Lightweight Clustering in Wireless Sensor-Actuator Networks on Obstructed Environments

Ricardo Lent and Edith C.-H. Ngai
Department of Electrical and Electronic Engineering,
Imperial College London, United Kingdom
Email: {r.lent, e.ngai}@imperial.ac.uk

Abstract— **Wireless sensor-actuator networks (WSANs) can be useful to cope with the connectivity limitations of sparse networks by allowing powerful and mobile actuators periodically collect data from sensors. We propose a low-overhead algorithm that takes advantage of any potential connectivity present in sensors to form clusters that can expose single collection points, therefore, optimizing actuator data collection rates. No prior knowledge assumptions on the location of sensors, localization algorithms,or environment conditions are made in the design of the algorithm. Environment exploration is introduced as well as self-correcting tour mechanisms. Detailed simulations of high level statistical accuracy support our clustering approach and demonstrate the critical design issues of the algorithm.**

I. Introduction

Wireless sensor networks (WSNs) have been applied to a wide range of applications such as environment monitoring, target tracking, fire detection and battlefield surveillance [1][2]. Recently, actuators, such as mobile sinks [3] or mobile relays [4], have been introduced for collecting data in traditional WSNs. Actuators, which have much stronger computation and communication power than uni-purpose micro-sensors, are suggested as an efficient way for solving the network partition problem and prolonging sensor network lifetime [5][6].

In Wireless Sensor-Actuator Networks (WSANs), mobile actuators can move around in the sensing field and collect data from the static sensors. The sensors store sensing data in their buffers temporarily. When an actuator approaches, they will report the data and free their buffers. In many applications, the actuator may know little about the environment, such as the obstacles and the locations of sensors in the field. It is crucial for the actuator to explore the environment and discover the sensors automatically. It can then design a path avoiding the obstacles and collect data efficiently, so as to reduce the service waiting times and minimize buffer overflows.

In this paper, we propose a practical and efficient approach for actuator to explore the environment and design the path for data collection automatically. It aims at collecting data with obstacle avoidance, while reducing the service time to sensors. We consider an network area with obstacles and the locations of sensors are not given in advance due to random deployment, i.e. by plane. We provide an automatic algorithm for actuator to explore the environment and determine the communication points with sensors in the discovery phase. This is followed by data collection and path refinement in the operation phase. We then explore the connectivity in the network and extend the approach by forming clusters among the sensors. This approach further reduces the number of communication points, hence, leads to a shorter path for actuator and shorter service times for sensors. Our simulation results show that the proposed approach can adapt well to the environment and minimize buffer overflows effectively.

II. Related Work

Actuators, or mobile elements, have been considered for carrying data in wireless sensor networks. Shah et al. [7] presented an architecture using moving entities (data mules) to collect data in sparse sensor networks. Similar approaches on mobile sinks with predictable and controllable movement patterns [8][9], and optimal time schedule for locating sojourn points [3] have also been studied. Apart from that, Gu et al. [10] proposed a partitioning-based algorithm to schedule the movement of actuator (ME) to avoid buffer overflow in sensors and reduce the minimum required ME speed. Bisnik et al. [11] studied the problem of providing quality coverage using mobile sensors and analyzed the effect of controlled mobility on the fraction of events captured. Luo et al. [4] investigated a joint mobility and routing algorithm with mobile base station to prolong the lifetime of wireless sensor networks.

978-1-4244-1653-0/08/$25.00 ©2008 IEEE

For point-to-point communications, Zhao et al. [12] proposed a message ferrying (MF) approach to address the network partition problem in sparse ad hoc networks. [13] proposed a route design algorithm for multiple ferries, which 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 path design and data collection in a network area with obstacles, where the environment information and locations of sensors are not provided in advance.

III. Network Model

We consider a wireless sensor network consisting of M actuators and N motes on a field that may contain mobility obstructions. Such obstacles are physical structures with arbitrarily location and of different sizes and shapes. Common examples of obstacles are lakes, rivers, holes, buildings, etc., which may also affect wireless communications.

