Optimizing Data Acquisition by Sensor-Channel Co-allocation in Wireless Sensor Networks

Yinfeng Wang, Cho-Li Wang  
Department of Computer Science  
The University of Hong Kong  
Pokfulam Road, Hong Kong  
{yfwang, clwang}@cs.hku.hk

Jian-Nong Cao, Alvin Chan  
Department of Computing  
The Hong Kong Polytechnic University  
Hong Kong, the P.R. China  
{csjcao, cstschan}@comp.polyu.edu.hk

Abstract—Wireless sensor networks (WSNs) should handle multiple sensing tasks for various applications. How to improve the quality of the data acquired in such resource constrained environment is a challenging issue. In this paper, we propose a sensor-channel co-allocation model for scheduling the sensing tasks. The proposed model considers the capability, coupling and load balancing constraints for sensing data acquisition, and can guarantee transmission of sensed data in real-time while avoiding data incompleteness in an efficient way. A spatiotemporal metric called sensing-span is proposed to evaluate the tasks’ execution cost of achieving desired data quality. We extend computation task scheduling algorithms to support sensor-channel co-allocation problem and a heuristic called Minimum Service Capability Fragment (MSCF) is introduced for task scheduling to minimize the waste of reserved channel capacity. Simulation results show that MSCF can improve the performance of data acquisition in WSNs as compared with other heuristics, when scheduling a large number of concurrent data acquisition tasks.

Keywords: Data Quality, Timeliness, Completeness, Co-allocation, Scheduling, Heuristic, Wireless Sensor Network

I. INTRODUCTION

Wireless sensor networks (WSNs) are widely used to acquire contextual data about the environment through event-driven or demand-driven mode. The sensed data are of desired quality [1] if they satisfy the requirements of intended use. Data acquisition in WSNs has to meet various requirements, such as keeping data completeness, guarantee the response time, maximizing the channel utilization to improve routing efficiency and minimizing the task execution cost, etc.

Existing work in WSNs mostly focuses on efficient data collection, aggregation and dissemination, thus extending network lifetime and guaranteeing quality of service (QoS). Currently, most of WSNs data management frameworks, coverage mechanisms and sensor selection schemas pay little attention to the sensor data acquisition. They regard the sensing tasks as simple as a few read operations and the desired data are ready for collection [2, 3]. This assumption does not always hold true as the sensors are getting more powerful these days and the sensing tasks assigned to sensors are more sophisticated than before [4, 5].

In demand-driven applications sensors remain silent until they receive a request from the monitoring station [6]. We assume on-demand sensing task has the control information and instructions which enable expected type sensor to operate execution plan. The sensing task execution needs to achieve the specified Data Quality (DQ) dimensions and preference, such as measurement frequency, resolution of field-of-view, etc. Since more and more applications try to obtain interested data from the same WSN, the limited bandwidth and energy become obstacles that prevent desired quality data acquisition from WSNs.

For example, we can choose more sensors (e.g., k-coverage [7]) to improve the coverage and accuracy. However, if the selected sensor output continuous, ordered data streams, the bandwidth can be easily saturated. A large amount of sensed datum must be transferred in real time among sensor nodes. Otherwise, it could hold back the execution of some time-sensitive sensing tasks or the network bandwidth could be easily saturated due to network jitter. Moreover, the ineffective energy consumption would reduce the WSNs lifetime, so the overall data quality could be affected. Therefore, the WSNs data management should embrace the data acquisition process not only proactively determine the “best” combination of the available sensors and tasks, but also guarantee the obtaining of desired quality data cost-effectively.

The problem of sensor scheduling arises when one (or multiple) sensor(s) has to be selected out of N given sensors at every time step for taking measurements or cooperation. One way to reduce energy consumption is to dynamically schedule sensors’ work/sleep cycles [8]. TinyDB [2] adopts acquisitional query processing in query execution through actively control when and where data is collected. Other methods include query optimization through cache [3, 9], cross-layer information scheduling [10] based on efficient construction of neighbors nodes overlay. Cluster based scheduling [11] focuses on reducing the scheduling communication overhead by considering the devices to be grouped into logical clusters. But it did not consider how to guarantee the data transmission. Sensor-mission assignment in constrained environment [5] tries to support multiple mission assignment to minority sensors. It focuses on choosing best “match” of sensors and missions for improving the system utility. Many sensor scheduling algorithms employ ad hoc sensor scheduling techniques that
modify communication requirements in response to network conditions. However when a large number of concurrent data acquisition tasks emerge, WSNs need to offer sensed data varying from amount to DQ, so there is no “one size fits all” DQ as sensed data is application-specified. Therefore, using scheduling method to allocate required WSN resources for sensing task independently can facilitate meeting different DQ demand.

We propose a novel sensor-channel co-allocation method for on-demand data acquisition problems: assigning tasks to sensors for the tradeoff between accurate detection and rapid response considering the capacity, coupling and load balancing constraints of sensing task execution, as well as assuring timeliness and completeness of sensed data. In the sensor-channel co-allocation solution, once the assigned task could be executed, the allocator needs to ensure that a dedicated communication channel is also allocated to delivery data produced by the task.

