RC25253 (W1112-080) December 29, 2011
Computer Science

IBM Research Report

QoI-Aware Energy Management in Internet-of-Things Sensory Environments

Zhanwei Sun[1], Chi Harold Liu[2], Chatschik Bisdikian[3], Joel W. Branch[3], Bo Yang[2]

[1]Department of Electrical Engineering
University of Notre Dame
USA

[2]IBM Research Division
China Research Laboratory
Building 19, Zhouguancun Software Park
8 Dongbeiwang West Road, Haidian District
Beijing, 100193
P.R.China

[3]IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

QoI-Aware Energy Management in Internet-of-Things Sensory Environments

Zhanwei Sun[§], Chi Harold Liu[†], Chatschik Bisdikian[♮], Joel W. Branch[♮] and Bo Yang[†]
[§]Department of Electrical Engineering, University of Notre Dame, USA
[†]Department of Networked Computing, IBM Research, Beijing, China
[♮]IBM T. J. Watson Research Center, Hawthrone, USA
E-mail: [§]zsun2@nd.edu, [†]{chiliu, boyang}@cn.ibm.com, [♮]{bisdik, branchj}@us.ibm.com

***Abstract*—Considering physical objects with certain sensing capabilities in an Internet-of-Things (IoT) sensory environment, in this paper, we propose an efficient energy management framework to control the duty cycles of these objects under quality-of-information (QoI) experience in a multi-task-oriented IoT sensory environment. Contrary to past research efforts, our proposal is transparent and compatible both with the underlying low-layer protocols and diverse applications, and preserving energy-efficiency in the long run without sacrificing the QoI levels attained. Specifically, we first introduce the novel concept of QoI-aware "object-to-task relevancy" to explicitly consider the sensing capabilities offered by an object to the IoT sensory environments, and QoI requirements required by a task. Second, we propose a novel concept of the "critical covering set" of any given task in selecting the objects to service a task over time. Third, energy management decision is made dynamically at runtime, to reach the optimum for long-term application arrivals and departures under the constraint of their service delay. Finally, an extensive case study based on utilizing the sensing objects to perform water quality monitoring is given to demonstrate the ideas and algorithms proposed in this paper, and a complete simulation is made to support all performance analysis.**

I. Introduction

The Internet of Things (IoT, [1]) represents a next stage in the evolution of computerized interconnectivity. In it, not only computers but smart physical objects, or "things" will interconnect. It is powered by the proliferation of computerized intelligence, sensors and RFID tags embedded in things (vehicles, buildings, habitants, humans, utility grids, containers, garments, goods, cell phones, etc.) that allows sensing the state of the objects and their surroundings [2]. Through the gathering and processing of this state, a variety of smart applications and services gain awareness of situations impacting the objects (and their surroundings) and act to affect them (e.g,. through environmental control, or utility (electricity, water) grid management systems), or trigger actions involving third party entities (e.g., supply chain monitoring, remote patient monitoring, infrastructures monitoring [3]).

Fig. 1 shows the architectural elements of the IoT systems of interest in this paper. They comprise a collection of applications and services at the top that act and react on the collective knowledge gathered by the computerized intelligence embedded in objects at the bottom. As the figure shows, these objects may belong to a number of domains (e.g., belong to different city agencies) and the IoT applications can cut across

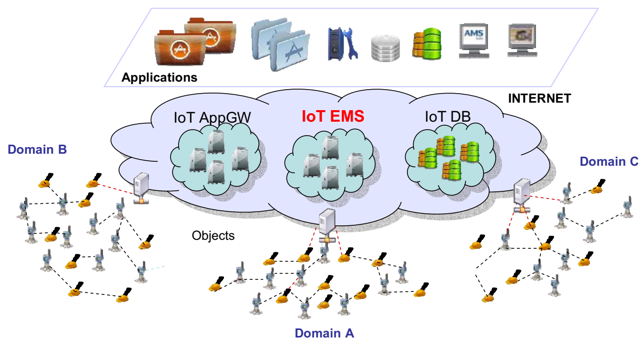

Fig. 1. The overall IoT architecture, where a middleware platform (composed of the IoT application gateway, EMS, and real-time operational database) bridges a variety of applications with the physical objects.

these domains (e.g., a smart cross-agency application, such as traffic management and emergency response). Between the two ends, there is a number of supporting middleware services that facilitate the effective bridging between the varieties of IoT applications and objects, managing the various resources they provide.

In the figure these middleware services are shown across a distributed platform comprising an application gateway for information collection, processing and delivery, a real-time operational database for information archiving and query, and a resource management entity that includes our proposed IoT *energy management server* (EMS) for object energy and application quality management. It is worth noting that the presence of EMS capabilities is closer to the objects, where possibly the EMS functionality could be hierarchically spread/deployed on several intermediary nodes located as necessary anywhere within the end-to-end system. Nevertheless, for ease of exposition, in this paper we consider a centralized system focusing on the fundamentals of our proposal, leaving their extensions to distributed system for future studies.

EMS controls/optimizes the object duty cycles upon receiving the orders from the applications waiting to be executed to deploy different sensing tasks (or, simply, tasks) to the physical world. For instance, "to monitor the water quality of relative position (x_1, y_1, z_1) of Hoover Dam" and "to monitor the pollutant concentration of relative position (x_2, y_2, z_2) of Hoover Dam" are two tasks. Nevertheless, the heterogeneity of the huge number of objects deployed in IoT sensory environments (e.g, with different object types, protocols, manufactures, data schema, etc.), motivates the need of a *system-level* management operation that can work with systems that engage to medium access control (MAC) level energy conservations, as much of previous research efforts[4]. Meanwhile, an efficient scheme should minimize sending the number of control signals crossing different domains, and thus we are seeking a *long-term* optimal solution. Furthermore, many low-level network concerns (e.g., security, networking, energy etc.) remain only within the individual access network, but not exposed to the applications, as end-users mainly focus on the reusability and interoperability of the sensory environments to build/run larger, more dynamic applications. Finally, maintaining these battery-dependent objects in such an environment is far too labor intensive and costly and, hence, energy conservation while producing good quality-of-information (QoI) produced is of paramount importance.

Broadly speaking, QoI relates to the ability to judge whether information is *fit-for-use* for a particular purpose [5], [6], [7]. For the purposes of this paper, we will assume that QoI is characterized by a number of attributes including accuracy, latency, and physical context (specifically, object coverage in this paper [5]).