Actuators can move at a normal speed over the sensing field surrounding obstacles as needed if any block their way. However, their actual moving speed may vary depending on the characteristics of the terrain. For example, the presence of slopes or vegetation may affect the final moving speed of actuators.

We assume that the M actuators move independently, so that it would make sense to partition the field into sub-fields to be assigned to different actuators.

Actuators are equipped with a positioning reading device, such as a GPS receiver that would allow them to determine their approximate location. In our proposed algorithm, motes are not required to have localization facilities. Actuators start without initial knowledge of the sensors' location and are given just the target number of sensors to visit and the size of the sensing field. Therefore, sensor discovery will be required.

Motes produce data samples at a given rate, which are stored on a buffer of finite capacity. Motes and actuators communicate over a shared wireless channel, so channel contention and collisions could be expected. Actuators may collect data samples from sensors as long as they lie within their communication range. Once collected, sensors may free their buffers to hold new samples. Excessive delays between actuator visits would produce sample losses due to buffer overflows. Overflow management may vary (e.g. drop newest samples, drop oldest samples or drop least important samples). Unlike previous works, we assume that one visiting point may serve to access more than one sensor.

The tour design problem is to determine an efficient cycle for an actuator to visit motes to collect data and minimize chances of data loss, while allowing motes to organize and elect collection points.

For the sake of clarity, we will describe the touring algorithm in two steps. The first without clustering but considering the possibility of obstacles in the environment. The second part will complete the description of our lightweight clustering algorithm.

IV. Touring Algorithm

As stated in Section III, an starting actuator is required to discover a predetermined number of N motes before proceeding to creating and following a tour. A protocol supporting these activities can naturally distinguish two phases: discovery and operation.

A. Discovery Phase

During the discovery phase, the actuator explores the environment to learn the approximate location of sensors. We have adopted a simple spiral walk for the discovery phase. While following the walk, an actuator periodically broadcast *Enable* messages (ENB). If a non-enabled sensor receives such message, it gets enabled (starts collecting data) replying with an *Acknowledgement* (ACK) message containing the mote ID.

With each ACK, the actuator associates its current location with the mote ID in the packet. The set of locations where ACKs were received will form the waypoints of the tour.

Obstacles in the environment may restrict the actuator visiting certain locations when following just the spiral walk. Moreover, collisions on the channel may produce packets losses preventing some sensors to be discovery on a first try. If the target number of sensors N is not achieved by the spiral walk within the boundaries of the sensing field, the actuator will attempt to follow a random waypoint walk broadcasting ENB messages as previously for a period of time before giving up (Algorithm 1).

However, the obstacles and the landscape can complicate the problem, which will be addressed by our path refinement algorithm in the operational phase.

At the moment, the actuator will form an initial path with the locations of communication points based on the TSP, as shown in Algorithm 2. Note that the TSP itself

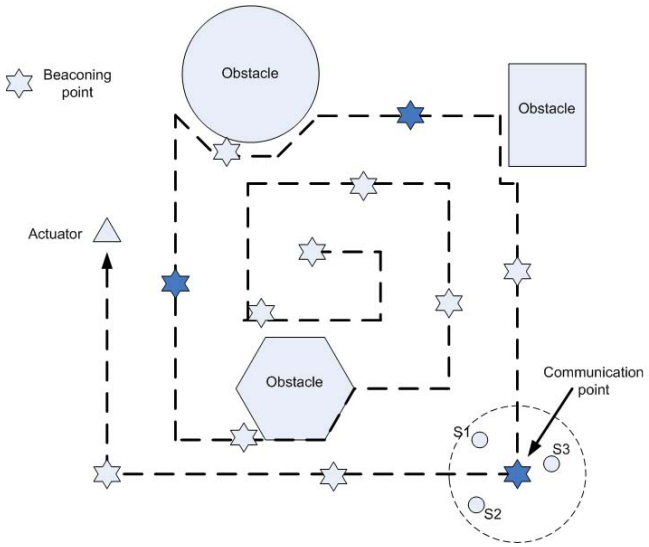

Fig. 1. Spiral walk in environment exploration.

