \times |atr_c|) \Big\rfloor}_{\text{for non-overlapping requirements of } 1^{st} \text{ parent query}} + \underbrace{\Big\lfloor (|reg_2| \times |atr_2|) - (|reg_c| \times |atr_c|) \Big\rfloor}_{\text{for non-overlapping requirements of } 2^{nd} \text{ parent query}} + \ldots\ldots\ldots\ldots + \underbrace{\Big\lfloor (|reg_{p_k}| \times |atr_{p_k}|) - (|reg_c| \times |atr_c|) \Big\rfloor}_{\text{for non-overlapping requirements of } p_k^{th} \text{ parent query}} \tag{9}$$

$$|\mathcal{T}^k_{cmn}| = \underbrace{\Big\lfloor |reg_c| \times |atr_c| \Big\rfloor}_{\text{for overlapping requirements}} + \underbrace{\sum_{j=1}^{p_k} \Big\lfloor (|reg_j| \times |atr_j|) - (|reg_c| \times |atr_c|) \Big\rfloor}_{\text{for non-overlapping requirements of all the corresponding parent queries}} \tag{10}$$

Eq. 10 can be rewritten as:

$$|\mathcal{T}^k_{cmn}| = |\mathcal{T}_c| + \sum_{j=1}^{p_k} (T_j - |\mathcal{T}_c|) = \sum_{j=1}^{p_k} T_j - (p_k - 1)|\mathcal{T}_c| \tag{11}$$

As the number of parent queries associated with a common query increases, it significantly reduces the total number of sensing tasks. In contrast, if all the queries (p_k) were executed individually without pre-processing, the total number of sensing tasks required would be:

$$|\mathcal{T}| = \sum_{j=1}^{p_k} \Big\lfloor |reg_j| \times |atr_j| \Big\rfloor = \sum_{j=1}^{p_k} T_j \tag{12}$$

On comparing Eq. 11 and 12, it can be concluded that the number of sensing tasks generated from common queries is lesser than the number of sensing tasks that would have been generated if the associated parent queries were executed individually.

5.1.1 Time complexity analysis. The algorithm 1 takes m queries as input. The step 3 of this algorithm performs a pairwise calculation of the functional requirement similarity followed by the sorting of different FRS_Q values. The pairwise comparison and sorting takes $O(m^2)$ and $O(mlogm)$ time respectively. However, $Query_p(\mathbb{Q})$ is an iterative process that is repeated every time a common query (Q_{cmn}) is created and inserted into $\mathbb{Q}$. The process terminates when no common query is created. Thus, the overall time complexity of Algorithm 1 can be estimated as $O(k(m^2 + mlogm))$, where $k << m$ is the number of iterations before the process terminates.

5.2 Task Allocation Module ($Task_a$)

After solving the ***min-RST*** problem, we move forward to solve the problem of task allocation. The tasks generated by $Query_p$ are handled by $Task_a$ to allocate them on sensor nodes satisfying their functional and QoS requirements. With the help of SDD, $Task_a$ identifies relevant sensor

nodes complying with the tasks' functional requirements (refer Eq 5). To ensure the fulfilment of QoS requirements (temporal deadline) of sensing tasks, $Task_a$ estimates end-to-end delay in transmitting sampled data from relevant sensor nodes to the gateway. End-to-end delay is estimated for all the relevant sensor nodes concerning a sensing task to identify an appropriate node among them providing data within temporal deadline of the task.

In sensor networks, end-to-end delay depends on: (i) the average waiting time a packet spends in the queue (ρ_q) on a sensor node, (ii) transmission time (T_χ), and (iii) number of intermediate nodes from source node to the gateway. A queuing model of type $(M/M/1) : (GD/L/\infty)$ [19] is considered to estimate the average waiting time (see Eq. 13) in the queue where sensor node acts as a server with the queue size of L. The packet arrival rate λ follows Poisson distribution and service time μ is exponentially distributed.

$$\rho_q = \frac{1-(\lambda/\mu)^L - (1-\lambda/\mu)(L(\lambda/\mu)^L + 1 - (\lambda/\mu)^{L+1})}{\mu(1-\lambda/\mu)(1-(\lambda/\mu)^{L+1})} \tag{13}$$

For a packet, T_χ depends on different factors, such as number of re-transmissions (η_r), wake time of the receiver (T_ψ), sleep time of a sensor node (T_s), and back-off period (T_β) at the MAC layer. Hence, T_χ can be represented using Eq. 14.

$$\mathrm{T}_\chi = \eta_\mathrm{r}(\mathrm{T}_\psi + \mathrm{T}_\mathrm{s}) + E[\mathrm{T}_\beta] + R(0, \mathrm{T}_\mathrm{s}) \tag{14}$$

where, T_ψ and T_s are MAC layer parameters. $R(0, \mathrm{T}_\mathrm{s})$ is a random number distributed between 0 and T_s for the case when the receiver is sleeping and the sender has to wait until the receiver wakes up [1]. The average number of re-transmissions (η_r) can be modeled as Geometric distribution and calculated as $\frac{1-p}{p}$, where p is the probability of successful transmission. According to the exponential back-off mechanism in CSMA/CA, number of slots (η_s) that has been waited by a sensor node after detecting the collision is $2^c - 1$, where c ($\approx \eta_\mathrm{r}$) is the number of collisions occurred so far for that sensor node. The expected back-off time for a sensor node can be calculated as

$$E[\mathrm{T}_\beta] = \frac{1}{\eta_\mathrm{s}+1}\sum_{j=1}^{\eta_\mathrm{s}} j \Rightarrow \frac{\eta_\mathrm{s}(\eta_\mathrm{s}+1)}{2(\eta_\mathrm{s}+1)} \Rightarrow \frac{\eta_\mathrm{s}}{2} \Rightarrow \frac{2^{\frac{1-p}{p}}-1}{2} \tag{15}$$

Hence, Eq. 14 can be re-written as

$$\mathrm{T}_\chi = \frac{1-p}{p}(\mathrm{T}_\psi + \mathrm{T}_\mathrm{s}) + \frac{2^{\frac{1-p}{p}}-1}{2} + R(0, \mathrm{T}_\mathrm{s}) \tag{16}$$

By using Eq. 13 and 16, average 1-hop delay in transmitting a packet from a sensor node can be characterized by Eq. 17 whereas Eq. 18 represents end-to-end delay incurred in transmitting a data packet from source node to the gateway.

$$\begin{aligned} delay_{avg} &= \rho_q + \mathrm{T}_\chi \\ &= \rho_q + \frac{1-p}{p}(\mathrm{T}_\psi + \mathrm{T}_\mathrm{s}) + \frac{2^{\frac{1-p}{p}}-1}{2} + R(0, \mathrm{T}_\mathrm{s}) \end{aligned} \tag{17}$$

$$delay_{e2e} = \sum_{z=1}^{d} delay_{avg}^z \tag{18}$$

where, d is the number of intermediate nodes between source node, where sensing task is to be allocated, and the gateway. A relevant sensor node, as per Eq. 5, is considered for allocating the sensing task if $delay_{e2e}^n \leq R_{T_{ij}^k}^{temp}$.