Here is the central theme of the paper. We aim to design an energy management service (and supporting algorithms) that is transparent to and compatible with any lower layer protocols and up-running applications, while providing the *long-term* energy-efficiency under the satisfactory QoI constraints.

In support of our design, we first introduce the new concept of "object-to-task relevancy" to explicitly consider the sensing capabilities offered by an object (or a set of objects) to the applications and QoI requirements required by a task. Second, we use the generic information fusion function to compute the "critical covering set" of any given task in selecting the objects to service a task over time. Third, we propose a runtime energy management framework based on the previous design elements to control the duty cycles of an object in the long run, i.e., the control decision is made optimally considering the long-term task usage statistics where the service delay of each task serves as the constraint. Finally, an extensive case study related to water quality monitoring is given to demonstrate the ideas and algorithms proposed in this paper, and a simulation is made to support all performance analysis. To the authors' best knowledge, this is the first piece of research that manages the energy usage of a variety of objects from different domains, irrespective of how the provided sensing capabilities will be used by different applications.

The rest of this paper is organized as follows. The system model, including the objects and tasks, are described in Section II, and based upon which, the system flow of the proposed efficient energy management framework is given. Then, the object-to-task relevancy and the critical covering set are introduced in Section III, and the optimization problem of efficient energy management is formulated in Section IV, where several solutions are also given and analyzed. A case study of water quality monitoring is then followed and explained in detail in Section V, and its simulation results are provided in Section VI. Section VII presents the related research efforts. Finally, concluding remarks are drawn in Section VIII.

II. System Model

We consider an IoT sensory environment that comprises a collection $\mathcal{N}$ of N objects (indexed by $n \in \{1, 2, \ldots, N\}$), plus a gateway (the sink). Each object $n \in \mathcal{N}$ is associated with certain sensing, processing and communication capabilities. The sensing capability of an object represents its ability to offer a certain level of QoI to a task, but independently of any specific task. The sensing capability of object n is described by the K-vector $\underline{c}^n \in \mathbb{R}^K$, whose entries include QoI attributes such as the measurement errors, latency in reporting, its coverage, etc. Contrary to the collection $\mathcal{N}$, the gateway is assumed to have sufficient processing power and energy capacity. Finally, we assume that our energy management scheme is running within the IoT EMS, interacting with both the applications and the gateway, such that control signals can be computed, generated, and sent to the objects (see Fig. 1).

We also consider a collection $\mathcal{M}$ of M task types (indexed by $m \in \{1, 2, \ldots, M\}$). Each task type represents a specific class of activities that may share a common spatial property but not temporal properties, such as starting time or duration. For example, "monitor the water quality at location Ω" may represent one of the M tasks, while doing so between times t_1 and t_2 or t_3 and t_4, represents two instances of the same sensing task executed over two different time periods. Each task's desired QoI is described by a K-vector $\underline{q}^m$, describing the desired accuracy, latency, coverage, etc. Note that the elements in $\underline{q}^m$ can be vectors as well, as a QoI requirement can be defined by more than one parameters, as illustrated in a case study in Section V.

We assumed a discrete (or slotted) time system operation. Duty-cycling decision and control operations are made every L time slots, which define the duration of an (L-slot) frame in the system.

A. Task Model

We would like to model the behavior of tasks over time, including the evolution of any given task over time, and the relationship among different tasks in the long run. We begin our analysis with a generic model; a specific example is given in Section V.

An instance of a task represents a single continuous period that the task is in service, which is called the *service time*. An instance of task m has an average duration of τ_1^m slots, and the time between them has average duration of τ_0^m slots. We assume that $L \ll \min\{\tau_1^m, \tau_0^m\}$, $\forall m = 1, 2, \ldots, M$, which ensures that the probability that any task changes its status during a frame is negligible. Furthermore, we also assume that the current service time is known to the IoT EMS and gateway every time it starts. Therefore, at the beginning of each frame, the IoT EMS only needs to predict the starting time of each task in the current frame and then wake up/sleep the appropriate objects at the appropriate time.

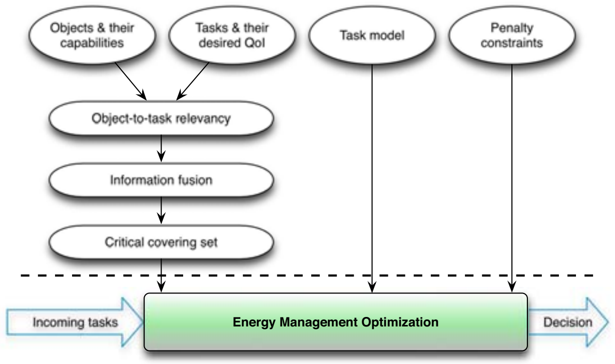

Fig. 2. System flow of the proposed energy management framework.

Let t denote the time of a decision point, i.e., the beginning of a certain frame, and Δ_t^m, $m \in \mathcal{M}$ be the starting time of the task m following time t. Let $\underline{\Delta}_t = \{\Delta_t^m, \forall m \in \mathcal{M}\}$. Specifically, Δ_t^m is defined in such a way that, if $\Delta_t^m = 0$, task m is already in service at time t, if $0 < \Delta_t^m < L$, $\Delta_t^m + t$ is the starting time of task m, and if $\Delta_t^m = L$, task m will not start in the current frame. Clearly, $|\underline{\Delta}_t| = ML$.

Let H_t^m describe the history information of task m up to time t, i.e., the sequence of times that task m has changed its state, and $\underline{\mathbf{H}}_t = \{H_t^m, m \in \mathcal{M}\}$. As a generic model, we assume that the conditional probability of $\underline{\Delta}_t$ given $\underline{\mathbf{H}}_t$ is known and denoted by Λ, i.e.,

$$\Lambda = \mathbf{Pr}\{\underline{\Delta}_t | \underline{\mathbf{H}}_t\}. \tag{1}$$

This generic model is used primarily to aid our analysis, and in general, the computation complexity can increase exponentially with M.

B. System Flow

Fig. 2 illustrates the procedure for the proposed application-layer energy management during one frame, which can be summarized as follows:

1) At the object deployment stage, compute the critical covering sets of each task based upon the object capabilities and the desired QoI of the tasks (see Section III).
2) At the beginning of a frame, the gateway makes a decision on *when* to activate/deactivate which object and for *how long* in the current frame based on the task model and the penalty constraints, and sends the control message back to each object (see Section IV).
3) During a frame, each object follows its predetermined waking-up schedule without further communications with the gateway until the next frame.