Providing delay-guarantee in WSNs is a very difficult problem due to both the scarcity of the wireless resources and unpredictable channel-variation. Hence, we assume no global time synchronization in distributed WSNs. A token-based data delivery mechanism is adopted to avoid data delivery collision and guarantee delivery within bounded latency. Sensor does not perform a given task until the channel token is acquired. Even the sensor’s clock is likely unsynchronized it will not affect task execution at other sensors through the token passing mechanism.

Sensing-span is a weighted spatiotemporal cost metric for assessing the location-based sensing accuracy, execution time and transmission delay so as to guarantee the scheduling efficiency. Because the bandwidth is a load-dependant resource, its service response time will decrease as the number of requests grows. How to maximize the utilization of channel capacity is important since the bandwidth cannot be reused.

The major contributions of this paper are summarized as follows.

(1) We propose a sensor-channel co-allocation approach to allocate the sensors and limited communication resources for sensing task (e.g., surveillance or latency-critical applications) scheduling. We analyze correlations of DQ and propose a novel DQ metric: sensing-span (Section II).

(2) We extend the traditional scheduling algorithms for computation task only to support the sensor-channel co-allocation problem. Furthermore, in order to better utilize the channel resource, a Minimum Service Capacity Fragment (MSCF) heuristic is introduced (Section III).

(3) Extensive simulations were carried out on a wide variety of sensor, channel scales to evaluate the performance of these heuristics. Empirical results show that on average the Sufferage algorithm combined MSCF heuristic achieved higher performance and lower sensing cost (Section IV).

The paper is structured as follows: in section II we define the formal model for sensing task execution, DQ metrics and co-allocation problem statement. In section III, we propose a minimum service capacity fragment heuristic and apply it in extended co-allocation scheduling algorithm. Simulation results, observations and comparison with traditional scheduling algorithms are proposed in Section IV. Related work, including some WSN routing techniques try to strike a balance between energy consumption and data quality, multi-hop task mapping and scheduling are discussed in Section V. We conclude in Section VI.

II. CO-ALLOCATION MODELING

A fundamental information processing challenge over WSNs is to process sensor information to improve quality and performance while minimizing required resources in terms of bandwidth and energy reserve. A quality-aware sensor data engine should cost-effectively manage both sensor capacity and restricted radio bandwidth to provide on-demand DQ.

4. Queuing performance model with multiple-classes

Since resources of WSN have a finite capacity of performing task, e.g., TinyOS and many other sensor embedded operating systems only have a single thread of execution, a wireless link can only transmit a certain number of bits per second. As the WSN shared by many applications, when several applications running concurrently and may want to access the same resource at the same time, tasks can be grouped as multiple-classes through K-means clustering to characterize the heterogeneous DQ demands. Thus, the WSN can be characterized as a queue consists of R different classes of tasks and M service centers (sensors & channels).

The sensing task life-cycle states and notions are presented in Fig.1 and Table I.

![Figure 1. Sensing task life-cycle](image)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ready($X_i$)</td>
<td>Task ready time, the Task $X_i$ is waiting to be assigned to a sensor.</td>
</tr>
<tr>
<td>Wait($X_i$)</td>
<td>Period of time that the task is waiting in sensor’s task queue for some event to occur (such as sensor awake).</td>
</tr>
<tr>
<td>Start($X_i$, $S_j$)</td>
<td>Task start time, once task $X_i$ starts execution on the assigned sensor.</td>
</tr>
<tr>
<td>Exe($X_i$, $S_j$)</td>
<td>Period of time that $X_i$ are being executed on sensor $S_j$.</td>
</tr>
<tr>
<td>Finish($X_i$)</td>
<td>Task finish time means task $X_i$ has finished execution.</td>
</tr>
</tbody>
</table>
The sensing work state of the WSN represents a distribution of sensing tasks over classes and sensors. The WSN work state is denoted by a vector \( \mathbf{N} = (n_1, n_2, \ldots, n_k) \) where component \( n_i (i=1, \ldots, k) \) is a vector that represents the number of tasks of each class at sensor \( i \). Given the arrival task request rate \( \lambda \), and let \( \mu (k), k=1, 2, \ldots, M \) be the complete rate, suppose there are at most \( M \) sensors awake, for any \( K>M \) requests will join the queue and its complete rate is \( \mu (M) \). Using Generalized Birth-Death (Markov Model) process [12] any particular state \( k \) in single queue performance is given below:

\[
P_k = \begin{cases} 
P_0 \lambda^k \left( \frac{1}{\mu(M)} \right)^{K-M} \prod_{i=0}^{k-1} \frac{1}{\mu_{i+1}} & \text{for } k = 1, \ldots, M \\
0 & \text{for } k = M + 1, \ldots 
\end{cases}
\]

If the WSN’s queue size is limit \( N \) (\( N>M \)) include requests which waiting or executing sensors, the rejection probability is \( P_{\text{reject}} = P_N \)

Where: \( P_0 = \sum_{k=0}^{N} \prod_{i=0}^{k-1} \frac{\lambda}{\mu_{i+1}} \).