Algorithm 2 Task Allocation

```
procedure Task_a(T)
    for each task x ∈ T do
        Identify relevant sensor nodes (N' ⊆ N) satisfying functional requirements of x. (refer Eq. 5)
        for each n ∈ N' do
            Estimate end-to-end delay between node n and gateway.
            if delay^n_e2e ≤ R^temp_x then
                Allocate sensing task x to node n.
            end if
        end for
    end for
end procedure
```

The sensor nodes to which sensing tasks are allocated transmit sensed data to the gateway within the specified time-slot at a regular interval. The service provisioning process at the gateway collaborates sensed data received from different tasked sensor nodes and provide appropriate responses to the queries. The proposed task allocation mechanism (Algorithm 2) is based on minimizing the data delivery time so that a task does not miss its temporal deadline. The data delivery time complexity analysis is as follows.

5.2.1 Data delivery time complexity. It is defined as the total time taken by the data transmission mechanism in delivering all the packets to the gateway from tasked sensor nodes. If there are total $\mathcal{P}$ packets to be transmitted by $\mathcal{N}$ sensor nodes in the network, any data transmission algorithm needs minimum $\mathcal{P}$ time slots [16].

Theorem: *Given a network of $\mathcal{N}$ sensor nodes, the data delivery time complexity of QueryPM is* $\Theta(\mathcal{P})$.

Proof: For $\mathcal{N}$ sensor nodes $(s_1, s_2, s_3, ..., s_{\mathcal{N}})$ in the network, if $\mathcal{P}_k$ is the number of packets to be transmitted by node s_k corresponding to a task, then the overall data delivery time can be estimated as

$$\sum_{k=1}^{\mathcal{N}} \mathcal{P}_k \Big(\sum_{j=1}^{m+1} delay_{avg} + T_{sense} \Big) \leq \sum_{k=1}^{\mathcal{N}} \mathcal{P}_k \Big(\sum_{j=1}^{Depth} delay_{avg} + T_{sense} \Big) \tag{19}$$

where, m is the number of intermediate sensor nodes, T_{sense} is the sensing time and $Depth$ is calculated for the communication tree (Section 3.4). Since $delay_{avg}$, T_{sense} and $Depth$ are constants, the above equation can be re-written as

$= \Theta(\sum_{k=1}^{\mathcal{N}} \mathcal{P}_k)$

$= \Theta(\mathcal{P})$

This finishes the proof.

5.3 Service Provisioning

The Service provisioning process is an important part of the *QueryPM* framework. *Service* can be defined as a piece of information which is acquired by collaborating the data from certain set of physical attributes relevant to the application queries [10]. As a result of $Query_p$, a set of common queries and individual queries are created which are further decomposed into the corresponding set of sensing tasks. In *QueryPM*, sensing tasks corresponding to the common queries and individual

queries are allocated to the appropriate sensor nodes. The data sensed by the tasked sensor nodes is provided to the service provisioning process at the gateway where it is shared and collaborated to serve the application queries.

As mentioned earlier, $\mathcal{T}_{cmn} = \{T_{ij} | 1 \leq i \leq |reg_o| \text{ and } 1 \leq j \leq |atr_c|\} \cup \{T_p\}$ contains the sensing tasks which are required to fulfill the overlapping as well as non-overlapping requirements of their parent queries (see Section 3.3). The data sensed by the *common sensing tasks* (first term of $\mathcal{T}_{cmn}$) of a common query is desired by all of its parent queries, hence need to be shared among them. This shared data is further collaborated with the data sensed by the task set $\{T_p\}$ (second term of $\mathcal{T}_{cmn}$) corresponding to the non-overlapping requirements of the parent queries to provide them appropriate services. In addition, individual queries are served by collaborating the sensed data provided by their corresponding set of sensing tasks ($\mathcal{T}_{ind}$) only.

6 IMPLEMENTATION AND EVALUATION

The *QueryPM* framework and its components, i.e., $Query_p$ and $Task_a$, are implemented on NS-2 and simulations are conducted to study its influence on the constraint network resources. The consequences are analyzed in terms of amount of downstream traffic (total number of sensing tasks), upstream traffic, energy consumption, and network lifetime. Since proper allocation of sensing tasks is most important for the successful execution of the queries, therefore, *QueryPM* with the underlying end-to-end delay estimation model is evaluated in terms of percentage of queries meeting their temporal deadlines.

The proposed framework is compared with the CAQO [29] and CATS [28]. CAQO reduces the query requests from concurrent applications and respond them through cooperative caching mechanism. CAQO forecasts the query requests to prefetch and cache the required sensory data at sink node to respond the forthcoming queries. CATS [28] does *not* pre-process the queries and disseminate all the sensing tasks corresponding to each query into the network and provide sampled data from tasked sensor nodes. However, CATS takes into account the time-slot overlapping among sensing tasks allocated on a sensor node to reduce the amount of data sampling.Table 4 lists the parameter settings for the simulations. 200 sensor nodes are uniformly deployed in the monitoring area which is divided into 12 grid cells, and four different environmental attributes ($|\mathcal{A}| = 4$) can be sought from each grid cell.

Table 4. Parameter settings in simulations

Parameter Name	**Value**
Monitored area	$1750 \times 700\ meter^2$
Grids in sensor field	3×4
Number of sensor nodes	200
Number of attributes ($\mathcal{A}$)	4
Communication radius	50 *meters*
Number of applications (ρ)	5, 10, 15
Number of queries in the pool $\mathcal{M}$	500 - 1500
Number of queries in a decision-window (m)	10 - 50
Transmission power ($txPower$)	35.28e-3 W
Receiving power ($rxPower$)	31.32e-3 W
$sleepPower$	144e-9 W
$idlePower$	712e-6 W
Initial energy	10 *Joules*

In a realistic scenario, a large number of queries from various applications arrives in a decision-window (dw) and each query can have different functional requirements. Therefore, to simulate such scenarios, a population of $\mathcal{M}$ ($= \sum_{i=1}^{\rho} n_i$) queries from ρ distinct applications is created that contains equal number of queries $n_1 = n_2 = n_3 = ... = n_\rho$ for each application. Different samples of queries (m) are drawn from this population in each dw according to the Eq. 20.

$$\sum_{i=1}^{\rho} w_i n_i = m \leq \mathcal{M} \tag{20}$$

$$\text{where,} \qquad \sum_{i=1}^{\rho} w_i = 1$$

Each sample consists of queries from ρ different applications in different proportion. A random weight factor, $w_i \in [0, 1]$ such that $0 \leq (n_i w_i \in \mathbb{Z}^+) \leq m$, is assigned to each application to vary the proportion of the queries from that application in a sample. It ensures that the set of queries arriving in a decision-window will be different from the previous decision-window(s).