III. QoI-Aware Object-to-Task Relevancy

In [8], the 5WH principle was proposed to summarize the information needs of a task and the sensing capabilities of network resource, and in [9], the spatial relevancy of the provided information was introduced along with a way to measure it. Inspired by the above work, we propose the relevancy of an object to a task as the degree to which the object can satisfy the task's QoI requirements. Specifically, we define:

$$r_{nm} = f\Big(\underline{c}^n, \underline{q}^m\Big) \in [0,1], \quad \forall n \in \mathcal{N}, m \in \mathcal{M}, \tag{2}$$

where r_{nm} denotes the relevancy of object n to task m, and $f(\cdot)$ is a generic relevancy function that takes value in $[0,1]$ by definition. A specific example of f is given in Section V.

We define an object *irrelevant* to a task if and only if its relevancy to the task is 0. Examples of irrelevant objects to a task include objects whose sensing region have no overlap with the desired service region of a task, and objects that cannot provide the type of information the task requires, such as an object providing temperature readings to an air pressure related task. On the other hand, for the coverage requirement of a task, we say an object *covers* a task if and only if the computed relevancy is 1. By definition, an object covers a task if and only if it can individually satisfy the desired QoI of the task. In an IoT sensory environment, the retrieved information of a single relevant object usually cannot satisfy all QoI requirements of a task (with relevancy value varies between 0 and 1). Therefore, to fully satisfy a task's QoI requirement, fusing information collected from multiple coordinating objects is needed.

A. Information Fusion

Some QoI requirements, like the coverage of a region, can be achieved by certain fusion algorithm (function) even if no individual object can achieve it. The authors in [9] proposed to select a number of providers that cumulatively provide the most relevant information using an abstract, scalar-valued representation of QoI. While similar in principle, here we consider a more general way to accommodate a vector-valued QoI in information fusion. Specifically, let $Z_n, \forall n \in \mathcal{N}$, be the retrieved information of object n. We assume that information fusion is always beneficial, i.e., the information produced by the fusion of information from the objects in a set $\mathcal{S} \subseteq \mathcal{N}$ is always more relevant than the information offered by one single object alone in $\mathcal{S}$. For ease of presentation, we use $g(\cdot)$ for the generic fusion function, and clearly, g should take a variant number of single object "capabilities" and output an aggregated capability in all aspects. Denoting the capability of a subset $\mathcal{S}$ of objects by $\underline{c}^{\mathcal{S}}$, we have:

$$\underline{c}^{\mathcal{S}} = g\Big(\{\underline{c}^n | n \in \mathcal{S}\}\Big). \tag{3}$$

Then, the relevancy of a subset of objects to a task can be defined in the same way as that of a single object to a task based upon their aggregated sensing capability, i.e.,

$$r_{\mathcal{S},m} = f\Big(\underline{c}^{\mathcal{S}}, \underline{q}^m\Big), \quad \forall \mathcal{S} \subseteq \mathcal{N}, m \in \mathcal{M}. \tag{4}$$

B. Critical Covering Set

We define the *critical covering set* (CCS) of a task as a set of objects whose aggregated object-to-task relevancy always achieves 1; and if the retrieved information of any object is lost, the aggregated relevancy will drop below 1. It is worth noting that there is finite probability that the smart objects may not be able to cover the entire area of interest when randomly deployed. Furthermore, the desired QoI of certain tasks may be too demanding that even multiple collaborated objects could not satisfy it. Therefore, it is possible that a task has no CCS. In this paper, we assume that there is sufficient density of deployed objects to always guarantee the existence of at least one CCS for each task, leaving other cases for future studies. However, it is worth noting that the system performance metric we defined in Section IV-B also fits the case in which there exists no CCS for certain tasks. For ease for presentation, let $\mathbb{S}^m, \forall m \in \mathcal{M}$, be the set of all CCSs for task m and $\underline{\mathbb{S}} = \{\mathbb{S}^m\}$ the collection of all these sets.

IV. QoI-Aware Long-Term Energy Management

As discussed earlier, in order to fully exploit the energy-efficiency in an IoT sensory environment without sacrificing the QoI delivered to a task, only (a) the irrelevant objects, i.e., objects that are not relevant to any future incoming tasks, and (b) the redundant objects, i.e., objects that are not critical at all to any tasks, are allowed to be switched to the sleeping mode permanently (or OFF). In this section, we propose a framework to control the duty-cycling of these objects based upon the task model outlined in Section II-A.

A. Duty-Cycling of Objects

We assume that there are only two power consumption levels for each object $n \in \mathcal{N}$: (a) ϵ_n during the active sensing model (or ON); and (b) for simplicity, 0 during the sleeping mode (or OFF, and this number is relatively very small if compared with ϵ_n in reality). The duty-cycle of an object is defined as the fraction of time that the object is ON, i.e., $T^n_{\text{ON}}(T)/T$, $\forall n \in \mathcal{N}$, where $T^n_{\text{ON}}(T)$ is the aggregation of the ON times during the lifetime T. Note that here we express the aggregated ON time as a function of T to explicitly describe its dependency on the lifetime T. However, this straightforward definition of duty-cycle does not directly reflect the energy spent while switching between the two modes. Therefore, we propose a *generalized duty cycle* to explicitly incorporate the extra energy penalty paid each time the objects switch modes. Specifically, let E_n denote the energy consumed each time the objects switch modes, and $N^n_s(T)$ the number of switches object n makes up to time T, the generalized duty cycle η^n of object n is defined as

$$\eta^n = \frac{E_n}{\varepsilon_n} \cdot \frac{N^n_s(T)}{T} + \frac{T^n_{\text{ON}}(T)}{T}, \quad \forall n \in \mathcal{N}. \tag{5}$$

The goal of energy management is to minimize the (generalized) object duty cycle in an IoT sensory environment, without sacrificing the QoI levels attained. At the beginning of each frame, the IoT EMS informs the gateway on the decisions as when to turn ON/OFF objects in the current frame. For simplicity, we assume an object keeps ON in a frame after it is waken up. Let $\underline{A}_t = \{a^n_t\},\ 0 \le a^n_t \le L,\ n \in \mathcal{N}$, denote the set of actions for all objects when decisions are made at a decision point t. Then, $\underline{A}_t$ specifies the scheduled waking-up time for every object. Specifically, a^n_t is defined in such a way that, if $0 \le a^n_t < L$, $a^n_t + t$ is the next wake-up time for task n, and if $a^n_t = L$, object n will remain asleep in the current frame. Clearly, the cardinality of the decision space of $\underline{A}_t$ is NL.