Then the average response time \( R_{\text{time}} \) can be computed from Little’s law [13]:

\[
R_{\text{time}} = \frac{\text{queue length}}{\text{Arrive rate} \times R_{\text{time}}} = \frac{\sum_{k=1}^{K} R \times \lambda \times (1-P_{\text{reject}})}{\lambda \times (1-P_{\text{reject}})} = \frac{\sum_{k=1}^{K} R \times \lambda}{\lambda \times (1-P_{\text{reject}})} \tag{1}
\]

The workload intensity of a multi-class model with \( R \) classes and \( K \) sensors is represented by the vector \( \mathbf{N} = (N_1, \ldots, N_R) \), where \( N_r \) indicates the number of class \( r \) tasks in the WSN. The weakness of Markov model is the state space explosion. Mean Value Analysis (MVA) [12] adopts recursion instead of solving a set of simultaneous linear equations to get performance metrics from possible system steady state. The residence time \( R_{\text{r}} \) corresponds to the total time a class \( r \) task spends at sensor \( i \) during its execution. The average response time of class \( r \) tasks can be written as \( R_{\text{r}} (\mathbf{N}) = \sum_{i=1}^{K} R_{\text{r}} (\mathbf{N}) \). Class \( r \) system throughput based on MVA yields

\[
\text{Throughput}_{r} = \frac{N_r}{\sum_{i=1}^{K} R_{\text{r}} (\mathbf{N})} \tag{2}
\]

Moreover, the WSN bandwidth efficiency depends on the number of the sensors trying to transmit data. If sensor tries to transmit and medium is busy, 802.11 exponential backoff mechanism increase maximum backoff time exponentially. According to Queuing Theory, the time spent in the waiting line at an exponential server is on average twice the time spent in the waiting line of a constant speed server.

Transmitting sensed data in real-time has two advantages, that is, enhancing the utilization of WSN and reducing the likelihood of the “gridlock”, task will not finish until the sensed data is sent out. Therefore, we propose a sensor-channel co-allocation approach to enable WSN efficiently produce timely and complete sensed data without having to worry about available bandwidth and sensor’s data buffer.

B. Data Quality correlations

To guarantee the data accuracy, the completeness and timeliness are two dimensions that can be controlled during the task execution. Data completeness can be affected by the fluctuation conditions such as sensor’s capability or sensing coverage etc. WSNs control or updating ways of management will affect the data timeliness as Fig.2 shows. These uncertainties raise inaccuracies which consist of data missing due to delayed, incomplete or that exists but should not. We define three DQ metrics: average task data missing ratio, average task finish time loss ratio and sensing-span for measuring the data timeliness, completeness and sensing cost.

![Figure 2. DQ dimensions and correlations of data acquisition](image)

Average task data missing ratio reflects the situation that sensing tasks fail to deliver sensed data to the receiver. The remnant of task \( X_i \) generated data which need to save into \( S_j \) buffer storage during the time interval \( E(x_i, s_j) \) can be represented as:

\[
R_{s_j}(t) = \int_0^{E(x_i, s_j)} (G_{X_i}(t) - Trans_{X_i}(t)) d(t)
\]

Where \( G_{X_i}(t) \) is the varying amount of generated data when \( X_i \) be executed on \( S_j \), \( Trans_{X_i}(t) \) is the varying amount of data successfully transmitted during \( X_i \) execution.

If \( Trans_{X_i}(t) > 0 \), means no other task’s data waiting in buffer for transmission, the affected task number:

\[
\delta(S_j) = \begin{cases} 
1; & \text{Buffersize} < R(t) \\
0; & \text{otherwise.} 
\end{cases}
\]

If \( Trans_{X_i}(t) = 0 \), means there are other tasks’ data waiting in buffer for transmission, so:

\[
\delta(S_j) = \begin{cases} 
0; & R(t) < \text{Available Buffersize} \leq N \\
N + 1; & \text{Buffersize} < R(t)
\end{cases}
\]

Where \( N \) is the total number of tasks whose data saved in \( S_j \) buffer, \( n \) is the tasks number which data is overflowed by newly saved data. The total (\( K \)) tasks’ average task data loss ratio can be represented as follows:

\[
\bar{\delta} = \frac{1}{K} \sum_{j=1}^{K} \delta(S_j) \tag{3}
\]

Assume task \( X_i \) is the only job to be executed and its
routing just takes one hop, the earliest finish time of this task is $\text{Finish}(X_i)$; when scheduling with other tasks, its finish time is $\text{Finish}(X_i)$, so the finish time loss ratio of task $X_i$ is: $\omega = (\text{Finish}(X_i) - \text{Finish}(X_j))/\text{Finish}(X_j)$, it ( $\omega$ ) reflects the degree of delay which is caused by resources competition of scheduling.

The total (K) tasks’ average finish time loss ratio can be represented by arithmetic mean value:

$$\overline{\omega} = \frac{1}{K} \sum_{i=1}^{K} \omega(X_i)$$  \hspace{1cm} (4)

C. Sensing-span

Because the WSNs are resource constrained, we think there are three kinds of constraints on sensing data acquisition.