6.1 Impact on downstream traffic

In SSN, the downstream traffic is caused due to sensing tasks dissemination. Therefore, the reduction in number of sensing tasks will reduce the downstream traffic. To study the impact of *QueryPM* on the volume of sensing tasks (downstream traffic) in a decision-window, we took a population ($\mathcal{M}$) of 1000 queries for $\rho = 10$ applications. The query population includes $n_i = 100$ queries from each of the 10 applications and have varying functional requirements. We analyze the percentage reduction in sensing tasks based on the simulation results, considering the number of queries (m) 10, 30, and 50 taken from $\mathcal{M}$ according to Eq. 20.

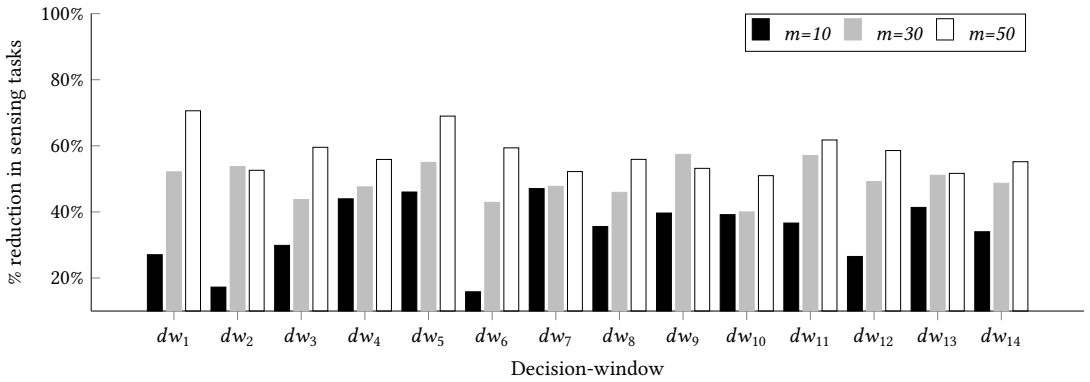
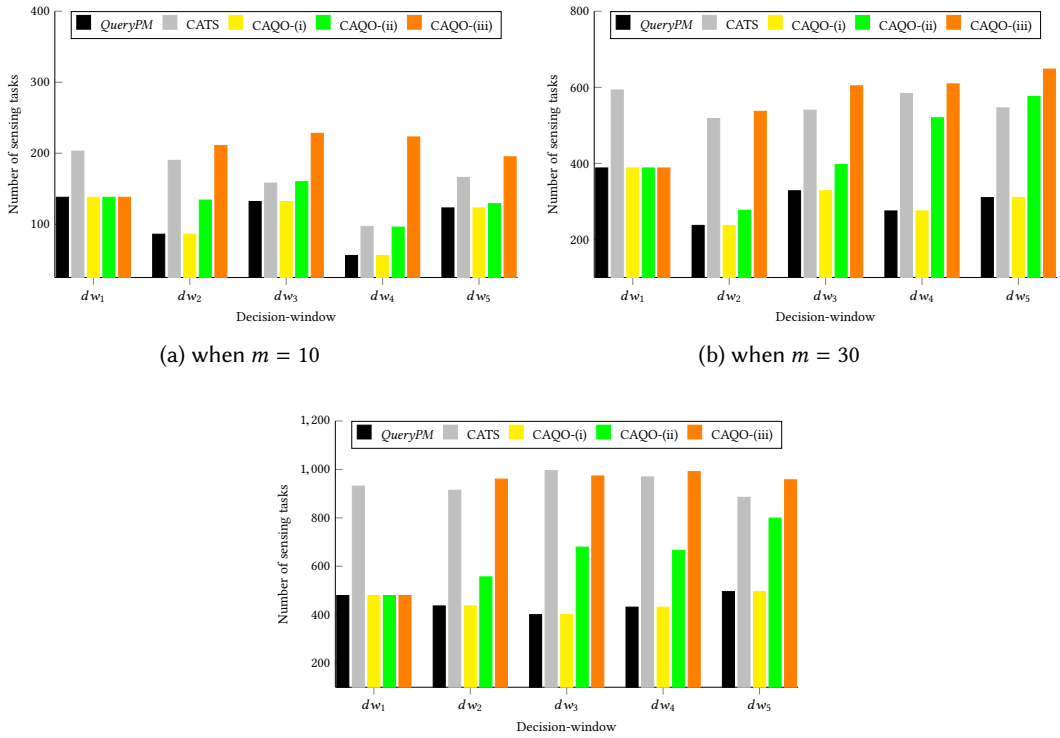


Fig. 4. Percentage reduction in sensing tasks in different decision-windows, when $\rho = 10$

Fig. 4 shows percentage reduction in number of sensing tasks in different decision-windows ($dw_1, ..., dw_{14}$) when $Query_p$ pre-processed the queries. A consistent pattern in the task reduction is not observed for any of the scenarios (when $m = 10,\ 30,$ and 50) over different decision-windows. This is due to the variations in queries' requirements in different decision-windows that results in varying number of sensing tasks for the same number of concurrent queries. However, the percentage reduction in sensing tasks is high for the cases when $m = 30$ and $m = 50$ as compared to $m = 10$. This reduction is due to large number of queries with the overlapping requirements in the samples. This percentage reduction in sensing tasks is relative to the number of sensing tasks

disseminating into the network in CATS [28]. CATS focuses on eliminating the communication redundancy in the upstream traffic by performing on-node data sharing among sensing tasks and significantly reduces the upstream traffic. However, the downstream traffic due to the sensing tasks is still a concern with the view point of network resources. In contrast, $Query_p$ deployed at the gateway reduces the downstream traffic upto 60%.

Further, we compared the efficiency of *QueryPM* with CAQO [29] in which sensory data required by future queries are prefetched and cached at the sink node. However, the relevant sensor nodes have to be tasked first during the prefetching process which contribute in both downstream and upstream traffic in the network. In this regard, we considered three scenarios, namely, CAQO-(i): all the queries are successfully responded by providing the prefetched cache data at the gateway, CAQO-(ii): half number of queries in a decision-window are answered by cached data and remaining queries are answered by fetching the data in real-time due to cache misses at the gateway, and CAQO-(iii): queries arriving in the subsequent decision-windows are different and the prefetched data do not satisfy the query requests. Thus, data are fetched in real-time by disseminating the sensing tasks again corresponding to the arrived queries. Fig. 5 shows a comparison among *QueryPM*, CATS and the above mentioned scenarios of CAQO in terms of downstream traffic, i.e., number of sensing tasks disseminated into the network.