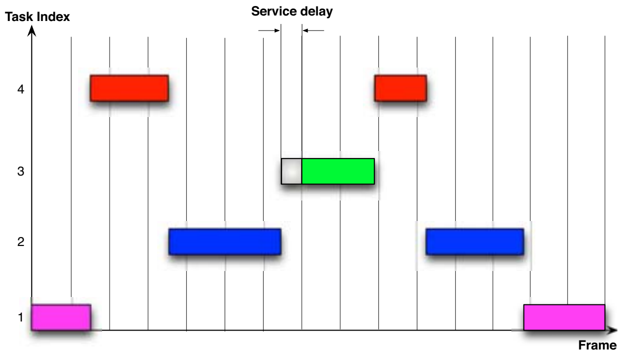

Fig. 3. An illustrative example of service delay under the proposed task model, where four tasks follows different arrival and departure statistics, and during one single emergence, its maximum allowed service delay is shown. Task 1, 2 4 have two instances and task 3 has only one instance in this example.

B. Delay Penalty for Tasks

It is likely that task instances may not be serviced immediately following their arrival, i.e., when a task starts again, an object's duty cycle is controlled periodically by the IoT EMS. As a result, it is possible that no active CCS of a task exists when it starts. Therefore, the task may have to wait for the next frame when the EMS informs the gateway to wake up a CCS for its service. This service delay may be tolerable for elastic traffic (e.g., a few seconds delay for reporting the water quality levels are highly likely tolerable), and thus we define it as:

$$d^m_i = t^m_{i,s} - t^m_{i,a}, \quad \forall m \in \mathcal{M},\ i \in \mathbb{N}^+, \tag{6}$$

where $\mathbb{N}^+$ is the set of non-negative integers, $t^m_{i,a}$, $t^m_{i,s}$ and d^m_i are the task starting time, the service start time and the service delay of task m for its i-th instance, respectively. An example of service delay is illustrated in Fig. 3.

In order to characterize the performance degradation of tasks by the potential service delay, we propose a delay-induced, non-decreasing penalty function:

$$h(\cdot) : d^m_i \to \zeta^m_i, \tag{7}$$

where ζ^m_i denotes the penalty for the i-th arrival of task m. By specifying the delay requirement D^m, an example of $h(\cdot)$ can take the following form:

$$h(d^m_i) = \begin{cases} \nu^{d^m_i}, & \text{if } 0 < d^m_i < D^m, \\ \nu^{D^m}, & \text{if } d^m_i \ge D^m, \\ 0, & \text{otherwise}, \end{cases} \tag{8}$$

$\forall i \in \mathbb{N}^+, m \in \mathcal{M}$, and parameter ν is chosen to be $\nu \ge 1$. Particularly, if $\nu = 1$, we always have $h(d^m_i) = 1$, or: the associated task m is delay-sensitive application.

Finally, the *average penalty* of task m up to its I-th instance can be computed as:

$$\zeta^m = \frac{1}{I} \sum_{i=1}^{I} \zeta^m_i,\ \forall m \in \mathcal{M},\ I \in \mathbb{N}^+, \tag{9}$$

and if $\nu = 1$, the average penalty is reduced to the probability that the task is not served immediately when it starts (during the life time of the monitoring system).

C. Problem Formulation

At the beginning of each frame, the EMS informs the gateway of decisions made on the energy consumption state of each object $n \in \mathcal{N}$, i.e., which set of objects should be waken up for task service in the current frame, and which set of objects are allowed to be turned OFF, given the historical task service information and the historical object activity information that is denoted by $\underline{\mathbf{G}}_t$. Therefore, a decision policy π is defined as a mapping from $\underline{\mathbf{G}}_t$ and $\underline{\mathbf{H}}_t$ to $\underline{A}_t$, given the task model and the CCS information:

$$\underline{A}_t = \pi(\underline{\mathbf{G}}_t, \underline{\mathbf{H}}_t | \Lambda, \mathbb{S}). \tag{10}$$

The goal of EMS algorithm is to find the optimal decision policy π^* that optimizes the object duty-cycles under the penalty constraints for tasks. We propose two performance metrics to describe the system performance, and then formulate two corresponding optimization problems.

1) Minimize the maximum duty cycle: As a collection of $\mathcal{N}$ objects comprise the IoT sensory environments, the optimization of one single object duty-cycle does not represent the overall optimum, and this model starts from the overall IoT lifetime perspective that aims at providing a degree of fairness among all objects (or in other words, the usage of all objects are relatively comparable). The optimization problem is:

$$\begin{aligned} &\underset{\pi}{\textit{minimize:}} \quad \eta_{\max} = \max_{n \in \mathcal{N}} \eta^n \\ &\textit{subject to:} \quad \zeta^m \le \xi_m, \quad \forall m \in \mathcal{M}, \end{aligned} \tag{11}$$

where the constraint is the average penalty for service delay should be smaller or equal than the tolerable value $\xi_m, \quad \forall m \in \mathcal{M}$.

2) Minimize weighted average duty cycle: This model aims at minimizing the average cost of the entire IoT sensory environment, where in reality smart objects execute different tasks and thus of different importance to the lifetime of the system, which are then defined as the weights. The optimization problem is:

$$\begin{aligned} &\underset{\pi}{\textit{minimize:}} \quad \overline{\eta} = \sum_{n \in \mathcal{N}} \beta_n \eta^n, \\ &\textit{subject to:} \quad \zeta^m \le \xi_m, \quad \forall m \in \mathcal{M}, \end{aligned} \tag{12}$$

where β_n are weight factors with $0 \le \beta_n \le 1$, $n \in \mathcal{N}$, and $\sum_{n\in\mathcal{N}} \beta_n = 1$; for important objects EMS will assign higher β_n accordingly.

It is worth noting that another possible constraint condition for the optimization problem (11) and (12) could be the weighted average penalty for all tasks, i.e.,

$$\overline{\zeta} = \frac{1}{M} \sum_{m=1}^{M} \alpha_m \zeta^m \le \xi, \tag{13}$$

where α_m are weight factors with $0 \le \alpha_m \le 1$, $m \in \mathcal{M}$, and $\sum_{m\in\mathcal{M}} \alpha_m = 1$.