Algorithm 1 Finding the waypoints

```
Actuator a follows Spiral Walk within field boundaries;
for each time interval t do
  a beacons ENB message;
  if ACK message is received from sensor {S} then
    Add actuator position (x,y,z) as new waypoint C if one
    does not exist yet
    Associate mote ID {S} with C;
  end if
  if N motes discovered then
    Finish
  end if
end for
Repeat, replacing Spiral Walk with Random Waypoint Walk
```

1) Tour Setup: After the discovery phase, W waypoints ($W \leq N$) are known, which will define the visiting locations of an actuator tour. It is desirable to find short tours to visit all W waypoints for quick inter-visit times and early buffer discharge. Solving for the shortest tour is equivalent to solving the well-known travelling salesman problem (TSP) with travelling times as edge weights, which is an NP-complete problem. Note that although the location of the waypoints and the nominal actuator speed v are known, the exact travelling time between two waypoints i and j is not know ($Cost_{i,j}$) at this point, as it will depend on the environment. However, a first approximation of the tour can be calculated assuming Euclidean distances and improved later once the actuator measures travelling times by using the tour.

Several approximate solutions to the TSP exist [14][15][16]. We have adopted the well-known Approx-TSP-Tour algorithm [16] because of its low cost and bounded performance. The algorithm starts by defining a complete graph with the W waypoints as vertices and calculating a minimum spanning tree (MST) with Prim's algorithm in polynomial time. The weight of the MST determines a lower bound on the length of the resulting traveling-salesman tour. The actuator tour results from the pre-order traversal of the MST.

Algorithm 2 Tour setup

```
Create graph G
if Cost_{i,j} known by measurement then
  Insert edge(i,j) into G with measured Cost_{i,j}
else
  Insert edge(i,j) into G with Cost_{i,j} = Dist_{i,j}/v
end if
Runs Approx-TSP-Tour algorithm
```

B. Operational Phase

Once an initial tour has been established, the actuator can enter the operational phase and start visiting the waypoints on the tour. While on this phase, the actuator has two objectives: collect data and measure travelling times to improve the tour.

1) Data Collection: At each waypoint, the actuator broadcasts a *request* message (REQ) to induce sensors in the area to send their data. As long as a sensor has something to send, it will send one or more *reply* message (RPY) to the actuator with its collected data samples.

After reporting the data, a sensor can free its buffer. Intuitively, buffer overflows should be minimized with shorter actuator visiting paths, and hence, a shorter waiting time for the actuator. The path of actuator can be further reduced if the connectivity of sensors are put into consideration as we will develop in the next section.

2) Tour Update: As the actuator moves on a tour and measures actual travelling times, such measurements can be used to further improve tours by making again use of Algorithm 2. Actual travelling times are affected by obstacles in the environment and terrain characteristics. Tour updates can occur once in a cycle and typically follow a transient period of a few cycles before reaching a stable state.

A discussion of obstacle avoidance mechanisms is beyond the scope of this paper so we refer the reader to the literature [17][18].

V. Lightweight Clustering

The idea is to take advantage of any existing connectivity in sensors for data collection creating islands of groups of clusters and by making the actuator collect data from the cluster by visiting only one of its members. Note that in very sparse networks, there is little possibility of creating clusters whereas in very dense networks all sensors may form a single cluster (and no mobility in the actuator would be needed).

Because of energy limitations, our approach attempts to create minimum overhead to setup and operate clusters by reusing the basic protocol described in the previous section. Communications use broadcast at the MAC layer but include higher-layer information to identify the intended destination of the messages. MAC-layer broadcasts allow sensors overhear other sensor communications and take actions, whenever possible, to join or use the cluster to send their data.