(a) when $m = 10$

(b) when $m = 30$

(c) when $m = 50$

Fig. 5. Comparative analysis of downstream traffic in different decision-windows, when $\rho = 10$

Fig 5 depicts that the downstream traffic in cases of CAQO-(i) and CAQO-(ii) are less as compared to CATS. The reduction is due to the underlying query processing mechanism that eliminates

the reprocessing of shared subquery requests and avoid redundant sensing tasks concerning the queries to enter into the network. *QueryPM* outperforms in comparison to all the scenarios of CAQO, however CAQO-(i) results in the same amount of downstream traffic. It is due to the underlying query reduction mechanism and the assumption that all the forthcoming query requests are answered through prefetched data. Thus, the resulting sensing tasks in CAQO-(i) are concerning to the data prefetching for the upcoming query requests. On the other hand, in a realistic scenario, i.e., CAQO-(iii), queries arriving in subsequent decision-windows are random, thus prefetched data do not satisfy the actual query requests. Henceforth, the sensing tasks are transmitted again to fetch the required data in real-time due to cache misses, in addition to the tasks that have been disseminated earlier to prefetch the data. Thus, the resulted downstream traffic in CAQO-(iii) is approximately twice of CAQO-(i). The downstream traffic in case of *QueryPM* for the very first decision-window is nearly equal to all the scenarios of CAQO since there were no prefetching happened initially and the queries are responded by disseminating the sensing tasks in real-time. The impact on downstream traffic in these scenarios also reflects on upstream traffic as well as on network energy consumption.

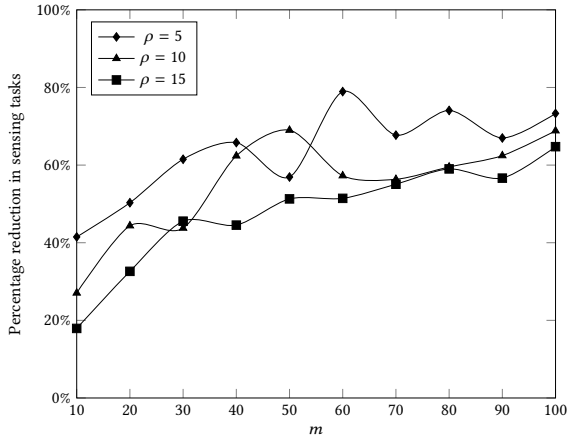
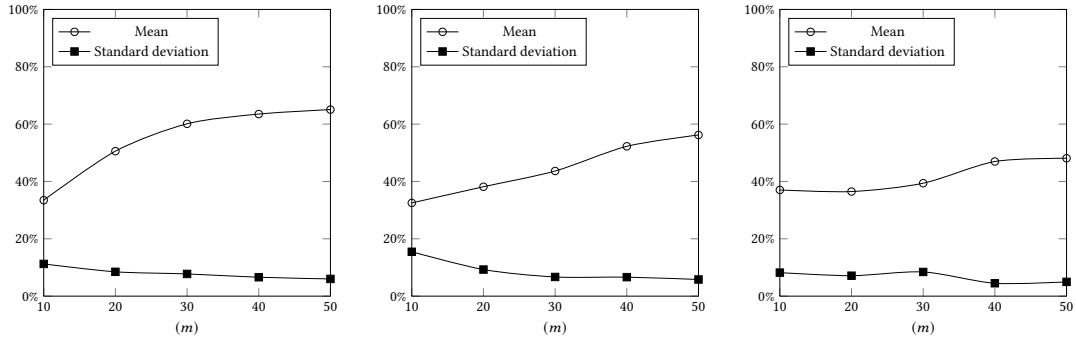


Fig. 6. Percentage reduction in sensing tasks when ρ and m vary

In realistic scenarios, number of applications in SSN can vary and rate of arrival of queries can be high in a decision window. To accommodate such scenarios in simulation and to test the scalability aspect of the proposed framework, three different scenarios are considered with varying number of applications ($\rho = 5, 10, 15$). In each of these scenarios, m is varied from 10 to 100. Fig. 6 depicts the percentage reduction in downstream traffic when queries are pre-processed by $Query_p$. Simulations are conducted for several times to study the variations in the downstream traffic. The mean and standard deviation of the percentage reductions in the sensing tasks are calculated from the outcomes of these simulations (see Fig. 7). However, we have shown the results corresponding to the 20 iterations since divergence in mean values and standard deviations was marginal after that. Increment in the mean along with negative inclination in the standard deviation with respect to m suggests that the reduction in sensing tasks is not varying as much as it varies for the smaller value of m in a decision-window. It shows that *QueryPM* is able to process large number of queries arriving in a decision-windows and avoids the network to get overwhelmed with high volume of sensing tasks. It makes *QueryPM* robust and suitable for real scenarios in large-scale deployments. It is also to be noted that proportion of individual and common queries resulting after pre-processing



(a) when $\rho = 5$ (b) when $\rho = 10$ (c) when $\rho = 15$

Fig. 7. Statistical analysis of percentage reduction in sensing task for different values of ρ and m