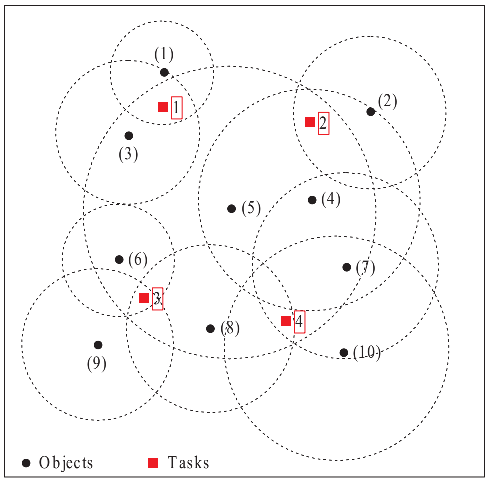

Fig. 4. An illustrative example of the reservoir plane graph to monitor the water quality of four locations (as red square), where ten randomly deployed objects with certain sensing range are shown (as black dots).

V. A Case Study

In this section, we show an example of our methodology. We illustrate this by assuming a monitoring IoT application, such as using (randomly deployed) pollutant-sensing objects with certain sensing range to measure the water quality of certain location in a reservoir supplying water to city dwellers. In this section, we present the system pertinent solutions and algorithms, and in the next section we show results from simulations that we conducted.

A. System Model

In our water quality monitoring system, each object, or in particular the *sensors* with pollution level monitoring capability, is randomly deployed and its spatial coverage follows a classic disk model. To close the real scenario, we assume that the sensory data within the sensing region is corrupted by noise during measurement and/or transmissions. Fig. 4 shows an illustrative example of the reservoir plane graph for sensor and task deployment.

In this example, accuracy is the only QoI requirement (however with multiple metrics) we considered, and we first define its probabilistic model as:

$$\mathbf{Pr}\Big\{|Z_t^m - z| \ge \delta_m z\Big\} \le \epsilon_m, \ \forall m \in \mathcal{M}, \tag{14}$$

where the random variable Z_t^m is the object-retrieved information for task m at time t, and z is the actual but unknown information, i.e., the ground truth. Analogously to the desired QoI functions in [9], we define $\underline{q}^m$ as:

$$\underline{q}^m = \Big\{Y_m, (\delta_m, \epsilon_m)\Big\}, \ \forall m \in \mathcal{M}, \tag{15}$$

where Y_m and $\{\delta_m, \epsilon_m\}$ are the geographical location and accuracy requirement of task m, respectively.

On the other hand, the capability of object n, i.e., $\underline{c}^n$, can be defined as:

$$\underline{c}^n = \Big\{(X_n, r_n), \gamma_n\Big\}, \ \forall n \in \mathcal{N}, \tag{16}$$

where X_n is the location of the object and r_n is its sensing radius. We model the measurement noise as additive white Gaussian noise (AWGN) with variance γ_n for object n. A object-to-task relevancy function for this model is

$$\begin{aligned} f(\underline{c}^n, \underline{q}^m) &= f(X_n, r_n, \gamma_n, Y_m, \delta_m, \epsilon_m) \\ &= \mathbf{1}\{\text{dist}(X_n, Y_m) \le r_n\} \cdot \max\left\{\frac{\epsilon_m}{\mathbf{Pr}\{|Z_t^n - z| \ge \delta_m z\}}, 1\right\} \\ &= \mathbf{1}\{\text{dist}(X_n, Y_m) \le r_n\} \cdot \max\left\{\frac{2\epsilon_m}{Q(\frac{\delta_m}{\sqrt{\gamma_n}})}, 1\right\}, \end{aligned} \quad (17)$$

$\forall n \in \mathcal{N}, m \in \mathcal{M}$, where $\mathbf{1}\{\textit{statement}\}$ is the indicator function that takes value 1 if the *statement* is true and 0 otherwise. Let dist(X_n, Y_m) be the Euclidean distance between two points, the random variable Z_t^n be the retrieved information of object n at time t, and $Q\{\cdot\}$ be the tail probability of the standard normal distribution.

If task m is serviced solely by object n, then $Z_t^m = Z_t^n$; otherwise, if it is serviced by a subset $\mathcal{S}$ of objects, then $Z_t^m = Z_t^{\mathcal{S}}$. A possible information fusion algorithm of relevant objects in this case can be:

$$Z_t^{\mathcal{S}} = \arg\min_{z'} \frac{1}{|S'|} \sum_{n\in\mathcal{S}} \frac{1}{\gamma_n} \left| Z_t^n - z' \right|^2 = \frac{\sum_{n\in\mathcal{S}} \frac{Z_t^n}{\gamma_n}}{\sum_{n\in\mathcal{S}} \frac{1}{\gamma_n}}. \quad (18)$$

The right hand of (18) is a specific example of the fusion function $g(\cdot)$ we defined in Section III-A. Specially, if all γ_n are equal and $Z_t^n \sim \mathsf{N}(1, 1/\gamma)$, the fused information of a group of N' relevant objects is the average of the individual ones and $Z_t^{\mathcal{S}} \sim \mathsf{N}(1, 1/(N'\gamma))$, where $\mathsf{N}(\mu, \sigma^2)$ is a Gaussian distribution with mean μ and variance σ^2. Based on the above fusion algorithm, CCSs of every task can be computed during the object deployment stage, and used in the online duty-cycling control.

We assume a simple penalty function for the task which fits both delay-sensitive and delay-insensitive tasks, i.e.,

$$h(d_i^m) = \begin{cases} 0, & \text{if } 0 \le d_i^m \le D, \\ 1, & \text{if } d_i^m > D, \end{cases} \quad (19)$$

$\forall m \in \mathcal{M}$, $i \in \mathbb{N}^+$ and D is the delay requirement for all tasks. Specifically, if $D = 0$, the task is delay-sensitive, and if $D > 0$, the task is delay-insensitive. Then, average penalty ζ^m in this case has the meaning of the *average probability* that task m's delay requirement is not satisfied over time.

We model the evolution of tasks as a (discrete) semi-Markov process. A semi-Markov process is a stochastic process which moves from one state to another, with the successive states visited forming a Markov chain, and that the process stays in a given state a random length of time (holding time). The state space of a semi-Markov process is countable and the distribution function of the holding times may depend on the current state as well as on the one to be visited next [10]. When modeling the task evolution by a semi-Markov model, the tasks are treated as the states. The behavior of the tasks can be summarized in the following three aspects:

- There is one, and only one task in service at any time slot (as illustrated in Fig. 3). And because of this, we call it the *exclusive task model*;
- A new task starts immediately after a current task ends with certain "task transition" probability;
- The service time of a task is known at the time it starts.

We denote $\mathbf{P} = \{p_{k,m}\}$ as the task transition probability from task k to task m.

