An Adaptive Delay-Minimized Route Design for Wireless Sensor-Actuator Networks

Edith C.-H. Ngai* , Jiangchuan Liu† and Michael R. Lyu*
*Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong
Email: {chngai, lyu}@cse.cuhk.edu.hk
†School of Computing Science, Simon Fraser University, Vancouver, BC, Canada
Email: jcliu@cs.sfu.ca

***Abstract*— Wireless sensor-actuator networks (WSANs) have recently been suggested as an enhancement to the conventional sensor networks. The powerful and mobile actuators can patrol along different routes and communicate with the static sensor nodes. This work is motivated by applications in which the objective is to minimize the data collection time in a stochastic and dynamically changing sensing environment. This is a departure from the previous static and deterministic mobile element scheduling problems.**

In this paper, we propose PROUD, a probabilistic route design algorithm for wireless sensor-actuator networks. PROUD offers delay-minimized routes for actuators and adapts well to network dynamics and sensors with non-uniform weights. This is achieved through a probabilistic visiting scheme along pre-calculated routes. We present a distributed implementation for route calculation in PROUD and extend it to accommodate actuators with variable speeds. We also propose the Multi-Route Improvement and the Task Exchange algorithms for balancing load among actuators. Simulation results demonstrate that our algorithms can effectively reduce the overall data collection time in wireless sensor-actuator networks. It well adapts to network dynamics and evenly distributes the energy consumption of the actuators.

I. Introduction

Wireless sensor networks (WSNs) have been applied in a broad spectrum of applications ranging from environment monitoring, target tracking, to battlefield surveillance and chemical attack detection [1][2]. The asymmetric communication patterns from sensors to the sink, however, often overload the sensors close to the sinks and consequently reduce the network lifetime. Moreover, network partitions may occur in sensor networks, which make multihop communications impossible. To alleviate these problems, mobile elements, such as mobile sinks [3] or mobile relays [4], have been suggested for collecting data in WSNs. Actuators, which have much stronger computation and communication power than uni-purpose micro-sensors, have also been introduced [5][6]. In a Wireless Sensor-Actuator Network (WSAN), a mobile actuator can move around to cover the sensing field and interact with static sensors. Each static sensor has a limited buffer, which stores locally sensed data until some actuator approaches. It can then upload the data to the actuator with short-range communications and free its buffer.

The amount and frequency of data generation in the sensors are often non-uniform [7]. For example, the sensors with higher data generation rate or the locations with higher event occurring probability expect more frequent visits. More formally, there is a Route Design Problem (RDP) for the actuators to minimize their average inter-arrival time to the static sensors. While deterministic algorithms may work for static route calculations, in reality, the weight of sensors and event frequency are time varying, which call for an adaptive algorithm. A distributed and load-balanced implementation is also necessary for a large-scale sensor network with multiple actuators.

In this paper, we propose an effective and adaptive algorithm, called Probabilistic Route Design (PROUD) for route calculation. Our algorithm is based on a probabilistic model, which allows sensors to have shorter expected waiting times for data uploading if they are assigned with higher weights. Specifically, PROUD constructs a priori route that consists of all the sensor locations, while the actuators visit them probabilistically and cyclically according to their weights. It can adapt to network dynamics by updating the visiting probability easily without any re-calculation on the priori route. We show that this approach works for both small-scale and large-scale sensor-actuator networks, and present a distributed implementation. We further introduce enhancements to accommodate actuators with variable speeds, which works

for applications that demand bounded inter-arrival times. Finally, we devise a Multi-Route Improvement and a Task Exchange algorithm that enables load balancing. Our simulation results show that the proposed algorithm can effectively reduce the overall data collection time and evenly distribute the energy consumption among the actuators.

The remainder of the paper is organized as follow: The related work is presented in Section II, followed by an overview of the route design problem in Section III. The proposed algorithm, PROUD, is described in Section IV, and a distributed implementation is shown in Section V. In Section VI, we discuss possible enhancements for integrating actuators with variable speeds and balancing their workloads. Simulation results are presented in Section VII. Finally, Section VIII concludes the paper.

II. Related Work

Mobile elements have been proposed to carry data in wireless networks. Shah et al. [8] presented an architecture using moving entities (data mules) to collect sensor data. There have also been studies on mobile sinks with predictable and controllable movement patterns [9][10], and the optimal time schedule for locating sojourn points [3]. Apart from the above, Zhao et al. [11] proposed a message ferrying (MF) approach to address the network partition problem in sparse ad hoc networks. Luo et al. [4] investigated a joint mobility and routing algorithm with mobile relays to prolong the lifetime of wireless sensor networks. Gu et al. [12] proposed a partitioning-based algorithm to schedule the movement of mobile element (ME) to avoid buffer overflow in sensors and reduce the minimum required ME speed. Their solution was customized for an "eyes" topology, where the events are concentrated at certain locations. Solutions for sensor networks with general uniform distribution were left to be explored. Bisnik et al. [13] studied the problem of providing quality coverage using mobile sensors and analyzed the effect of controlled mobility on the fraction of events captured. Their focus is not on the route design problem. Recently, a route design algorithm for multiple ferries is proposed by Zhang et al. [14]. It considers a delay tolerant network scenario with point-to-point data transfer between sensors of uniform weights.