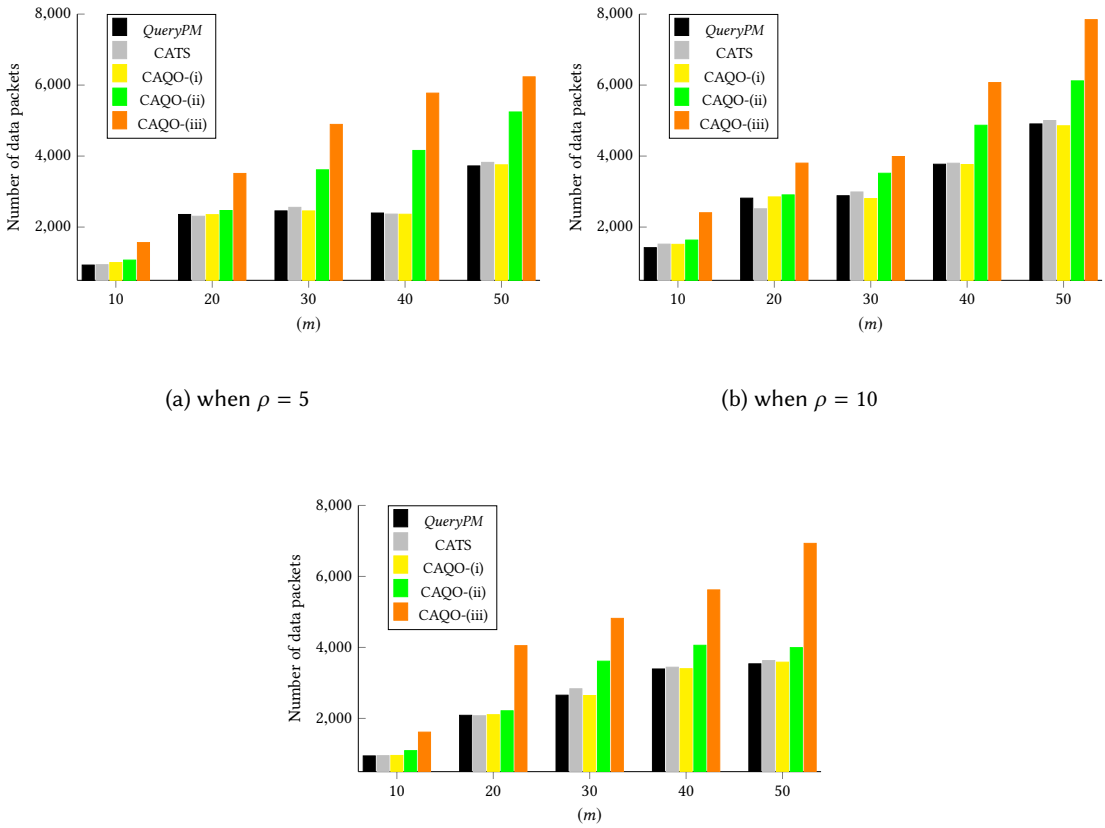
of the application queries depends upon the similarity in their functional requirements. If large number of queries in a decision-window has similar functional requirements, their pre-processing will result in less number of individual queries. We observe the average percentage of individual queries in a decision window which is found as ≈39%, ≈43%, and ≈ 42% for ρ = 5, 10, and 15 respectively.

6.2 Impact on upstream traffic

The effect of *QueryPM* on upstream traffic (number of data packets transmitted by tasked sensor nodes) is observed in the same experimental settings as used in Section 6.1. Fig. 8 illustrates the number of data packets sent from the tasked sensor nodes towards the gateway in the proposed framework, CATS, and all the three considered scenarios of CAQO (as mentioned in Section 6.1) for different number of queries arriving in a decision-window. Since CATS schedule the sampling interval into the overlapping time-window of the sensing tasks allocated on a sensor node and samples the data for that duration only. Therefore, it would result in lesser upstream traffic than the *QueryPM*. However, *QueryPM* considers the continuous sampling during the specified time-slot of the sensing tasks (see Section 1). Therefore, CATS as well as CAQO are evaluated with the assumption of continuous sampling over the specified time-slots of the sensing tasks. CAQO performs query reduction at the gateway to avoid redundant execution of similar sub-queries. This is the reason CAQO-(i) results in nearly same amount of upstream traffic as *QueryPM* and CATS. It is evident from Fig. 5 that CAQO-(iii) results in high downstream traffic due to transmission of sensing tasks twice, which significantly increases the upstream traffic as well. Similarly, CAQO-(ii) results in comparatively high upstream traffic in comparison to CATS and *QueryPM* since half of the queries are responded through cached data and remaining queries have to fetch the data in real-time due to cache misses at the gateway. Henceforth, the data transmission happens twice, first, at the time of data prefetching, and second, at the time of cache misses at the gateway. Furthermore, the variation in upstream traffic for same value of m across different value of ρ (Fig. 8 (a), (b), and (c)) is due to the diverse functional requirements of the queries in the decision-windows.

6.3 Overall network energy consumption

Proposed query preprocesisng mechanism is tested for the energy consumption incurred due to downstream as well as upstream traffic. Fig. 9 shows the comparison of total network-wide energy consumption for *QueryPM*, CATS, and CAQO (scenarios discussed in Section 6.1) for varying



(a) when $\rho = 5$

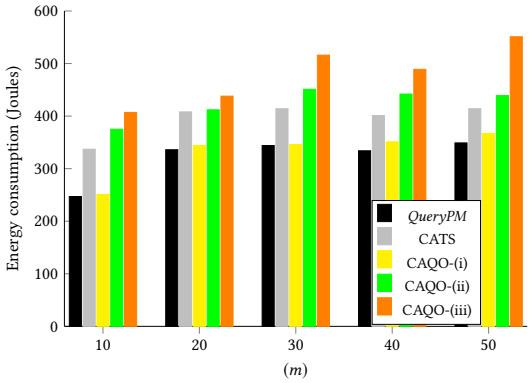
(b) when $\rho = 10$

(c) when $\rho = 15$

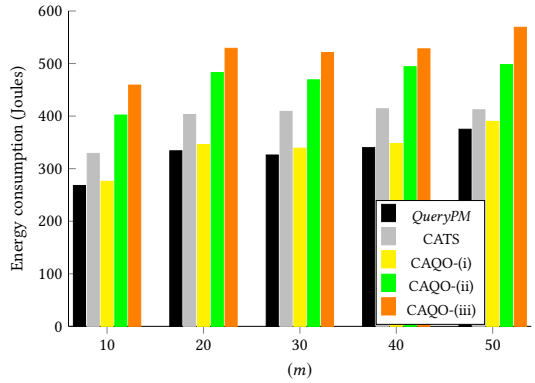
Fig. 8. Comparison of total number of data packets transmitted by the tasked sensor nodes in *QueryPM*, CATS, and CAQO

number of queries arriving in a decision-window from different number of applications. The overall network energy consumption involves energy consumption in task dissemination as well as energy consumption in data transmission from/to tasked sensor nodes and the gateway.

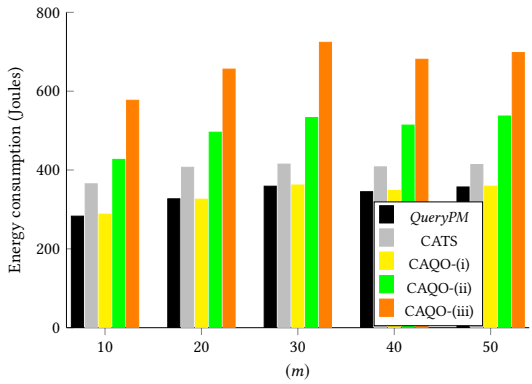
Since *QueryPM* exploits the overlapping requirements among the application queries and reduces the number of sensing tasks in the network, it directly benefits in conserving the energy consumption. Furthermore, reduction in data packet transmissions from sensor field to the gateway in *QueryPM* (as shown in Fig. 8) has direct influence in reduction of overall energy consumption. On the other hand, CATS results in relatively high energy consumption since it disseminates all the sensing tasks into the network without preprocessing. However, CATS conserves the energy consumption in upstream traffic by exploiting the interval data sampling and data sharing like *QueryPM*, hence, both the mechanisms are more energy-efficient in comparison to CAQO-(ii) and CAQO-(iii). On the other hand, CAQO-(i) results in nearly same energy consumption as *QueryPM* and CATS as a consequence of low downstream and upstream traffic as evident in Fig. 5 and 8, respectively. The reason behind high energy consumption in CAQO-(ii) and CAQO-(iii) is the