1. Capacity constraint: according to the size and cost restriction, the sensor node usually owns the limited sensing radius, buffer size, power and computational speed. Due to the uncertainty of detecting target, the larger sensing radius, the more successful detections.

2. Coupling constraint: time and location coupling require the sensing data should available at right place at right time. Sensing data often needs several nodes forward data packets to the base station. That demand bandwidth and relay nodes are all available simultaneously. Obviously a large capacity data buffer size entails more opportunities to transmit data successfully. However, more energy is required for corresponding data retransmission.

Data accuracy is the most important dimension in DQ, the others dimensions of DQ are meaningless without correct value. According to the sensor’s position, target place and sensing radius [8], each sensor’s the occupied sensing range is simply based on the ratio between the Euclidean distance from target to sensor and sensing radius, so the

$$\text{SurplusRange}(S_j) = 1 - \frac{\text{EuclideanDistance}(S_j,\text{Target})}{\text{SensingRange}(S_j)}$$  \hspace{1cm} (5)

The more surplus range means the greater robust perception and the less relative distance between the sensor and its target, so the fewer detection time and the higher sensing resolution will be, that will improve the data accuracy. E.g., sound amplitude and the amount of object details decaying with distance, the strength of the electromagnetic signal decreases $1/r^2$ in strength where $r$ is the distance away from the transmitter.

In addition to exploring the location of sensing task execution, we are particularly interested in the impact of the tradeoff among task execution location, execution time and data transmission delay, hence a spatiotemporal metric is proposed to evaluate the scheduling efficiency.

**Definition 1. Sensing-span:** the sensing task’s execution cost is determined by the assigned sensor’s occupied sensing range, execution delay (task finish time loss ratio), and the data transmission delay ratio.

$$\text{Sensing-span} = W_1 \times \omega + W_2 \times \frac{\text{EuclideanDistance}(S_j,\text{Target})}{\text{SensingRange}(S_j)} + W_3 \times \frac{\text{DeliveryDelay}}{\text{Timeout}}$$  \hspace{1cm} (6)

Where $W_1$, $W_2$, $W_3$ are the weight factors in the range of $[0,100]$. In WSNs, the DQ tend to co-vary with the constraints, providing a rich collection of contextual associations, date engine should adjust the weight based on the preference for more SurplusRange($S_j$) or shorter delay. Delivery delay occurs when data cannot be sent in real-time. The timeout is the maximum time the receiver will wait for sensed data before aborting.

Assuming that the task set to be scheduled is $X = \{X_i | 1 \leq i \leq K\}$, the total sensing-span can be represented as $\sum_{\text{task}} \text{sensing-span}(X_i)$ for calculating the overall sensing tasks’ execution cost for achieving required DQ.

D. Resources allocation of data engine

Generally the wireless frequency range is divided into multitude of channels. TDMA/FDMA/CDMA protocols are not preferred for a WSN that has limited computing power and lack of special hardware. Multi-channel communication is an efficient method through parallel transmissions over different frequency channels [14]. However, global time synchronization of WSNs is a non-trivial issue.

In order to provide a stringent delay and bandwidth guaranteed control, the Wireless Token Ring Protocol (WTRP) is implemented on top of 802.11 [15] is adopted in sensor-channel co-allocation mechanism, each channel only
has one token. WTRP can facilitate the channel reservation without worry the time synchronisation; its saturation operating mode supports a station sending data packets continually. Moreover, WTRP advantages include of robustness against single node failure, supporting flexible topologies, in which nodes can be partially connected and not all nodes need to have a connection with a master.

**Definition 2.** Channel Server Service Time means the latest time when channel server (CS) can meet the bandwidth request, if the task needs channel number is k, the Channel Server Service Time can be represented as CS(Ck, t).

The channel server C_{Si} can provide C_{i}^{\text{max}} channels at most, and C_{i}^{\text{used}}(t) represents the numbers of channel which have been used, then C_{i}^{\text{avail}}(t) = C_{i}^{\text{max}} - C_{i}^{\text{used}}(t) means the available channels number at the time t.

**Definition 3.** Task Required Resources: Assuming the data engine assigns global identifier X_{i}(1 \leq i \leq n) to accepted task, so every task required resources can be expressed as X_{i} = (S_{i}, C_{i}^{R}), S_{i} is the sensor on which task X_{i} is assigned to execute, C_{i}^{R} here stands for the channel numbers of task X_{i} demands.

**Definition 4.** Sensor Availability: A(j, t) represents the availability of the sensor S_{j} at time t, it can be either 1 or 0, if there is no task executing on sensor S_{j} at time t, then A(j, t) = 1, otherwise A(j, t) = 0.

A sensor data engine’s schedule policy can be formalized as a 4-tuple: P = (X_{i}, S_{i}, C_{ik}, t).

Where S_{j} is the sensor on which task X_{i} is being executed, C_{ik} is the channel resource that allocated to task X_{i}, t is the start time Start(X_{i}, S_{j}).

Fig.4 shows data engine workflow. Because the sensor is error-prone, once response timeout error occurs the task has to be reallocated if needed.