Our work is motivated by the above investigations. The key difference is that we focus on adaptive and distributed route design for multiple mobile components in WSANs, specifically, actuators moving along independent routes. Our objective is to minimize the average actuator inter-arrival time in dynamically changing environment. We also address issues regarding the non-unform weights of the static sensors. This is different from the previous mobile element scheduling problems where one seeks to minimize the waiting time in a static and deterministic environment.

A closely related problem to RDP is the vehicle routing problem (VRP), which considers scheduling vehicles stationed at a central facility to support customers with known demands, targeting at minimizing the total distance travelled [15]. There are a number of variations to VRP, including the Capacitated VRP (CPRV)[16] and VRP with time windows (VRPTW)[17]. While these investigations have studied the routes of mobile components, the unique features of the actuators and the heterogeneous sensor networks have yet to be explored.

III. Overview of The Route Design Problem

In this section, we describe the network model and give an overview to the route design problem.

A. Network Model

We consider a Wireless Sensor-Actuator Network (WSAN) consisting of M mobile actuators and N static sensors. Each of the sensors and actuators is equipped with a wireless transceiver. The actuators move in the sensing field along independent routes, at constant or variable speeds. Each static sensor has a limited buffer to accommodate locally sensed data. When an actuator approaches, the sensor can upload the data to the actuator and free its buffer. We also assume that the sensors have different weights, according to their data generation rates or event frequencies. Intuitively, sensors with higher weights expect shorter average actuator inter-arrival times. The weight of sensors may change dynamically according to the varying data generation rate and event frequency in the network.

B. The Route Design Problem

We consider the design of routes for multiple actuators in WSANs, and the objective is to minimize the waiting time for the sensors to upload data to the actuators. Specifically, we try to minimize the weighted average actuator inter-arrival time to sensors, which is defined as

$$Minimize \sum_{\forall i} A_i * w_i * N_i, \tag{1}$$

where A_i and N_i are the actuator inter-arrival time and the total number of sensors with weight w_i. We consider periodical routes, i.e., each actuator visits along a route cyclically. It is easy to show that this periodic route

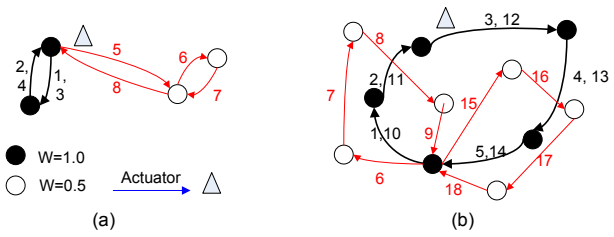

Fig. 1. Two examples of route design.

design minimized the expected inter-arrival time if the route is optimized.

Figure 1 illustrates two examples of route design in a single actuator case. The set of black nodes S_b and white nodes S_w have weights of $W_b = 1.0$ and $W_w = 0.5$ respectively. Let A_b and A_w be the actuator inter-arrival time of all black and white nodes. We expect the inter-arrival time of S_b to be half of S_w, such that $A_w = 2{*}A_b$. The periodic route of the actuator is marked with numbers which indicate the visiting sequence. In this simple example, we assume that actuators move at a constant speed. In Figure 1(a), the actuator would visit the black nodes twice and the white nodes once every cycle. The average inter-arrival time of white nodes is thus,

$$A_w = \frac{2|TSP(S_b)| + |TSP(S_w)| + 2 * \|S_w, S_b\|}{v}, \quad (2)$$

where $|TSP(S_b)|$ is length of the shortest path in the traveling salesman problem (TSP) that contains the set of nodes S_b, $\|S_b, S_w\|$ is the closest distance between the set S_w and S_b, and v is the moving speed of the actuator. Note that the TSP itself is an NP-complete problem, but there are many approximation algorithms available [18].

Figure 1(b) shows a more complicated example with one actuator. Again, it is easy to see that the route design problem is NP-hard even in this single actuator case, so will be the multi-actuator case.

IV. The Probabilistic Route Design (PROUD) Algorithm

We now describe the probabilistic route design (PROUD). In this algorithm, a priori route is calculated at the very beginning. Then, the sensors are visited along the route probabilistically according to their weights. For instance, sensors with a visiting probability 1.0 are visited in every cycle, while sensors with a visiting probability 0.5 only have half chance to be visited in each cycle. In this scheme, an actuator can easily update the visiting probability of the sensors based on its observed data generation rate or event frequency. In other words, the priori route can be re-used without re-calculation when adapting to network dynamics.

In the following, we first give a centralized design that depends on one of the actuators or the base station to execute the algorithm. We then extend it to a distributed implementation in the next section.