(a) when $\rho = 5$



(b) when $\rho = 10$



(c) when $\rho = 15$

Fig. 9. Network-wide energy consumption incurred in dissemination of downstream (sensing tasks) as well as upstream traffic in *QueryPM*, CATS, and CAQO

overhead of downstream and upstream traffic caused due to cache misses at the gateway. In CAQO-(iii), the transmission of sensing tasks and sensed data happens twice for all the queries, whereas in CAQO-(ii), it happens only for half of the queries. Therefore, CAQO-(iii) has higher energy consumption than its counterparts. The peripheral effects of high energy consumption reciprocate to the network lifetime as well. To avoid the uncertainty of the results, simulations are conducted 20 times, and the average of their results is depicted in Fig. 9.

6.4 Impact on network lifetime

Generally, network lifetime is defined as the time duration until the first node drained of its energy and can be characterized by Eq. 21.

$$lifetime = \min_{n \in N} LT_n \tag{21}$$

where LT_n is the lifetime of a node n.

However, this definition of network lifetime is not adequate since *first dead node (FDN)* does not necessarily leads towards the network partitioning or severe connectivity disruption [6]. In WSN, network operations are of prime concern and the network is said to be alive till it is providing the services without interruption. Therefore, we argue to identify the lifetime of *critical nodes (CN)* in the network to define network lifetime since their death majorly affects the network operations. In the network of uniformly distributed sensor nodes, a set of critical nodes and network lifetime are characterized by Eq. 22 and 23, respectively.

$$\mathbb{CN} = \{n \in \mathcal{N} | d_n \; is \; maximum\} \tag{22}$$

where d_n is the degree of node n.

$$network \; lifetime = \min_{n \in \mathbb{CN}} LT_n \tag{23}$$

The network is considered as a connected graph $G = (\mathcal{V}, \mathcal{E})$. Let $v \in \mathcal{V}$ be a node in the network and its neighbors can be respresnted as,

$$Nb(v) = \{u \in \mathcal{V} | vu \in G\}$$

where, an edge vu exists between node v and u if $dis(v, u) < R$, where R is the communication range of v. Therefore, the degree of node v is $d_v = |Nb(v)|$. In a connected graph, failure of a node implies the deletion of all the edges incident on that node from its neighbors. Two arbitrary nodes are said to be connected if there exists a path consisting of consecutive edges between them. Let the sequence $P_{1,k+1} : e_1 e_2 .... e_k$ is a path of length k from node u_1 to u_{k+1}. It can also be represented as $P_{1,k+1} : u_1 \rightarrow u_2 \rightarrow .... \rightarrow u_k \rightarrow u_{k+1}$. In general, the shortest path between two arbitrary nodes u_i and u_j can be represented as $P_{i,j} : u_i \overset{*}{\rightarrow} u_j$, such that $u_i \neq u_j$, for all $i \neq j$.

Considering gateway ($\mathcal{G}$) as a destination in sensor network, total number of source-destination pairs will be N, where N is the number of sensor nodes. $\mathcal{P} = \{P_{i,\mathcal{G}} | i = 1, 2, 3 .... N\}$ represents a set of shortest path for all the pairs such that $|\mathcal{P}| = N$. Let the degree of CN is d_{cn}, i.e., $|Nb(cn)| = d_{cn}$. Let $\mathcal{P}' \subset \mathcal{P}$ is a set of paths such that $\exists u (\in Nb(cn)) \in \mathcal{P}'$ and $\mathcal{P}'' \subset \mathcal{P}$ represents a set of paths such that $\exists u (\in Nb(fdn)) \in \mathcal{P}''$, and $\mathcal{P}' \cup \mathcal{P}'' \neq \mathcal{P}$. If FDN is not the critical node, i.e., $d_{fdn} < d_{cn}$, that means $\mathcal{P}'' < \mathcal{P}'$. Therefore, the failure of CN is more prune for connectivity disruption of the network.

We measured the elapsed time to the death of CN and FDN to compare the network lifetime for *QueryPM*, CATS, and CAQO (see Fig. 10). The network lifetime in all of three mechanisms are a direct consequence of network-wide energy consumption as shown in Fig. 9. Packet delivery ratio (PDR) at the gateway is considered as a measure of connectivity and the corresponding observations are depicted in Fig. 11. The failure of first node is not necessarily due to its frequent involvement in routing. Instead, comparatively high demand by the queries nearby its region and corresponding data transmissions can also cause its failure. Previous results show the significant reduction in downstream traffic, upstream traffic, and overall network energy consumption in *QueryPM* that relaxes the burden on *CN* but demand of *FDN* can not be controlled, which causes its early death.

The elapsed time of *FDN* is 55 minutes in *QueryPM* whereas *CN* lasts for 8 minutes longer (see Fig. 10). The peripheral effects on PDR at these timestamps are clearly visible in Fig. 11. The PDR drops with a huge margin at the instance of *CN*'s death as compared to that of *FDN*. It is due to the fact that failure of *CN* results in disconnection of large number of neighboring nodes (as *CN* has maximum degree). Furthermore, its failure results in disconnection of several source nodes to the gateway, hence, results in reduced PDR. The variations in PDR throughout the simulation is an