A. Discovery Phase

As explained previously, during discovery not-enabled sensors receiving the ENB message reply with an ACK message. Because of the broadcast nature of the communications, the ACK can be received by potential sensors located within the communication range of the transmitting sensor by beyond the range of the actuator. Such sensors, not previously enabled, would become members of the cluster and the transmitting sensor the cluster head for them. To become a new cluster member, a sensor just need to send an ACK message as before but addressed to the next sensor from which it overheard the previous ACK. Cluster members are therefore required to remember that next hop (relay) for sending future messages. The process can be repeated to a desired depth by including in the ACK the distances (in hops) that the sender is from the actuator. On the other hand, other sensors receiving an ACK and already enabled may need to forward the message.

Replicated ACKs function the same as normal ACKs and allow the actuator learn the sensors included in a given waypoint.

B. Operational Phase

Clustering only introduce additional functionality to the sensors but not to the actuator, which operates in the same way in both cases. The actuator can continue collecting data from a single sensor or a whole cluster by visiting the pre-determined waypoints. As before, at each waypoint, the actuator broadcasts a REQ message. Sensors received the REQ message reply with a RPY message, which can be overheard, as with the case of ACK, by nearby sensors. The cascading RPY then can proceed in the same way.

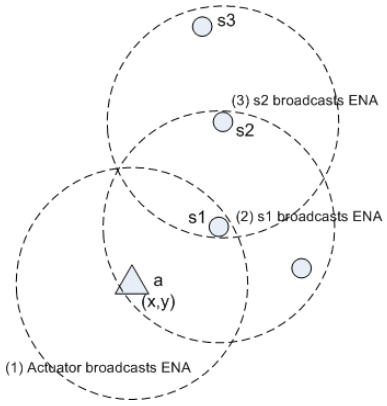

Fig. 2. Forming clusters.

VI. Evaluation

The actuator-sensor system described in Section III was implemented in INES [19], which has support for wireless communications, as well as obstacle and mobility models. The simulations consider a squared field with a 250 m side that contains obstructions, such as vegetation, water bodies and man-made structures (Figure 3).

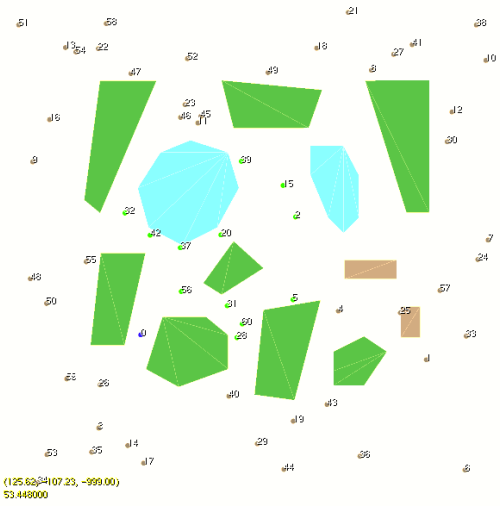

Fig. 3. Obstructed field used in the evaluation of the system.

Sensors and actuator communicate over a 2.4 GHz wireless channel and use a transmission power of 0.1 dBm that permits a communication range of about 22m with the free space radio propagation model. Channel

access is controlled by a CSMA MAC layer identical to that of the IEEE 802.11, but with a transmission rate of 250 Kbps and a buffer capacity of 50 packets.

Each simulation consists on starting an actuator at the center of the field to follow a preset pattern and to discover N sensors to create a tour. Afterwards, the actuator will follow the tour and possibly correct it if needed, while collecting data from sensors at each waypoint until the end of the simulation, 6–20 hours of simulated time. Motes are placed on the field at random locations not occluded by any obstacle and exceeding by 20% the target tour size given to the actuator to speed up the discovery phase. Once a sensor is discovered by the actuator, it enters its *enabled* mode and starts producing data samples at a given rate ($\lambda = 0.2$). Each sensor has capacity for storing up to 100 samples, which the actuator collects when visiting the node. Sample losses may occur due to buffer overflow or transmission loss. Simulations were repeated at least 10,000 times to obtain the average values reported next.