A. Small-Scale Network

1) Forming a Priori Route: A priori route is formed by constructing a TSP path which contains all locations to be visited. Many polynomial-time approximation algorithms have been proposed for the NP-hard TSP problem [19][20][21]. We adopt the well-known Approx-TSP-Tour algorithm [21] here for its low cost and bounded performance. This algorithm first creates a minimum spanning tree (MST) whose weight is a lower bound on the length of an optimal traveling-salesman tour. It then creates a tour based on the MST, and the cost of that tour is not more than twice of the optimal. The MST can be created using a polynomial-time, e.g., Prim's algorithm [21].

2) Visiting Sensors Probabilistically: We then apply a probabilistic visiting model, in which actuators visit the sensors on the priori route in sequence, but selectively. Let $s_1, s_2, ..., s_i, s_{i+1}, ..., s_n$ be the sequence of sensor locations along the priori route. After visiting location s_i, the actuator determines whether to visit s_{i+1} by generating a random number between 0.0 and 1.0. If the random number is smaller than the visiting probability of s_{i+1}, i.e., p_{i+1}, then it visits s_{i+1} in the next step. If not, the actuator skips s_{i+1} and determines whether to visit the next location s_{i+2}. This process repeats in every cycle.

Intuitively, the sensors with higher weights should be assigned with a higher probability, such that they are visited more frequently. Hence, we set the visiting probability p_i of a location i to be w_i, where w_i is the (normalized) weight of the sensors. Figure 2 shows an example of probabilistic route design with two types of sensor nodes. The black nodes have visiting probability 1.0, which indicates that they will be visited in every cycle. On the other hand, each white node is visited only with a probability 0.5 in every cycle. The variance of the actuator inter-arrival time can be further reduced by visiting the sensors constantly according to their probability. For instance, a white node can be visited once and then skipped once by an actuator periodically.

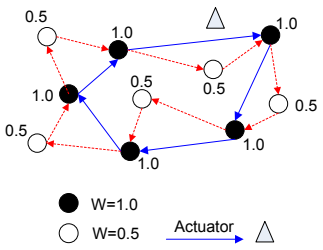

Fig. 2. Visiting nodes probabilistically according to their weights.

3) Allocating the Actuators: For a small network, the actuators can be placed evenly on the priori route during initialization. The actuators then work along the priori route and visit the sensor locations probabilistically according to the weights.

The expected route length with probabilistic visiting can be calculated as

$$E[R] = \sum_{r=0}^{n-2} \sum_{i=1}^{n} \|i, i+r\| * p_i * p_{i+r+1} \Pi_{k=1}^{r} (1 - p_{i+k}), \tag{3}$$

where $1, ..., n$ is a sequence of nodes on the route R of actuator, $\|i, j\|$ is the distance between i and j, and p_i is the visiting probability of i. Note that $E[R]$ depends not only on the visiting probability, but also on the network topology. For instance, an actuator may have to visit several sensor nodes before visiting a particular one if these nodes are all arranged on the same line.

For a sensor i with a visiting probability p_i, its average actuator inter-arrival time A_i is thus

$$A_i = \frac{E[R]}{p_i * v * M}, \tag{4}$$

where v is the moving speed of the actuators.

In a dynamic environment, the visiting probability of the sensors can be updated according to their data generation rate or event frequency, but the route does not have to be re-calculated for each individual change.

B. Large-Scale Network with Partitions

In large-scale sensor networks, network partitions may happen, which divide the sensors into different clusters. In this case, actuators sharing the same route may not be as efficient as walking on distinct routes. Sensors in different clusters should be visited by actuators on independent routes to minimize the inter-arrival time.

1) Forming Clusters: We use a simple algorithm for clustering the sensors, as shown in Algorithm 1. It divides an MST into two sub-trees by removing its longest edge e, provided that $w(e)/w(m) \geq \delta$, where $w(e)$ is the length of edge e. By doing this, the sensors which are geographically far apart will be involved in different sub-trees, and later, distinct routes. Note that δ is set to ensure the number of clusters is smaller than the number of actuators.

Algorithm 1 Clustering the sensors

```
Function Cluster(MST(S))
Find the edge m with the median length;
Find the longest edge e;
if w(e)/w(m) ≥ δ then
    delete edge e;
    Cluster(MST(S_1));
    Cluster(MST(S_2));
end if
```

2) Forming Priori Routes and Visiting Sensors Probabilistically: After clustering the sensors, the probabilistic route design algorithm can be applied in each cluster following the simple case in a small-scale network.

3) Allocating the Actuators: Multiple routes are formed from the above. They may have different expected route lengths due to the various sensor locations and visiting probabilities in the clusters. The uneven expected route lengths may cause unequal inter-arrival times for the sensors with the same weight. To address this problem we can allocate different number of actuators to the routes. Intuitively, routes with longer expected lengths should be allocated with more actuators. This is illustrated in Algorithm 2, where N_R is the total number of routes, $remain_a$ is the number of remaining unassigned actuators, and n_j is the number of actuators assigned to route R_j.