B. A Greedy Algorithm

The optimization problems in (11) and (12) are generally NP-hard and their optimal solution are difficult to find without an exhaustive search. In this paper, we propose a greedy algorithm for optimization problem (12). The algorithm is greedy in that at any decision point, it chooses the action that leads to the least *marginal increment* in $\overline{\eta}$.

For ease of presentation, suppose the system starts at $t = 0$. Denote $t = iL$ as the beginning of the i-th frame, where $i \in \mathbb{N}$. Denote η_t^n as the *runtime* generalized duty cycle of object n up to time t. η_t^n can be updated recursively by

$$\begin{aligned} \eta_t^n = \frac{1}{t}\Bigg[&\frac{E_n}{\varepsilon_n}\Big(N_s^n(t) - N_s^n(t-L)\Big) + \Big(T_{\text{ON}}^n(t) - T_{\text{ON}}^n(t-L)\Big) \\ &+ \eta_{t-L}^n \cdot (t-L)\Bigg], \quad t = iL, i \in \mathbb{N}^+, \end{aligned} \quad (20)$$

with η_0^n defined to be zero. Note that $N_s^n(t) - N_s^n(t-L)$ and $T_{\text{ON}}^n(t) - T_{\text{ON}}^n(t-L)$ are the number of state switches and the aggregated ON time between time $t-L$ and time t for object n, respectively. We define the marginal increase in the normalized energy consumption of object n between time t and $t+L$ as:

$$\Theta_t^n \triangleq \frac{E_n}{\varepsilon_n}\Big(N_s^n(t+L) - N_s^n(t)\Big) + \Big(T_{\text{ON}}^n(t+L) - T_{\text{ON}}^n(t)\Big), \quad (21)$$

and clearly, we have:

$$\eta_t^n = \frac{1}{t}\Big(\Theta_{t-L}^n + \eta_{t-L}^n \cdot (t-L)\Big), \quad \forall n \in \mathcal{N}. \quad (22)$$

Further, define the weighted average marginal increase in the normalized energy consumption of all objects between time t and $t+L$ as

$$\overline{\Theta}_t \triangleq \sum_{n\in\mathcal{N}} \beta_n \Theta_t^n. \quad (23)$$

At the i-th decision point, the gateway needs to predict the task activities in the i-th frame and prepare the objects accordingly. Rather than a global algorithm that minimizes $\overline{\eta}$ throughout the lifetime of the IoT sensory environment, we specify an algorithm that minimizes $\overline{\Theta}_t$ at each decision point. It is fairly reasonable that minimizing $\overline{\Theta}_t$ at every decision point helps to reduce the overall average duty cycle.

On the other hand, consider the penalty constraint for task m, which we rewrite in the following as

$$\zeta^m = \frac{1}{I}\sum_{i=1}^{I} \zeta_i^m \le \xi_m, \quad \forall m \in \mathcal{M}. \quad (24)$$

If $\zeta_i^m \leq \xi_m$ is satisfied for all $i = 1, 2, \ldots, I$, the penalty constraint condition is clearly satisfied as well. This implies a greedy solution from the penalty constraint side: for a given task, prepare the objects in such a way that the expected penalty does not exceed the prescribed penalty constraint. Specifically, define $P_i^m(t)$ as the probability that task m's i-th starting time is t, and $Q^m(t)$ the probability that at least one CCS of task m exists at time t. Clearly,

$$\mathbb{E}[\zeta_i^m] = \sum_{t=0}^{T} P_i^m(t)\Big(1 - Q^m(t)\Big), \quad \forall m \in \mathcal{M}, \tag{25}$$

where T can be viewed as the system lifetime. Note that $P_i^m(t)$ is zero almost everywhere. To see this, $P_i^m(t) = 0$ if: (a) t is not a task transition time, (b) either less than $i-1$ or greater than $i+1$ instances of task m have occurred. In other words, $P_i^m(t)$ takes non-zero value only at the time of task transition and task m is expecting its i-th instance. Therefore, the above summation is easy to compute.

As an illustrative example, the exclusive task model is simple in that the gateway knows exactly the time when the current task ends and the next task starts, whereas which specific task succeeds the current one is uncertain. Therefore, if there is no transition between tasks in a frame, the system only needs to keep awake the CCS of the current task that leads to the least $\overline{\Theta}_t$ and set the other objects to sleep. Otherwise, if a task transition is bound to happen in a frame, the system has to wake up the corresponding objects to make preparation for all possible succeeding tasks under their specific penalty constraints. Suppose the current task will end at time $t'-1$ and a new task will start at time t, where $t \in [iL, (i+1)L)$. Our greedy algorithm at the i-th decision point is the solution to the following optimization problem

$$\begin{aligned} &\underset{\pi}{\textit{minimize:}} && \overline{\Theta}_{iL} \\ &\textit{subject to:} && P^m(t')(1 - Q^m(t')) \leq \xi_m, \quad \forall m \in \mathcal{M}. \end{aligned} \tag{26}$$

where $\overline{\Theta}_{iL}$ is defined in (21) and (23). This optimization problem is easy to solve. To see this, the constraint condition requires that

$$Q^m(t') \geq 1 - \min\left\{1, \frac{\xi_m}{P^m(t')}\right\}, \tag{27}$$

where $P^m(t')$ is the transition probability from the current task to task m. Essentially, (27) specifies the minimum required probability of existence of CCSs for each task at the task transition time t'. Therefore, the gateway can determine whether to wake up a CCS for each task according to $Q^m(t')$, either jointly or separately. Clearly a decision that considers all tasks jointly will do no worse, and in all likelihood outperform, the separate per task decision. However, this will induce further computational complexity, especially when M is large. Therefore, as another "degree of greediness", we assume the gateway makes decision separately for each task. After the decision on making preparation for which group of tasks is made, the gateway chooses and schedules the wake-up times for a subset of objects that can cover that selected group of tasks and induces the minimum increase in the marginal energy consumption $\overline{\Theta}_{iL}$. In the $(i+1)$-th frame, which task follows the previous task is already known and all the irrelevant objects prepared for the possible occurrence of other tasks can be sent to sleep.

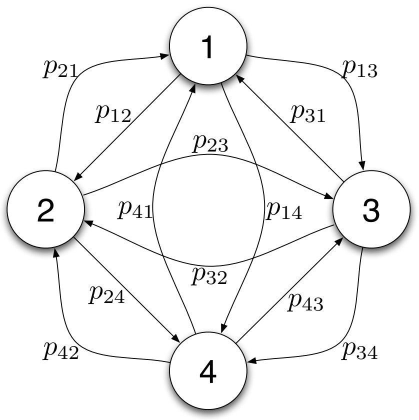

Fig. 6. Markov chain model for task dynamic involvements.