A. Effects of Obstacles

Obstacles reduce the effective operational space of the field allowing for faster setup times. The average setup time for the unobstructed case (U) and two cases with obstacles (O1 and O2) are depicted in Figure 4 versus the target tour size. The cluster-based cases are discussed on the next section. Case O1 considers tours computed just from the Euclidean distance between waypoints regardless of the actual distance that the actuator travels. Case O2 allows for tour correction after the actuator learns the actual travel time.

A longer setup time was observed for the U case. As expected, there was no time difference in the discovery phase of O1 and O2 because they work under the same conditions at the beginning. On the other hand, case U produced a larger number of waypoints per tour (Figure 5).

The actual distances travelled on each tour are depicted in Figure 6. The figure shows two effects. First, actual travelled distances tend to be higher when obstacles are present on fields with low sensor density because of the extra travel to avoid obstacles. On the other hand, as sensor density increases the extra 20% of sensors deployed allow for smaller tours on the reduced space of the O1 and O2 cases. Obstacles also produced longer waiting times (Figure 7) and a higher sample loss ratio (Figure 8).

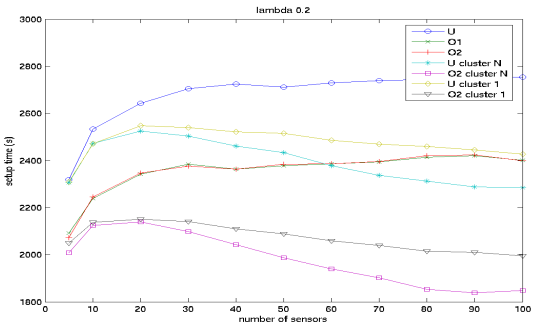

Fig. 4. Tour setup time.

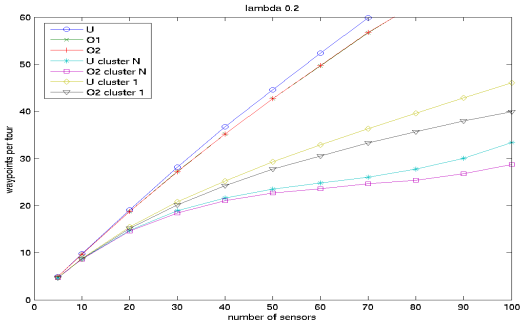

Fig. 5. Number of waypoints in a tour.

B. Effects of Clustering

Clustering reduces the mobility requirements of the actuator to achieve the target tour size. In this experiment, we consider four cases: U cluster 1, O2 cluster 1, U cluster N with unlimited size and O2 cluster N with unlimited size for the unobstructed and obstructed scenarios with unrestricted and 1-hop restriction clustering.

Clusters are formed during the discovery phase by taking advantage of the overhearing of ACK packets sent by motes recently discovered by the actuator. Motes with direct connection to the actuator may serve as cluster heads. Clearly, the location and density of sensors would determine which motes may become cluster heads. Figure 9 shows the ratio of cluster heads to motes with direct connection to the actuator as observed during the simulations.

Motes overhearing ACK packets and not previously discovered would try to join the cluster by sending their own ACK. The size of clusters can be affected by limiting the number of hops beyond motes with direct connection to the actuator during cluster formation. We have considered both a 1-hop restriction and unrestricted

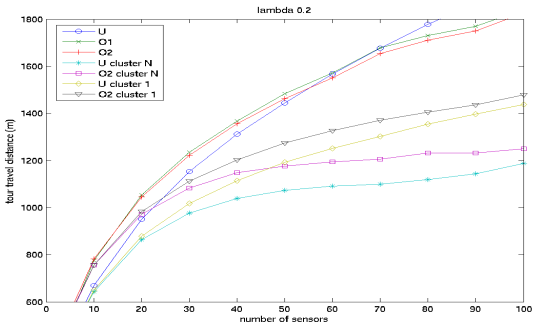

Fig. 6. Length of the tour.