Algorithm 2 Actuator allocation for distinct routes

```
for j = 1 to N_R do
    n_j = 1;
end for
remain_a = M − N_R;
while remain_a > 0 do
    Find the maximum E[R*_j];
    E[R*_j] = E[R*_j] * n*_j/(n*_j + 1);
    n*_j ++;
    remain_a −−;
end while
```

V. Distributed Implementation

For large network, it can be difficult for a single node to collect the information and execute the route

design algorithm in a centralized manner. In this section, we present a practical distributed implementation for PROUD, in which sensors and actuators form clusters by constructing MSTs cooperatively, then the actuators construct the priori routes by traversing the MSTs independently.

1) Forming R-clusters: First, the sensors construct MSTs locally by communicating with their neighbors. Given the communication range of sensors is R_s, the weight of each edge e in the MST must be smaller than or equal to R_s; that is, $w(e) \leq R_s$. We refer to such an MST as an *R-Cluster*, $RC(V, E)$, which includes all the sensors that are within R_s to some sensors in $RC(V, E)$. The cost of the R-cluster is denoted by $Cost(RC)$, which is the sum of $w(e)$, $\forall e \in E$. It will be stored by the sensors in $RC(V, E)$. There are many existing distributed algorithms for forming an MST [22][23], and we apply a fast algorithm from [24] for this purpose.

2) Connecting R-clusters: An R-cluster forest is formed by the sensors as above. These R-clusters can be connected together to form MSTs that contain more sensor locations. We divide the network into M subareas, each of which is explored by one actuator. Each actuator looks for the R-clusters in its area and connects them if they are within a certain distance, say $\|RC_1, RC_2\| \leq \delta$. Then, a new cluster is formed with cost $Cost(RC_1) + Cost(RC_2) + \|RC_1, RC_2\|$.

Similarly, the actuators also connect their R-clusters/clusters with those in their neighboring areas. Algorithm 3 shows how two actuators $A1$ and $A2$ connect their R-clusters RC_1 and RC_2, where BD is the boundary of the two corresponding areas.

Algorithm 3 Connecting the R-Clusters

```
Function Connect-Cluster(RC1(V1, E1), RC2(V2, E2))
if (‖RC1, BD‖ ≤ δ) and (‖RC2, BD‖ ≤ δ) then
    Actuators A1 and A2 exchange locations close to BD;
    Find the shortest edge e that connects RC1 and RC2;
    if e ≤ δ then
        Form new cluster C_new(V, E);
        V = V1 ⋃ V2;
        E = E1 ⋃ E2 ⋃ {e};
        Cost(C_new) = Cost(RC1) + Cost(RC2) + w(e);
    end if
end if
```

3) Allocating Actuators: Actuators are then allocated to the clusters, such that each cluster is assigned to at least one actuator and no clusters are unassigned. This can be achieved by running Algorithm 4 by individual actuators. Each actuator associates itself to any unassigned clusters in its area. If the associated cluster is crossing two or more areas, the actuator has to inform the actuators in those areas. It is possible that the number of clusters is greater than the number of actuators. The unassigned clusters can be connected with some assigned clusters to ensure they are served by at least one actuator. On the contrary, a remaining actuator can associate itself with a nearby cluster with the highest cost. If multiple actuators are serving one cluster, they can divide it equally and serve the sensors involved independently.

Finally, a priori route is computed by the actuator in each cluster using the Approx-TSP-Tour algorithm [21].

Algorithm 4 Allocating actuators to clusters

```
Function Allocate-Actuator (Actuator A)
if ∃ unassigned C_i in A's area then
    A associates with any C_i*;
    if C_i* across other areas χ then
        A alerts the actuators in χ;
    end if
    for ∀ remaining unassigned C_i do
        Find closest assigned cluster C' to C_i;
        Connect-Cluster(C_i, C');
    end for
else
    Find C* with the highest cost in neighboring areas;
    A associates with C*;
    Divide C* equally with other associated actuators;
end if
```

VI. Enhancements to PROUD

In this section, we discuss the integration of actuators with variable speeds and also show two enhancements for load balancing among actuators.

A. Actuators with Variable Speeds

So far we have considered actuators with constant speeds only. Actuators with variable speeds however could achieve even shorter inter-arrival time for heterogeneous networks.

Let o_i be the expected average actuator inter-arrival time for the sensors with weight w_i. For simplicity, we assume the visiting probability p_1 of the sensors with the shortest expected average actuator inter-arrival time o_1 to be 1. The visiting probability of the remaining sensors with the expected average actuator inter-arrival time, say o_i, can be calculated by o_1/o_i. The visiting probability to sensors can be updated adaptively by the actuators according to the dynamic change of the expected average actuator inter-arrival time. By adjusting the speeds of the

actuators, we can ensure sensors with the same visiting probability can achieve similar inter-arrival times, even they are visited by different actuators on distinct routes.

Assume that node i on R_j has a probability p_i of being visited by actuator j every cycle. Its average actuator inter-arrival time A_i can be calculated as

$$A_i = E[R_j]/(p_i * v_j), \tag{5}$$

where v_j is the moving speed of actuator j.