The algorithm can be summarized in the following steps:

1) At the beginning of each frame, shut down any object that is not critical to the current task, if there exist such objects;
2) If no task transition is going to happen in the current frame, keep the current status of each object until the next frame;
3) If task transition is bound to happen in the current frame, for each task, compute the minimum required probability of existence of a CCS based on the delay-induced penalty constraint by (27), and determine (by random tests) whether to make preparation for that task according to the derived probability. At the time of task transition, wake up a subset of objects that critically covers all the tasks to be prepared for, yet induces the minimum increase in the marginal energy consumption.

The algorithm is greedy in three aspects:

1) The algorithm satisfies the penalty constraints every time task transitions happen;
2) The algorithm minimizes the marginal increase in energy consumption at every decision point;
3) The algorithm makes decision on whether to prepare for the possible occurrence of each task *separately*.

Note that we can revise the greedy algorithm in such a way that it makes decision on whether to prepare for the possible occurrence of each task *jointly*. Compared to the original greedy algorithm whose computation complexity increases linearly with M, the computation complexity of the *revised greedy algorithm* increases exponentially with M and it is therefore much more computationally challenging. In order to show the difference, we plot the results for both greedy algorithms in Section VI for a small M.

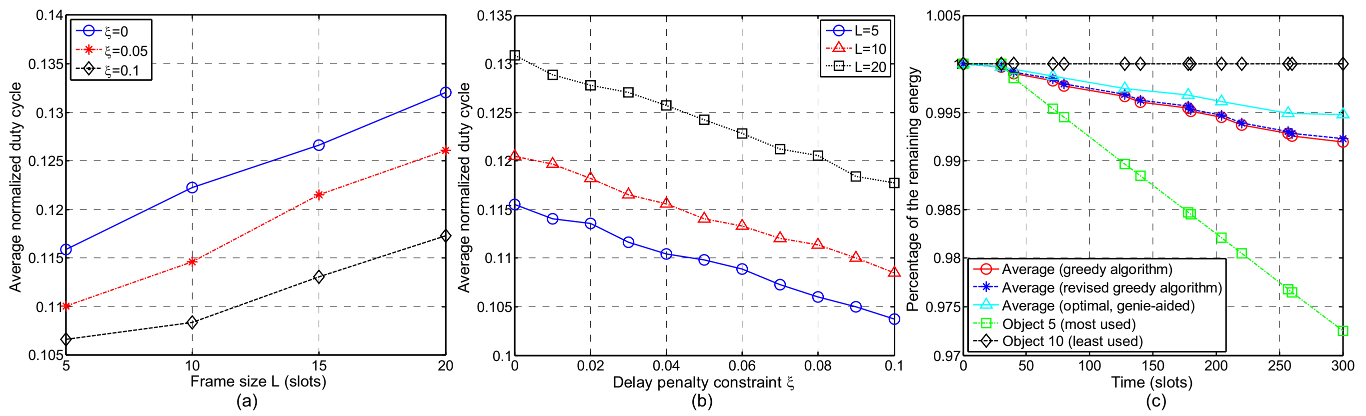

Fig. 5. Simulation results for: (a) average normalized duty cycle $\overline{\eta}$ with different frame size L, (b) Average normalized duty cycle $\overline{\eta}$ with different penalty constraint ξ, and (c) The run-time remaining energy of the system corresponding to the task evolution in Fig. 6.

VI. Simulation Results

Our simulation setup is based on the water quality monitoring system discussed above. Specifically, the system consists of $N = 10$ objects and $M = 4$ tasks, as illustrated in Fig. 4. The capabilities (exclusive of sensing radius which is illustrated in Fig. 4) of all objects are: $\gamma_1 = \gamma_6 = 2$, $\gamma_2 = \gamma_3 = \gamma_4 = \gamma_8 = 1$, $\gamma_5 = \gamma_9 = \gamma_{10} = 0.7$ and $\gamma_7 = 0.5$. The desired QoI of all tasks are: $\epsilon_m = 0.1$, $\delta_m = 1, \forall m \in \mathcal{M}$. Moreover, $E_n = 5$ and $\epsilon_n = 1, \forall n \in \mathcal{N}$ and the initial energy reserve of each object is set as 10,000. The service time of each task follows identical geometric distribution with average duration 50. The task transition probability matrix is given by:

$$\mathbf{P} = \begin{pmatrix} 0 & 1/10 & 2/5 & 1/2 \\ 1/5 & 0 & 3/5 & 1/5 \\ 1/3 & 1/3 & 0 & 1/3 \\ 4/5 & 1/10 & 1/10 & 0 \end{pmatrix}. \qquad (28)$$

The object-to-task relevancy and the CCSs can therefore be computed and the result is listed as:

$$\mathbb{S}^1 = \Big\{\{1\}, \{3,5\}\Big\},$$
$$\mathbb{S}^2 = \Big\{\{2,4\}, \{2,5\}, \{4,5\}\Big\},$$
$$\mathbb{S}^3 = \Big\{\{6\}, \{5,8\}, \{8,9\}\Big\},$$
$$\mathbb{S}^4 = \Big\{\{5,8\}, \{8,10\}, \{5,7,10\}\Big\}.$$

We consider the optimization problem in (12) with $\beta_n = 1/N, \forall n \in \mathcal{N}$, and use the greedy algorithm we derived in Section V-B. We run several simulations for different system setups.

In Fig. 5(a), we plot the generalized duty cycle with different frame size L. As shown in the simulation result, for fixed penalty constraint, the duty cycle increases with the frame size L. This is because every time a task transition happens, more than eventually necessary objects have to be waken up for the possible starts of certain group of tasks, though in fact only one task actually starts. These unnecessary objects will stay awake until the next decision point when they can be turned OFF by the gateway. Clearly, the wasted ON times of objects increase linearly with the frame length. In Fig. 5(b), we plot the generalized duty cycle with different penalty constraints. Clearly seen, for fixed frame size, the duty cycle decreases with the penalty constraint, as higher tolerance in service delay allows the objects to spend more time in the sleeping mode.