![Figure 4. Data engine workflow](image)

**E. Assumption**

To simplify the construction of co-scheduling model, we give some common assumptions here:
(1) No priorities associated with sensing tasks.
(2) Each sensor node is stationary and knows its own location. An event can be detected by multiple sensors

**nodes through k-coverage [7] sensor placement scheme.**

(3) For the same type sensors, the shorter distance between the sensor and its target, the less detection cost and higher sensing resolution will be.

(4) First come first served policy is used in sensing task queue and channel token distribution.

(5) All the sensors are under the control of data engine. The estimated sensing-span that a task executes on all possible sensors is known.

**F. The performance metrics**

We make use of three performance metrics in table II to evaluate the system throughput and availability. Makespan describes the duration of total tasks execution time. Average token utilization is the channel utilization efficiency measurement. Average load balancing loss ratio represents unbalanced workload distribution among WSN which reflects task competition for perfect sensor due to accurate detection or lower execution cost. Since WSN usually be randomly deployed, it is hardly to achieve the well-balanced task distribution. If scheduling algorithm can offer lower average load balancing loss ratio, the longer WSN lifespan will be achieved.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Makespan</td>
<td>Makespan = Finish(task) - Ready(X_{i})</td>
</tr>
<tr>
<td>Average token utilization</td>
<td>( \lambda = \frac{\sum_{i=1}^{n}\text{Exe}(X_{i}, S_{i}) \times C_{i}^{R}}{\text{Makespan} \times C_{i}^{R}} )</td>
</tr>
<tr>
<td>Average load balancing loss ratio</td>
<td>( \theta = \frac{\sum_{i=1}^{n}(\text{TotalSensing-span@S}_{i} - \text{TotalSensorNumber})}{2 \times \text{TotalSensorNumber}} )</td>
</tr>
</tbody>
</table>

**G. Co-allocation problem statement**

Based on the above assumptions and analysis, the quality-aware sensor data acquisition problem can be depicted as a multi-objective optimization problem (MOP):

**Input:** task set \( X = \{X_{i}|1 \leq i \leq K\} \), sensor set \( S = \{S_{j}|1 \leq j \leq N\} \), and channel server set \( CS = \{CS_{k}|1 \leq k \leq L\} \).

**Output:** \( P = \{(X_{i}, S_{j}, C_{ik}, t)|1 \leq i \leq K, 1 \leq j \leq N, 1 \leq k \leq L\} \). Where optimize the following objectives:

1. \( \min(\theta) \);
2. \( \min(\delta) \);
3. \( \min(\text{total sensing-span}) \);
4. \( \min(\text{Makespan}) \);
5. \( \max(\alpha) \);
6. \( \min(\theta) \).

**Constraints:**

1. Where \( \text{SurplusRange}(S_{j}) > 0 \),
\( \forall((X_{i}, CS_{k}, \text{Start}(X_{i}, S_{j})), (X_{k}, CS_{k}, \text{Start}(X_{k}, S_{j})) \in P \)
\( S.t. \text{Finish}(X_{i}) < \text{Start}(X_{j}) \)
2. \( \forall t \in \big(\text{Start}(X_{i}, S_{j}), \text{Start}(X_{i}, S_{j}) + \text{Exe}(X_{i}, S_{j})\big) \)
\( S.t. C_{i}^{R} \leq C_{i}^{R}(t) \wedge A(j, t) = 1 \)
A common approach for solving the MOP is scalarization [16]: A new objective value is computed as a linear combination of the individual objective functions. Because of the task scheduling problem is a NP-hard problem in heterogeneous compute environment, there is no single optimal schema for all pervasive computing scenarios, so the heuristic algorithm is essential. In reference [17], Genetic Algorithm take significantly longer average execution time for most problem (around 60 seconds compared to under a second for min-min), while A* has an exponential complexity, so these two kinds of algorithms do not suitable for sensing task scheduling.

III. SCHEDULING ALGORITHM

Sensors and channel servers may have multiple matched available time slots, but not all of them long enough to run the tasks. We extend traditional heuristic scheduling algorithms (e.g. OLB, Fast Greedy, Min-Min, Max-Min, Xsufferage, etc.) originally used for computation task only allocation to the sensor-channel co-allocation problem by modifying the computation earliest task finish time algorithm.

Algorithm 1. The Earliest task Finish \((X_i,j,C_k)\) time

**Input**: Channel server set \([CS_k \mid 1 \leq k \leq L]\), \(\text{Min FinishTime} < T_{\text{max-value}}\) : Sensor set \([S_j \mid 1 \leq j \leq N]\), Task set \([X_i \mid 1 \leq i \leq R]\).