Since A_i must be shorter than o_i,

$$v_j \geq E[R_j]/(p_i * o_i). \tag{6}$$

Without loss of generality, v_j can be determined easily by assuming $p_i = 1$, that is, $v_j \geq E[R_j]/o_1$.

B. Load Balancing in Route Design

For mobile actuators, since its energy consumption is also increasing with its speed [25], the unequal moving speeds might cause imbalanced energy consumption. To tackle this problem, we propose two algorithms to balance the workload of the actuators, while guaranteeing the routes designed are energy-efficient.

1) Multi-Route Improvement Algorithm: Since the actuator on a route with a longer expected length consumes more energy, the loads of actuators can be balanced by forming routes with identical expected lengths. A loaded actuator may assign some of its sensor locations to its neighboring actuator with the minimum expected route length.

Consider two routes R_1 and R_2 involved in multi-route improvement. Their new expected route lengths become ideal if $E[R_1'] = E[R_2'] = (E[R_1] + E[R_2])/2$. In other words, R_1 should transfer a length of $(E[R_1] - E[R_2])/2$ to R_2. Although sensor locations can be transferred one by one from $R1$ to $R2$, until the expected lengths of the two routes become equal, but this kind of node-by-node consideration is inefficient. Therefore, we provide an approximation method to find the proportion of sensor locations ξ to be transferred from MST_1 to MST_2:

$$\frac{cost(\xi)}{cost(MST_1)} = \frac{(E[R_1] - E[R_2])/2}{E[R_1]}, \tag{7}$$

where $cost(\xi)$ and $cost(MST_1)$ represent the costs of the minimum spanning trees that contain the sensor locations in ξ and R_1, respectively.

2) Task Exchange Algorithm: In certain scenarios, it is more energy efficient for one actuator to take up more load than another in the overall energy consumption point of view. For example, the two actuators walking on two distinct routes with unequal lengths (see Figure 3) achieve better performance than those on routes

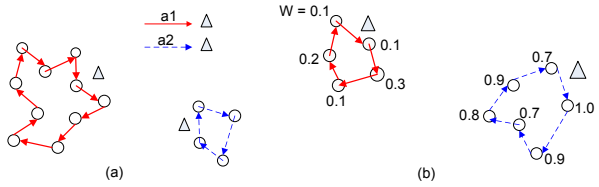

Fig. 3. Routes involve (a) different amount of sensors (b) sensors with different weights.

with identical lengths. It may happen in a network that involves clusters with different sizes or weights. It is unfavorable to enforce load-balancing by equalizing the lengths of the two routes as it will increase the lengths of both routes.

In this scenario, load balancing among the actuators can be achieved by exchanging their routes. Intuitively, an overloaded actuator may exchange its route with another actuator traveling at a lower speed. More formally, we define $Energy_{A1}$ and $Energy_{A2}$ to be the remaining energy of actuator $A1$ and $A2$, and v_1 and v_2 to be the minimum actuator speeds on routes $R1$ and $R2$. A task exchange algorithm is executed when one of the actuator $A1$ has less remaining energy than the other actuator $A2$, but it is required with a higher moving speed. As shown in Algorithm 5, tasks of the two actuators are exchanged by swapping their routes.

Algorithm 5 Task exchanges among actuators

if $(Energy_{A1} << Energy_{A2}) and (v1 >> v2)$ **then**
  $A1$ moves to $R2$;
  $A2$ moves to $R1$;
end if

VII. Performance Evaluation

We have conducted extensive simulations for our proposed route design algorithms with multiple actuators. The simulation settings are summarized in Table I, which are drawn from existing works [6][26][27].

A. Average Actuator Inter-Arrival Time

In the first set of experiments, we evaluate the average actuator inter-arrival time A_{avg} with various kinds of sensor distributions. Note that the average inter-arrival distance D_{avg} is shown in our results, instead of the exact A_{avg}. Given the moving speed v for a specific actuator hardware, the average inter-arrival time of actuator A_{avg} can be readily calculated as D_{avg}/v.

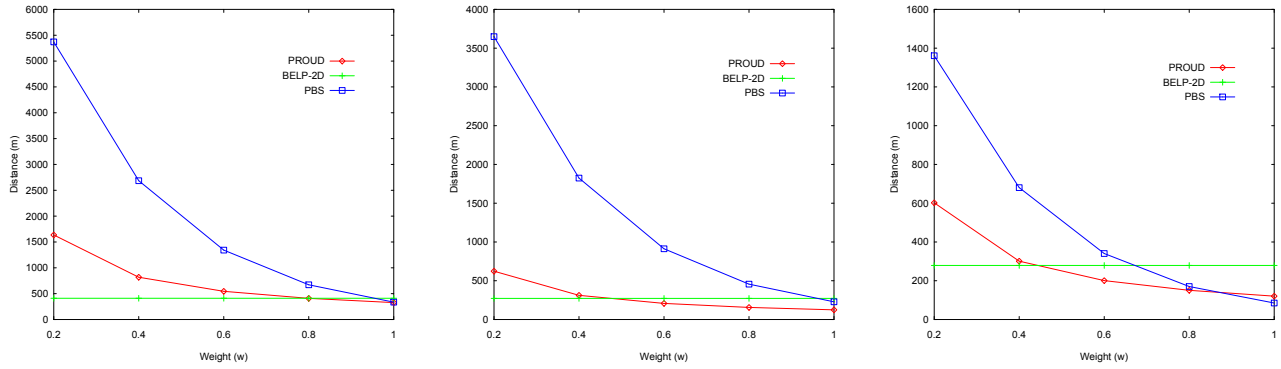

Fig. 4. Actuator inter-arrival time under (a) uniform random (b) cluster-based uniform (c) cluster-based non-uniform sensor distribution.