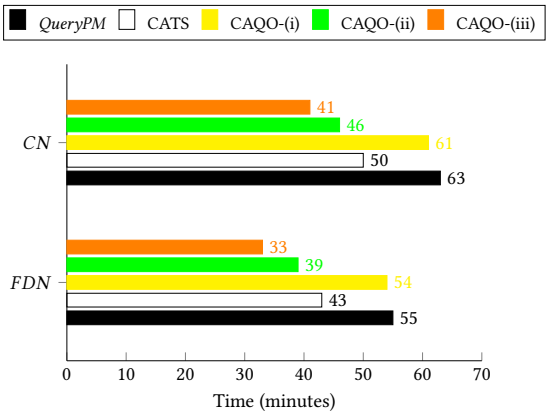


Fig. 10. Network lifetime

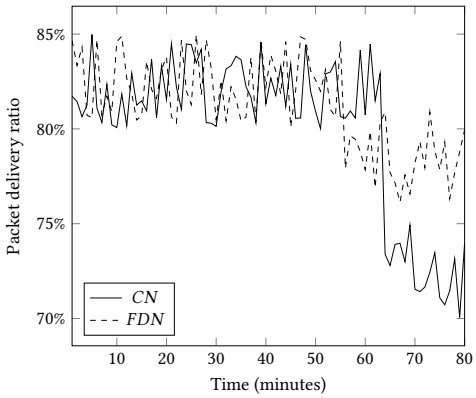
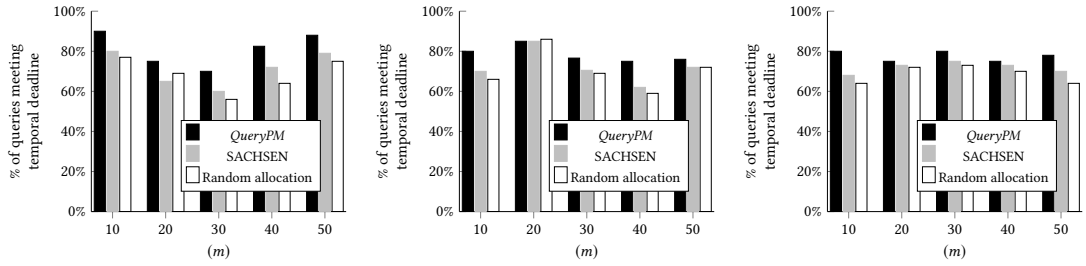


Fig. 11. Comparison of packet delivery ratio in case of the death of first node and critical node

effect of varying requirements of different queries and varying network characteristics at different time instances that results in varying amount of data transmissions.

6.5 Impact on application-level QoS parameter

Temporal deadline of the queries is a critical QoS aspect in sensor networks. Providing the sensed data from all the relevant sensing tasks corresponding to the queries to the gateway within their temporal deadlines is very crucial for their successful execution. Therefore, we observed the percentage of queries meeting their temporal deadlines in the light of proposed $Task_a$ module of the *QueryPM*, which estimates the end-to-end delay before allocating the sensing tasks to meet their QoS requirements (temporal deadlines). We chose two task allocation schemes, SACHSEN proposed in [15] and random approach as reference to study the performance of $Task_a$ module of proposed *QueryPM* framework. SACHSEN is a QoS aware task allocation scheme considering energy and data accuracy as the QoS requirements while random allocation scheme is a non-QoS-aware approach which does not consider the QoS requirements of the tasks and assigns a sensor node randomly from a list of candidate sensor nodes. However, SACHSEN selects the candidate sensor nodes for a task based on their relevancy to the task which is similar to our proposed scheme. SACHSEN does not study the effects of task allocation mechanism on temporal deadlines of the sensing tasks.



(a) when $\rho = 5$ (b) when $\rho = 10$ (c) when $\rho = 15$

Fig. 12. Comparison of *QueryPM*, SACHSEN, and Random approach of allocating sensing tasks in terms of percentage of queries meeting their temporal requirements

Since we are considering temporal deadline as a QoS parameter for application queries in this paper, it becomes the crucial parameter to study. Fig. 12 depicts the percentage of successfully executed queries when $Query_p$ is integrated with $Task_a$ and compared with SACHSEN and random allocation of sensing tasks, which does not consider the temporal or delay constraint of the sensing tasks before their allocation. However, the tasks are allocated on such sensor nodes that can fulfill their functional requirements. As depicted in the observations, *QueryPM* and SACHSEN performs better than the random allocation of sensing tasks in each scenario. The reason behind SACHSEN outperforming random allocation scheme is due to the consideration of energy as the QoS parameter in the application queries. It leads SACHSEN to select the candidate sensor nodes having minimum hop distance to the gateway. Therefore, it fulfils the temporal deadline of relatively large number of queries as compared to the random allocation scheme. However, SACHSEN does not work better than the *QueryPM* because considering minimum energy path does not always ensure less end-to-end delay. Since $Task_a$ estimates end-to-end delay from candidate sensor nodes before allocating the sensing tasks, *QueryPM* shows significant improvement in meeting temporal deadline of the queries as compared to both the mechanisms. However, the variations in the percentage of successfully executed queries are due to the random nature of their functional requirements.

7 CONCLUSION AND FUTURE WORK

This paper proposes a query processing framework (QueryPM) to efficiently process the queries and allocate the generated sensing tasks to the relevant sensor nodes. *QueryPM* comprises of query pre-processing module ($Query_p$) and task allocation module ($Task_a$) to solve the problem of redundant sensing tasks generation and their proper allocation, respectively. $Query_p$ creates common queries on the basis of functional requirement similarity of the queries, which results in non-redundant sensing tasks and significantly reduces the downstream traffic. $Task_a$ allocates the generated sensing tasks to the appropriate sensor nodes by estimating the end-to-end delay to ensure the fulfilment of temporal deadlines of sensing tasks. To validate our problems and corresponding solution approaches, extensive simulations are conducted which show that proposed framework reduces the network traffic and energy consumption significantly, and improves the network operations even if the number of concurrent query requests are relatively large.

Sensed data corresponding to the sensing attributes interested by queries are provided through the gateway. Further processing and analytics on the sensed data can be performed for providing the sophisticated services, such as high-level contexts [18][24], situation awareness [13], activity recognition, etc., to other WSN based IoT applications. For instance, context-aware applications seek high-level contexts (HLCs) instead of raw sensor data from the sensor network. In order to infer the HLCs, data from different sensors are required to be collaborated and processed either at the gateway or at a node inside the network. Considering resource-constrained SSN, the location of HLCs inference (out-network or in-network) has wide impact on the network resources and accuracy of the inferred context. We leave this aspect on top of the *QueryPM* framework as an open research challenge for the readers.

REFERENCES

[1] Boris Bellalta, Azadeh Faridi, Dirk Staehle, Jaume Barcelo, Alexey Vinel, and Miquel Oliver. 2013. Performance analysis of CSMA/CA protocols with multi-packet transmission. *Computer Networks* 57, 14 (2013), 2675–2688.

[2] Markus Bestehorn, Zinaida Benenson, Erik Buchmann, Marek Jawurek, Klemens Böhm, and Felix C Freiling. 2010. Query Dissemination in Sensor Networks-Predicting Reachability and Energy Consumption. *Ad Hoc & Sensor Wireless Networks* 9, 1-2 (2010), 85–107.

[3] Sourabh Bharti and Kiran Kumar Pattanaik. 2016. Task requirement aware pre-processing and Scheduling for IoT sensory environments. *Ad Hoc Networks* 50 (2016), 102–114. https://doi.org/10.1016/j.adhoc.2016.07.005

[4] Md Zakirul Alam Bhuiyan, Guojun Wang, and Athanasios V Vasilakos. 2014. Local area prediction-based mobile target tracking in wireless sensor networks. *IEEE Trans. Comput.* 64, 7 (2014), 1968–1982.

[5] Carmen Delgado, Sergio Batista, María Canales, José Ramón Gállego, Jorge Ortín, and Matteo Cesana. 2018. An Implementation for Dynamic Application Allocation in Shared Sensor Networks. In *2018 11th IFIP Wireless and Mobile Networking Conference (WMNC)*. IEEE, 1–8.