**Output**: Minimum FinishTime \((X_i,j,C_k)\).

/*Search for the \(X_i\)'s minimum finish time from all the sensor and channel combination time slots.*/

**Begin**

1. Set \(\text{Min FinishTime} = \text{Max-value}\);
2. for \((\text{int } i = 0; i < N; i++)\) {
3. do (4.
4. if \((C_i^{\text{avail}}(t) \geq C_i \land \text{Exe}(X_i,j) \leq CS_i(\text{avail-time-slot}))\) {
5. TaskFinishTime = \(t + \text{Exe}(X_i,j)\);
6. Break; // end if
7. Find next \(C_i^{\text{avail}}(t)\); // The available tokens at the time \(t\).
8. } while \((t < \text{Max-value})\)
9. if \((\text{Min FinishTime} > \text{TaskFinishTime})\)
10. \(\text{Min FinishTime} = \text{TaskFinishTime}\);
11. //End for;
12. Return Min FinishTime;

**End**

We regard the channel server service capacity as the token-time product:

\[
\text{ServiceDemand} = R \times (t_1 - t_0)
\]  

(7)

Where \(R\) is the amount of tokens occupied during the real time interval \((t_0, t_1)\), the requirement unit is 1 token.

**Definition 5. Maximum Service Capacity**: As the Fig. 5 time histogram shows, the available maximum channel server service capacity of a certain time is the biggest sub-rectangle area that histogram can accommodate.

There are many algorithms [18] for computing the biggest sub-rectangle area. After channel tokens being allocated to tasks, the unused service capacity will become new service capacity histograms: fragments. We introduce the heuristic to improve the channel resources utilization based on following considerations:

1. For these unused fragments have a few tokens and short available time, its probability of satisfying other tasks requirement is relative less.
2. Because of the sensor’s sequential execution mode and the lapse of time, these time-related fragments are unrecoverable. So the efficient scheduling should leave fragments as less as possible.

**Algorithm 2: Sufferage-E**

In order to cost-effectively manage the restricted channel resources, a Minimize Service Capacity Fragment (MSCF) heuristic is proposed. When channel server cannot meet the task \(A\) and \(B\) token requirements simultaneously, if execute \(A\) would generate fewer service capacity fragments than \(B\), then choosing \(A\) could utilize more channel service capacity.

The insight of Sufferage algorithm is: if the effect on the finish time of the task which couldn’t get required resources is more serious than on other tasks’, then the resources should be allocated to this task. So, when candidate tasks have same sensing-span sufferage value, use MSCF heuristic will improve the sensing task scheduling efficiency.

**Input**: Task set \([X_i \mid 1 \leq i \leq R]\), Sensor set \([S_j \mid 1 \leq j \leq N]\), Channel server set \([CS_k \mid 1 \leq k \leq L]\)

**Output**: \(\text{Schedules} = [(X_i,j,C_k) \mid 1 \leq i \leq R, 1 \leq j \leq N, 1 \leq k \leq L]\).

**Begin**

1. set \(\text{Schedules} = \emptyset\);
2. for each sensor \(S_j \in S\), and each channel server \(CS_k \in CS\),
3. set \(S_j\) and \(C_k\) as unallocated;
4. end for
5. do until \(X = \emptyset\);
6. for each task \(X_i \in X\),
7. find \(X_i\)'s earliest and second earliest finish time \(t_{1i}, t_{2i}\) by algorithm 1;
8. \(X_i\)'s Sufferage value = Sensing-span\(_{s1}\) - Sensing-span\(_{s1}\);
9. if the \((S_j,C_k)\) pair that gives \(X_i\)'s earliest finish time is unallocated;
10. then
11. add(Xᵦ, Sᵦ, Cᵦ, Start(Xᵦ, Sᵦ)) to Schedules;  
12. set Sᵦ, Cᵦ as allocated;  
13. else if the task Xᵦ to which (Sᵦ, Cᵦ) is allocated has less Sufferage value than Xᵦ’s;  
14. Or else if Xᵦ and Xᵦ have the same Sufferage value but Xᵦ is better than Xᵦ according to MSCF heuristic  
15. then  
16. remove (Xᵦ, Sᵦ, Cᵦ, Start(Xᵦ, Sᵦ)) from Schedules;  
17. Add (Xᵦ, Sᵦ, Cᵦ, Start(Xᵦ, Sᵦ)) into Schedules;  
18. end if;  
19. end if;  
20. end for;  
21. jump to step 5;  
22. end if;  
End  

IV. TEST AND EVALUATION  
A. Experimental settings  
We set all Expected Sensing-Span (ESS) matrixes are inconsistent through the methods from reference [17]. Considering the sensor’s location and heterogeneity, the value of expected sensing-span is within the range from 100 to 500. Token heterogeneity reflects a variety of task’s channel requirement according to sensing data volume.  

<table>
<thead>
<tr>
<th>Input Data</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task Number</td>
<td>Large &gt;1000</td>
</tr>
<tr>
<td>Sensor Number</td>
<td>&gt;300</td>
</tr>
<tr>
<td>Channel Token</td>
<td>&gt;150</td>
</tr>
<tr>
<td>Token Heterogeneity</td>
<td>≥3</td>
</tr>
<tr>
<td>Expected Sensing-span</td>
<td>100~200</td>
</tr>
</tbody>
</table>

Due to space limitation, only some representative test cases are presented in Table IV.  