TABLE I
SIMULATION PARAMETERS

Network size	200m x 200m
Sensor distribution	Uniform random or Cluster-based uniform or Cluster-based non-uniform
No. of sensors (N)	100
Weight of sensors (W_j)	0.0-1.0
No. of actuators	M
Speed of actuators	v
Radio range	40m
MAC layer	IEEE 802.11

We compare our results with that of two state-of-the-art algorithms: the partitioning based scheduling algorithm (PBS) [12] and the bounded event loss probability (BELP-2D) algorithm in the 2D case [13]. The PBS algorithm partitions all nodes into several groups (called bins) and forms a schedule that concatenates them, such that buffer overflow can be avoided in sensors with different data generation rates. The BELP-2D algorithm deals with the bounded event loss problem in a 2D space, which ensures that time elapsed between two consecutive visits is less than a critical time. It uses the solutions of the Traveling Salesman Problem with Neighborhoods (TSPN) to find routes. To achieve a fair comparison, we adapt the Approx-TSP-Tour algorithm [21] to approximate the TSP paths in all the three algorithms.

1) Uniform Random Sensor Distribution: Figure 4(a) shows the average inter-arrival distance D_{avg} for an actuator to visit the sensors periodically under uniform random sensor distribution with $N = 100$ and $M = 5$. It evaluates distances D_{avg} to the sensors with weights in the ranges 0.0-0.2, 0.2-0.4, 0.4-0.6, 0.6-0.8, and 0.8-1.0, respectively. The results of PROUD, PBS, and BELP-2D are shown for comparisons.

The experimental results demonstrate that PROUD, PBS, and BELP-2D have comparable inter-arrival distance D_{avg} for sensors with $w_i = 1$. Both PROUD and PBS differentiate the actuator inter-arrival times according to the weights of sensors. Sensors with higher weights achieve shorter inter-arrival distances D_{avg}, and hence shorter inter-arrival times A_{avg}. However, the D_{avg} of PBS is impractically long for most lower weighted sensors. PBS does not work well here as the locations of bins are widely spread.

On the other hand, BELP-2D achieves constantly low D_{avg} for all sensors, though it does not differentiate the inter-arrival times at all. This is because the route in BELP-2D is the shortest TSP path that contains all the sensor locations. Nevertheless, PROUD is still more suitable for sensor networks with different weights. Recall that the minimum required moving speed of the actuators is determined by o_1 in both PROUD and BELP-2D. Since the A_{avg} of sensors with $w_i = 1$ in PROUD is lower than that in BELP-2D, by $A_{avg} = D_{avg}/v$, the minimum required speed of actuators in PROUD is actually lower than that in BELP-2D.

2) Cluster-Based Uniform Sensor Distribution: We next evaluate our algorithm under cluster-based sensor distribution. Specifically, we place the sensors into three clusters and generate the weights of sensors uniformly and randomly in this experiment.

Similarly, Figure 4(b) shows the average actuator inter-arrival distance D_{avg} of the three algorithms. Under this cluster-based sensor deployment, PROUD achieves shorter D_{avg} than both BELP-2D and PBS algorithms for

sensors with high and median weights. PROUD is able to differentiate the sensor visiting frequency and provide the shortest D_{avg} to highly weighted sensors, which satisfies our main objective. An interesting observation is that the D_{avg} under cluster-based sensor deployment is generally shorter than that under uniform random deployment in all the algorithms. The reason is that the sensors are more concentrated under cluster-based deployment, so they have shorter Euclidean distances, which leads to shorter routes.

3) Cluster-Based Non-Uniform Sensor Distribution: We further evaluate our algorithm under a cluster-based and non-uniform sensor distribution. Apart from deploying the sensors into three clusters, we also put the sensors with similar weights into one cluster here. The weights of the sensors in the three clusters fall in the ranges 0-0.33, 0.33-0.66, and 0.66-1.0, respectively.

Again, Figure 4(c) shows the results for the same network with $M = 5$. We observe that PROUD performs generally better than BELP-2D. It achieves relatively short D_{avg} for sensors with high and median weights. Again, it differentiates the D_{avg}, and hence A_{avg}, among sensors according to their weights. PROUD also achieves comparable D_{avg} with PBS for $w_i = 1$ and much lower D_{avg} for all the remaining sensors.

Overall, PROUD performs generally better than BELP-2D and PBS under various sensor and weight distributions. It always achieves shorter D_{avg} than BELP-2D for highly weighted sensors, and much shorter D_{avg} than PBS for most sensors in all cases.