Fig. 5(c) illustrates the energy depletion process corresponding to the illustrative example of task evolution shown in Fig. 6 for $\xi = 0$, $m \in \mathcal{M}$ and $L = 20$. Besides the proposed greedy algorithm and its revised version, we also plot the result for a genie-aided case in which the gateway knows exactly which task succeeds the current one. The (average) slope of a curve represents the (average) energy depletion rate. From the simulation result, we can see that the revised greedy algorithm achieves a better system performance than the basic greedy algorithm (at the expense of more computation complexity). We also observe the gap between the greedy algorithm and the genie-aided optimal scheduling, indicating a potential improvement in future algorithm design. Also plotted in the figure are the energy depletion process for two extremes: the least used object has never been used, while on the other hand, the most used object has been active almost all the time. This is to be expected since we are minimizing the average duty cycle without considering fairness among objects.

VII. Related Work

QoI is proposed recently to judge how the retrieved information is fit-for-use for a particular task [8]. Inspired by the notion of QoI, the authors in [11] propose a QoI satisfaction index to describe the degree of QoI satisfaction the tasks received from the wireless sensor network (WSN), as well as a QoI network capacity to describe the marginal ability of the whole WSN when a new task with associated QoI requirements arrives. Based on these, the authors proposed an adaptive admission control to optimally accommodate the QoI requirements of all incoming tasks by treating the whole WSN as a "black box". Our work in this paper has built upon and expanded [8], [11], allowing dynamically controlling the energy consumption state of each object by explicitly taking into account the desired QoI in the long run.

The authors in [12] study the attained QoI performance for object tracking applications with energy constraints. The work focuses on a duty-cycled network with random wake-up schedules for different sensor nodes and studies the trade-off between tracking accuracy and response latency that the system observe. Our paper, however, focus on information collection and processing in the presence of energy constraints as well as desired QoI performance for IoT applications.

In [13], the authors propose a node-scheduling scheme to reduce the system overall energy consumption by turning off redundant nodes, while preserving the sensing coverage for the entire WSN at a given time. To calculate the sponsored coverage, only the sensing region (i.e., the radius) is taken into account in [13]. This essentially corresponds to the spatial requirement of the 5WH QoI requirement structure proposed in [8]. In this paper, we take every aspect in the desired QoI into account, which we then reflect in the critical covering set for each task. Another difference is that the authors in [13] sought a group of objects that can cover the entire region of interests at any given time, while we seek the critical covering set only for the on-going tasks based upon the multi-task oriented traffic model.

Finally, we note that there has been plenty of work on MAC layer protocol design for WSNs focusing on minimizing the energy consumption in order to achieve long system lifetime, such as S-MAC [14] and T-MAC [15]. In contrast to this work, our proposal is a system-level management operation and not a communications protocol; and more importantly, this proposal can work with systems that engage to MAC level energy conservation as well.

VIII. Conclusions and Future Work

In this paper, an efficient energy management framework to provide satisfactory QoI experience in IoT sensory environments is studied. Contrary to past efforts, our proposal is transparent and compatible to lower protocols in use, and preserving energy-efficiency in the long run without sacrificing any attained QoI levels. Specifically, we first introduce the new concept of QoI-aware "object-to-task relevancy" to explicitly consider the sensing capabilities offered by an object to the IoT sensory environments, and QoI requirements required by a task. Second, we propose a novel concept of the "critical covering set" of any given task in selecting the objects to service a task over time. Third, energy management decision is made dynamically at runtime, as the optimum for long-term traffic statistics under the constraint of the service delay. Finally, an extensive case study based on utilizing the object networks to perform water level monitoring is given to demonstrate the ideas and algorithms proposed in this paper, and a simulation is made to show the performance of the proposed algorithms.

As for the future work, we are interested in more realistic task models to better describe the relationship between tasks other than the exclusive task model based on the semi-Markov process. We would like to study how the system learns and maintains the task model and how sensitive the corresponding algorithms are to the accuracy of the model. Meanwhile, this paper mainly focus on a centralized control framework which prohibits the scalability as the number of objects increases, and therefore we are interested in extending the idea to a distributed control framework. Finally, we are planning to have a real world deployment of our proposal as a proof-of-concept.

Acknowledgment

Research was sponsored in part by US Army Research laboratory and the UK Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the U.S. Government, the UK Ministry of Defense, or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

[1] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Comput. Netw.*, 54:2787–2805, Oct. 2010.
[2] J. Zheng, D. Simplot-Ryl, C. Bisdikian, and H. T. Mouftah. The internet of things. *special topic in IEEE Communications Mag.*, Nove 2011.
[3] Microstrain, inc. http://www.microstrain.com/.
[4] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Comm. Mag.*, 40(8):102–114, Aug 2002.
[5] C. Bisdikian, L. M. Kaplan, M. B. Srivastava, D. J. Thornley, D. Verma, and R. I. Young. Building principles for a quality of information specification for sensor information. In *IEEE FUSION 2009*, July.
[6] Richard Y. Wang and Diane M. Strong. Beyond accuracy: what data quality means to data consumers. *J. Manage. Inf. Syst.*, 12(4):5–33, 1996.
[7] M. E. Johnson and K. C. Chang. Quality of information for data fusion in net centric publish and subscribe architectures. In *IEEE FUSION'05*, July 2005.
[8] C. Bisdikian, J. Branch, K. K. Leung, and R. I. Young. A letter soup for the quality of information in sensor networks. In *IEEE Percom'09*, Galveston, TX, USA, March 2009.
[9] G. Tychogiorgos and C. Bisdikian. Seeking provides of relevant sensory information. In *IEEE MDM'11*, June 2011.
[10] R. Pyke. Markov renewal processes: definitions and preliminary properties. *Ann. Math. Statist*, 32:1231 – 1242, 1961.
[11] C. H. Liu, C. Bisdikian, J. W. Branch, and K. K. Leung. QoI-Aware wireless sensor network management for dynamic multi-task operations. In *IEEE SECON'10*, Boston, MA, USA, June 2010.
[12] S. Zahedi, M. B. Srivastava, C. Bisdikian, and L. M. Kaplan. Quality tradeoffs in object tracking with duty-cycled sensor networks. In *IEEE RTSS'10*, pages 160 – 169, November 2010.
[13] D. Tian and N. D. Georganas. A coverage-preserving node scheduling scheme for large wireless sensor networks. In *ACM Int'l Workshop Wireless Sensor Networks and Applications*, Georgia, GA, USA, 2002.
[14] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks. *IEEE Commun. Mag.*, 4(44):115 – 121, April 2006.
[15] T. Dam and K. Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *ACM 1st Int'l Conf. on Embedded Networked Sensor Sys.*, October 2003.