[6] Isabel Dietrich and Falko Dressler. 2009. On the lifetime of wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN)* 5, 1 (2009), 5. https://doi.org/10.1145/1464420.1464425

[7] Xiaolin Fang, Hong Gao, Jianzhong Li, and Yingshu Li. 2013. Application-aware data collection in Wireless Sensor Networks. In *INFOCOM, 2013 Proceedings IEEE*. IEEE, 1645–1653. https://doi.org/10.1109/INFCOM.2013.6566961

[8] Claudio M De Farias, Wei Li, Flávia C Delicato, Luci Pirmez, Albert Y Zomaya, Paulo F Pires, and José N De Souza. 2016. A systematic review of shared sensor networks. *ACM Computing Surveys (CSUR)* 48, 4 (2016), 51. https://doi.org/10.1145/2851510

[9] Hong Gao, Xiaolin Fang, Jianzhong Li, and Yingshu Li. 2015. Data collection in multi-application sharing wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems* 26, 2 (2015), 403–412. https://doi.org/10.1109/TPDS.2013.289

[10] Sahin Cem Geyik, Boleslaw K Szymanski, and Petros Zerfos. 2013. Robust dynamic service composition in sensor networks. *IEEE Transactions on Services Computing* 6, 4 (2013), 560–572. https://doi.org/10.1109/TSC.2012.26

[11] William I Grosky, Aman Kansal, Suman Nath, Jie Liu, and Feng Zhao. 2007. Senseweb: An infrastructure for shared sensing. *IEEE multimedia* 14, 4 (2007), 8–13. https://doi.org/10.1109/MMUL.2007.82

[12] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. 2000. Energy-efficient communication protocol for wireless microsensor networks. In *System sciences, 2000. Proceedings of the 33rd annual Hawaii international conference on*. IEEE, 10–pp. https://doi.org/10.1109/HICSS.2000.926982

[13] Cory Henson, Amit Sheth, and Krishnaprasad Thirunarayan. 2012. Semantic perception: Converting sensory observations to abstractions. *IEEE Internet Computing* 16, 2 (2012), 26–34. https://doi.org/10.1109/MIC.2012.20

[14] Ilias Leontiadis, Christos Efstratiou, Cecilia Mascolo, and Jon Crowcroft. 2012. SenShare: transforming sensor networks into multi-application sensing infrastructures. In *European Conference on Wireless Sensor Networks*. Springer, 65–81. https://doi.org/10.1007/978-3-642-28169-3_5

[15] Wei Li, Flávia C Delicato, Paulo F Pires, Young Choon Lee, Albert Y Zomaya, Claudio Miceli, and Luci Pirmez. 2014. Efficient allocation of resources in multiple heterogeneous wireless sensor networks. *J. Parallel and Distrib. Comput.* 74, 1 (2014), 1775–1788.

[16] Xiang-Yang Li, Yajun Wang, and Yu Wang. 2010. Complexity of data collection, aggregation, and selection for wireless sensor networks. *IEEE Transactions on computers* 60, 3 (2010), 386–399.

[17] Mihaela Mitici, Martijn Onderwater, Maurits de Graaf, Jan-Kees van Ommeren, Nico van Dijk, Jasper Goseling, and Richard J Boucherie. 2015. Optimal query assignment for wireless sensor networks. *AEU-International Journal of Electronics and Communications* 69, 8 (2015), 1102–1112.

[18] Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. 2014. Context aware computing for the internet of things: A survey. *IEEE Communications Surveys & Tutorials* 16, 1 (2014), 414–454. https://doi.org/10.1109/SURV.2013.042313.00197

[19] Sheldon M Ross. 2014. *Introduction to probability models*. Academic press.

[20] Arsalan Tavakoli, Aman Kansal, and Suman Nath. 2010. On-line sensing task optimization for shared sensors. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*. ACM, 47–57. https://doi.org/10.1145/1791212.1791219

[21] Sonam Tobgay, Rasmus L. Olsen, and Ramjee Prasad. 2011. *Architecture for Running Multiple Applications on a Single Wireless Sensor Network: A Proposal*. Springer Berlin Heidelberg, Berlin, Heidelberg, 37–45. https://doi.org/10.1007/978-3-642-22726-4_5

[22] Niki Trigoni, Yong Yao, Alan Demers, Johannes Gehrke, and Rajmohan Rajaraman. 2005. Multi-query optimization for sensor networks. In *International Conference on Distributed Computing in Sensor Systems*. Springer, 307–321.

[23] Rahul Kumar Verma, Sourabh Bharti, and Kiran Kumar Pattanaik. 2018. GDA: Gravitational Data Aggregation Mechanism for Periodic Wireless Sensor Networks. In *2018 IEEE SENSORS*. IEEE, 1–4.

[24] Rahul Kumar Verma, KK Pattanaik, Sourabh Bharti, and Divya Saxena. 2019. In-network context inference in IoT sensory environment for efficient network resource utilization. *Journal of Network and Computer Applications* (2019).

[25] Weiwei Wu, Xiangping Zhai, and Yingchao Zhao. 2018. On Minimizing Sensing Time via Data Sharing in Collaborative Internet of Things. *IEEE Access* 6 (2018), 41633–41642.

[26] Yanjun Yao, Qing Cao, and Athanasios V Vasilakos. 2015. EDAL: An energy-efficient, delay-aware, and lifetime-balancing data collection protocol for heterogeneous wireless sensor networks. *IEEE/ACM Transactions on Networking (TON)* 23, 3 (2015), 810–823.

[27] Yong Yao and Johannes Gehrke. 2002. The cougar approach to in-network query processing in sensor networks. *ACM Sigmod record* 31, 3 (2002), 9–18.

[28] Yawei Zhao, Deke Guo, Jia Xu, Pin Lv, Tao Chen, and Jianping Yin. 2016. CATS: Cooperative Allocation of Tasks and Scheduling of Sampling Intervals for Maximizing Data Sharing in WSNs. *ACM Transactions on Sensor Networks (TOSN)* 12, 4 (2016), 29. https://doi.org/10.1145/2955102

[29] Zhangbing Zhou, Deng Zhao, Gerhard Hancke, Lei Shu, and Yunchuan Sun. 2016. Cache-aware query optimization in multiapplication sharing wireless sensor networks. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 48, 3 (2016), 401–417. https://doi.org/10.1109/TSMC.2016.2598398

[30] Zhangbing Zhou, Deng Zhao, Lu Liu, and Patrick CK Hung. 2018. Energy-aware composition for wireless sensor networks as a service. *Future Generation Computer Systems* 80 (2018), 299–310. https://doi.org/10.1016/j.future.2017.02.050