B. Multi-Route Improvement

We next evaluate the performance of PROUD with multi-route improvement in this experiment. A network with 100 sensors is deployed with uniform random distribution, together with two actuators. The actuators are assigned to two sub-areas at initialization and form distinct routes separately. Since the weights of sensors change dynamically, the two actuators have to update their routes accordingly.

We let the actuators update their routes every 10 mins. The speeds of the actuators with and without multi-route improvement are compared. Figure 5 shows that the two actuators with multi-route improvement walk at closer speeds than that without. It is clear that the multi-route improvement balances the expected lengths of the two routes and reduces the speed difference effectively. This translates into a balanced energy consumption.

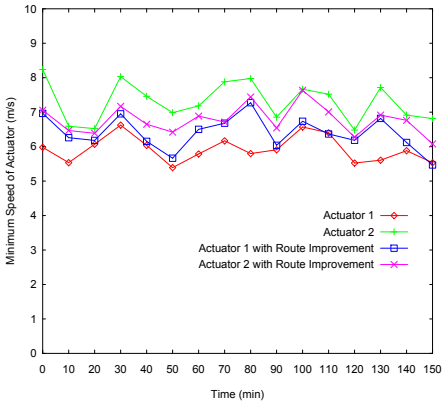

Fig. 5. Speed of actuators with multi-route improvement.

C. Task Exchange Among Actuators

Finally, we evaluate the task exchange algorithm and focus on the energy consumptions of actuators directly. We consider a network with $M = 5$ and $N = 100$ under cluster-based distribution. Again, Clusters I, II, and III are formed, which involve sensors with low, medium, and high weights, respectively.

Figure 6 shows that the actuators without task exchange have significantly different energy consumptions. Particularly, Actuator 1 and Actuator 5 have a significantly different energy consumption (lower and higher, respectively) than the others. Cluster III is assigned with three actuators (Actuators 2,3,4) due to the high weights of its sensors, while Clusters I and II are both assigned with only 1 actuator. Since Cluster II has a longer expected route length than Cluster I, its actuator (Actuator 5) will move faster and consume energy more quickly. With the task exchange algorithm, Actuator 1 and Actuator 5 walk on the routes of Clusters I and II interchangeably to balance their workloads, hence, achieve comparable energy consumptions.

VIII. Conclusion

In this paper, we focused on wireless sensor networks with multiple actuators and their route design. We proposed an adaptive Probabilistic Route Design (PROUD) algorithm, which aims at minimizing the overall inter-arrival time of actuators with non-uniform sensor weights in a dynamically changing environment. It constitutes a significant departure from traditional static and deterministic mobile element scheduling. In PROUD, sensors are visited by actuators probabilistically along a priori route. Sensors with higher weights are visited with higher probabilities, enabling shorter actuator inter-arrival times.

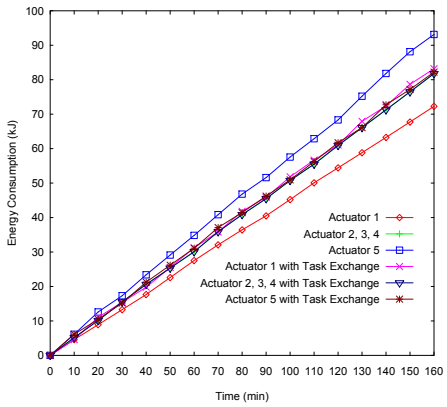

Fig. 6. Energy consumption of actuators with task exchange.

Most importantly, the visiting frequency to sensors can be updated easily by adjusting their visiting probability without complicated route re-calculations. We studied the proposed algorithm for actuators with constant velocity in both small-scale and large-scale networks. We also discussed a distributed implementation and extended the approach to accommodate actuators with variable speeds. We further proposed the Multi-Route Improvement and the Task Exchange algorithms for evenly distributing workload among the actuators. Simulation results suggested that the proposed algorithm can greatly reduce the average inter-arrival times in wireless sensor-actuator networks for highly weighted sensors. The approach also adapts well to the dynamic change of the network and effectively balances the energy consumption of the actuators.

ACKNOWLEDGMENT

The work described in this paper was substantially supported by grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK4205/04E). J. Liu's work was supported in part by a Canadian NSERC Discovery Grant 288325, an NSERC Research Tools and Instruments Grant, a Canada Foundation for Innovation (CFI) New Opportunities Grant, and an SFU President's Research Grant.

REFERENCES

[1] I. F. Akyildiz, W. Su, and T. Sandarasubramaniam, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 5, pp. 393–422, 2002.

[2] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next century challenges: Scalable coordination in sensor networks," in *Proc. of ACM MobiCom*, Seattle, Washington, U.S., 1999.

[3] Z. M. Wang, S. Basagni, E. Melachrinoudis, and C. Petrioli, "Exploiting sink mobility for maximizing sensor networks lifetime," in *Proc. of the 38th Hawaii International Conference on System Sciences (HICSS)*, 2005.