<table>
<thead>
<tr>
<th>Name</th>
<th>Task</th>
<th>Sensor</th>
<th>Token</th>
<th>Token heterogeneity</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>400</td>
<td>80</td>
<td>60</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>1000</td>
<td>80</td>
<td>60</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>2000</td>
<td>80</td>
<td>60</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>500</td>
<td>100</td>
<td>50</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>1000</td>
<td>100</td>
<td>150</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>2000</td>
<td>400</td>
<td>200</td>
<td>2</td>
</tr>
<tr>
<td>G</td>
<td>2000</td>
<td>400</td>
<td>300</td>
<td>3</td>
</tr>
</tbody>
</table>

To avoid stochastically generated ESS value matrix affecting the algorithms’ performance, the results presented in this section are averaged over 200 simulation runs of each test case.  

B. Results and observations  
Current WSNs just apply FCFS policy to execute sensing tasks. From below test result we can see FCFS algorithm’s timeliness will suffer more than 25~35% latency compare to the best and second best co-allocation scheduling algorithms in relative small scale tests. OLB assign task in arbitrary order to the next available sensor and Max-min algorithm always tries to run the longer task firstly, so their average task finish time loss ratio are the largest as Fig.6 shows.  

Bianchi’s IEEE 802.11 saturation throughput model [19] simplified assumption that all colliding packets are lost. We assume the task data will be successfully transmitted to destination use bulk-transfer protocol unless collisions occur, and set simulator’s throughput configuration consistent with the Robinson’s analysis [20], each sensor generated data rate is randomly chosen from 20~40 Kbps and the sensor data buffer size is 1024Kb.  

Fig.7 shows 20 runs result of each test case A, B and C using FCFS scheduling algorithm, when there are more tasks, the task data loss ratio will around 10~13% since the average sensing task work load is more balanced than fewer task number. Sensor data engine use co-allocation scheduling policy will not led to lose data because the sensed data can be transmitted in real-time and collision-free.  

Fig.8 shows average token utilization of each test case A, B and C.
The following observations are made:

(1) As the task and sensor number increased, channel token utilization will become the bottleneck for the wireless frequency is limited as Fig.8 shows.

(2) Once channel resources become the bottleneck, when task number increased, all the six scheduling algorithms result in the similar throughput as Fig.9 and 10 shows.

(3) Different scheduling algorithm cause diverse average load balancing loss ratio as Fig.11 shows.

For observation (1), the average token utilization is around 94% about test case F and G. When task need more tokens for data transmission, how to enhance the token utilization to improve the scalability become a challenging issue.

For observation (2), when WSN is heavy loaded (more tasks but relative less sensors), the OLB’s and Max-min performance degrade significantly. Fast Greedy, Greedy, min-min and X-Sufferage have good throughput performance. Results from simulation shows when task number is huge (2000), and sensor number also increased, max-min’s sensing-span and makespan are greatest in test case F and G, other algorithms’ performance deviation just about 1%, because the channel resource become the bottleneck.

For observation (3), the average load balance loss ratio of max-min and X-Sufferage are less than others in the test that represents these two algorithms can offer better load balancing.

All metrics except token utilization prefer the minimum value, the rank of these metrics is calculated by \( \sum_{i=1}^{Case number} \frac{Value-Minimum}{Minimum} \), while the rank of token utilization is calculated by \( \sum_{i=1}^{Case number} \frac{Value-Maximum}{Maximum} \).

The weight can be adjusted according to the importance of the metrics. We set the all the 5 metrics weighting equal, the less total rank value means the better performance this algorithm can offer. The six heuristic co-allocation algorithms’ performance is compared in table V and X-Sufferage achieve the best integrated performance than other algorithms.

### Table V: The Comparisons of Algorithms

<table>
<thead>
<tr>
<th>Name</th>
<th>Sensing-span</th>
<th>Make span</th>
<th>Token Utilization</th>
<th>Load Balancing Loss</th>
<th>Task Finish Time Loss</th>
<th>Total Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>OLB</td>
<td>0.33</td>
<td>0.33</td>
<td>0.07</td>
<td>0.3</td>
<td>0.33</td>
<td>1.26</td>
</tr>
<tr>
<td>Fast Greedy</td>
<td>0.05</td>
<td>0.05</td>
<td>0.04</td>
<td>0.39</td>
<td>0.14</td>
<td>0.67</td>
</tr>
<tr>
<td>Min-min</td>
<td>0</td>
<td>0</td>
<td>0.05</td>
<td>0.66</td>
<td>0</td>
<td>0.71</td>
</tr>
<tr>
<td>Max-min</td>
<td>0.26</td>
<td>0.24</td>
<td>0.02</td>
<td>0.07</td>
<td>0.35</td>
<td>0.94</td>
</tr>
<tr>
<td>Greedy</td>
<td>0</td>
<td>0</td>
<td>0.05</td>
<td>0.66</td>
<td>0</td>
<td>0.71</td>
</tr>
<tr>
<td>XSufferage</td>
<td>0.01</td>
<td>0.03</td>
<td>0</td>
<td>0</td>
<td>0.11</td>
<td>0.15</td>
</tr>
</tbody>
</table>

#### C. Comparison

The DQ and performance comparison of the X-Sufferage and Sufferag-E under large scale test is shown in Table VI.