[4] J. Luo and J. Hubaux, "Joint mobility and routing for lifetime elongation in wireless sensor networks," in *Proc. of the 24th IEEE Infocom*, Mar 2005.

[5] I. F. Akyldiz and I. Kasimoglu, "Wireless sensor and actor networks: research challenges," *Elsevier Ad Hoc Networks Journal*, Oct 2004.

[6] E. C.-H. Ngai, Y. Zhou, M. R. Lyu, and J. Liu, "Reliable reporting of delay-sensitive events in wireless sensor-actuator networks," in *Proc. of the 3rd IEEE MASS*, Vancouver, Canada, Oct 2006.

[7] A. A. Somasundara, A. Ramamoorthy, and M. B. Srivastava, "Mobile element scheduling for efficient data collection in wireless sensor networks with dynamic deadlines," in *Proc. of the 25th IEEE RTSS*, 2004, pp. 296–305.

[8] R. Shah, S. Roy, S. Jain, and W. Brunette, "Data mules: Modeling a three-tier architecture for sparse sensor networks," in *Proc. of the IEEE Workshop on Senosr Network Protocols and Applications (SNPA)*, 2003.

[9] A. Chakrabarti, A. Sabharwal, and B. Aazhang, "Using predictable observer mobility for power efficient design fo sensor networks," in *Proc. of the 2nd International Workshop on Information Processing in Sensor Networks (IPSN)*, Apr 2003.

[10] A. Kansal, A. Somasundara, D. Jea, M. Srivastava, and D. Estrin, "Intelligent fluid infrastructure for embedded networks," in *Proc. of the 2nd ACM MobiSys*, 2004.

[11] W. Zhao, M. Ammar, and E. Zegura, "A message ferrying approach for data delivery in sparse mobile ad hoc networks," in *Proc. of ACM MobiHoc*, Mar 2005.

[12] Y. Gu, D. Bozdag, E. Ekici, F. Ozguner, and C.-G. Lee, "Partitioning-based mobile element scheduling in wireless sensor networks," in *Proc. of SECON*, Santa Clara, U.S., Sep 2005, pp. 386–395.

[13] N. Bisnik, A. Abouzeid, and V. Isler, "Stochastic event capture using mobile sensors subject to a quality metric," in *Proc. of ACM MobiCom*, Sep 2006.

[14] Z. Zhang and Z. Fei, "Route design for multiple ferries in delay tolerant networks," in *Proc. of IEEE WCNC*, Mar 2007.

[15] N. Christofides, A. Mingozzi, and P. Toth, "Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations," *Mathematical Programming*, no. 1, pp. 255–282, Dec 1981.

[16] T. Ralphs, L. Kopman, W. Pulleyblank, and L. Trotter, "On the capacitated vehicle routing problem," *Mathematical Programming*, no. 2-3, pp. 343–359, Jan 2003.

[17] L. Lee, K. Tan, K. Ou, and Y. Chew, "Vehicle capacity planning system: A case study on vehicle routing problem with time windows," *IEEE Trans. on Systems, Man and Cybernetics, Part A*, pp. 169–178, 2003.

[18] A. Dumitrescu and J. S. B. Mitchell, "Approximation algorithms for TSP with neighborhoods in the plane," in *Proc. of SODA'01*, 2001, pp. 38–46.

[19] J. Bentley, "Fast algorithms for geometric traveling salesman problem," *ORSA Journal on Computing*, pp. 387–411, 1992.

[20] S. Arora, "Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems," *Journal of ACM*, vol. 45, no. 5, pp. 753–782, Sep 1998.

[21] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and et al., *Introduction to Algorithms*. The MIT Press, 2002.

[22] R. G. Gallager, P. A. Humblet, and P. M. Spira, "A distributed algorithm for minimum-weight spanning trees," *ACM Trans. Program. Lang. Syst.*, vol. 5, no. 1, pp. 66–77, 1983.

[23] B. Awerbuch, "Optimal distributed algorithms for minimum weight spanning tree, counting, leader election, and related problems," in *Proc. of ACM STOC*, 1987, pp. 230–240.

[24] M. Elkin, "A faster distributed protocol for constructing a minimum spanning tree," in *Proc. of ACM-SIAM SODA*, 2004, pp. 359–368.

[25] Y. Mei, Y.-H. Lu, Y. C. Hu, and C. G. Lee, "A case study of mobile robot's energy consumption and conservation techniques," in *Proc. of IEEE International Conference on Advanced Robotics*, 2005, pp. 492–497.

[26] T. He, J. Stankovic, C. Lu, and T. Abdelzaher, "SPEED: A real-time routing protocol for sensor networks," in *Proc. of IEEE ICDCS*, Providence, RI, U.S., May 2003, pp. 46–55.

[27] E. Felemban, C.-G. Lee, and E. Ekici, "MMSPEED: Multipath multi-SPEED protocol for QoS guarantee of reliability and timeliness in wireless sensor networks," *IEEE Trans. on Mobile Computing*, vol. 5, no. 6, pp. 738–754, 2006.