### Table VI: Test Result Comparison

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Test Case</th>
<th>X-Sufferage</th>
<th>Sufferage-E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average task finish time</td>
<td>E</td>
<td>704.210</td>
<td>703.043</td>
</tr>
<tr>
<td>ratio</td>
<td>F</td>
<td>609.732</td>
<td>607.262</td>
</tr>
<tr>
<td></td>
<td>G</td>
<td>620.356</td>
<td>617.935</td>
</tr>
<tr>
<td>Sensing-span</td>
<td>E</td>
<td>201629</td>
<td>201542</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>201772</td>
<td>201671</td>
</tr>
<tr>
<td></td>
<td>G</td>
<td>201653</td>
<td>201565</td>
</tr>
<tr>
<td>Makespan</td>
<td>E</td>
<td>1608</td>
<td>1609</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>1413</td>
<td>1412</td>
</tr>
<tr>
<td></td>
<td>G</td>
<td>1716</td>
<td>1717</td>
</tr>
<tr>
<td>Average Token utilization</td>
<td>E</td>
<td>94.72</td>
<td>94.81</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>94.145</td>
<td>94.167</td>
</tr>
<tr>
<td></td>
<td>G</td>
<td>96.823</td>
<td>96.736</td>
</tr>
<tr>
<td>Average load balancing</td>
<td>E</td>
<td>55.79</td>
<td>54.77</td>
</tr>
<tr>
<td>ratio</td>
<td>F</td>
<td>56.1</td>
<td>52.142</td>
</tr>
<tr>
<td></td>
<td>G</td>
<td>55.758</td>
<td>54.844</td>
</tr>
</tbody>
</table>

Using MSCF heuristic can improve the channel utilization and reduce the average load balancing loss ratio.
around 1%. Due to efficient channel utilization, the sensing-span and average task finish time loss ratio are also reduced. The added computation time of using MSCF heuristic can be overlapped by the tasks waiting-for-scheduling time. So applying MSCF heuristic would enhance the data engine’s performance while not postpone the task’s scheduled start time.

V. RELATED WORK

Some WSN routing techniques try to strike a balance between energy consumption and DQ through the coordination and management of the sensing activity. For example, hierarchical routing [21] is an efficient way to lower energy consumption but not for optimal route. The gradient set up phase of Gradient-based routing [22] is expensive in terms of latency and energy consumption. Negotiation-based routing protocol [21] solved the congestion problem by making concession of DQ, but cannot guarantee the data delivery. QoS-based routing [23] adapts between the QoS and energy saving. Hence the data completeness and timeliness cannot be guaranteed by these data-centric routing protocols.

At the sensed data management level, scheduling focus on using statistical method to derivate relationships between event detection and sampling [24], energy efficient routing and clustering [25], taking task’s QoS into account to determine when and where to perform the aggregation[26] in distributed networks to reduce the redundant data. Lance [27] couples the use of optimized, reliable data collection with an energy cost model for extracting data from WSNs.

Multi-hop task mapping and scheduling (MTMS) [4] model is most similar research to ours which schedule the computation tasks and associated communication events simultaneously. This joint scheduling uses multi-casting in tree-like topology and adopts penalty function to control the collision and find optimal routing. However, it focuses on computation task scheduling and large scale tasks mapping is not explicitly discussed in [4]. Different from MTMS, our proposed co-allocation solution guarantee the sensed data completeness of data and explicitly evaluate the scheduling algorithms’ performance under heavy workload. Fully decentralized just-in-time workflow scheduling method [28] allows each node to autonomously dispatch inter-dependent tasks, while how to reduce the communication cost still needs to be studied intensively in WSNs.

Data quality of database was extensively discussed in [29, 30] while the pervasive computing environment’s highly dynamic, inherent uncertainties and the sensor data’s ephemeral nature raise challenges [31]. Based on functional dependencies and contextual relationship constraints, data cleaning [32] and consistency checking [1, 34, 34] can help indentifying incorrect data and improve the DQ. However, how to obtain the correlations among the distributed sensed data can be difficult in some situations, especially those with less well understood domains.

VI. CONCLUSION

As more and more WSNs begin generating high volume, time critical and very high-rate “data streams” [30], DQ-guaranteed sensing task management is crucial in managing data uncertainty. We propose a quality-aware sensor data acquisition engine to facilitate the co-allocation of sensor and channel resources to produce timely and complete data in WSNs. The heuristic scheduling algorithms are extended from for computation resources only to support for sensor channel co-allocation problem. The test results show the MSCF heuristic can improve the sensor data engine’s performance in an efficient way.

The emergence of “Internet of things” [35] will start connecting myriads of objects and devices of all kinds which are interoperable and able to act independently depending on the context, using negotiation mechanism to facilitate satisfying of different tasks’ preferences is one of the promising solutions. In the future, we plan to investigate mutual agreed DQ contract and flexible sensors/resources allocation mechanism to acquire quality guaranteed data as well as ensure scalability for pervasive computing.

ACKNOWLEDGMENT

This project is supported by Hong Kong UGC Special Equipment Grant (SEG HKU09) and China 863 grant 2006AA01A111.

REFERENCES