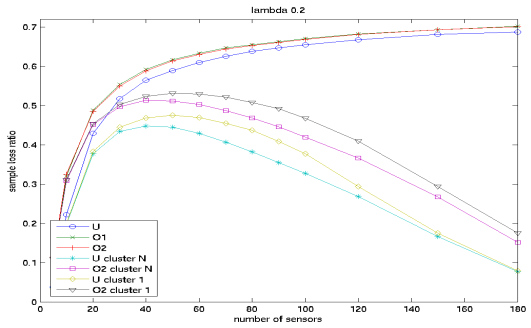

Fig. 8. Sample loss ratio.

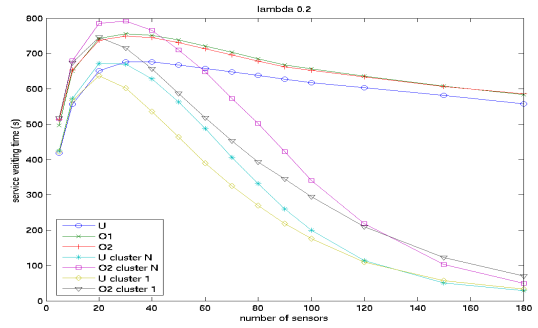

Fig. 7. Waiting time to actuator.

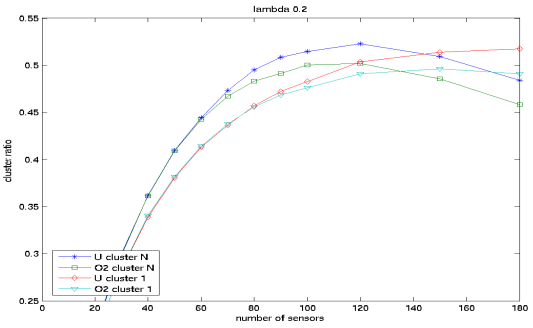

Fig. 9. Cluster ratio.

cases. The latter allows the formation of clusters as large as possible. Figure 10 depicts the resulting cluster size observed during the simulations. The cluster size excludes the cluster head itself.

Larger clusters required shorter discovery times and less number of waypoints, which produced shorter tours (Figure 6), such that cluster N with unlimited size always achieve less number of waypoints and short tours than cluster 1 with only one node. The overall effect is a very significant decrease in service waiting times and sample losses (Figure 8) as node density increased.

The down side of using large clusters is that the number of retransmissions for packet forwarding may scale up quickly, which can increase channel contention, therefore, degrading the service waiting times and increases path lengths (Figure 11). Moreover, forwarding produces higher energy consumption in motes. We denote transmission overhead as the ratio of the number of sample transmissions (by motes and including forwarding) to sample receptions by the actuator. Transmission losses may occur due to wireless collisions. However, when no clusters are used, the transmission overhead should be close to one.

On the other hand, overhead increases quickly with clustering as depicted in Figure 12 which produces extra energy consumption in motes.

VII. Conclusions

In this paper, we studied the problem of self navigation and data collection in wireless sensor-actuator networks. The main contribution of the paper has been the introduction of a low-overhead clustering algorithm in actuator tour formation that takes advantage of any connectivity present in sensors. The paper was based on realistic assumptions, therefore, introducing also support for environment exploration mechanisms to discover sensor locations and self-correcting tour mechanisms that use actual travelling times of the actuator. Such information is typically not known apriori and can be affected by the terrain and temporal changes. The proposed solution provides an efficient way to construct the path for actuators to collect data from sensors autonomously, such that the service waiting time and buffer overflows in sensors are minimized.

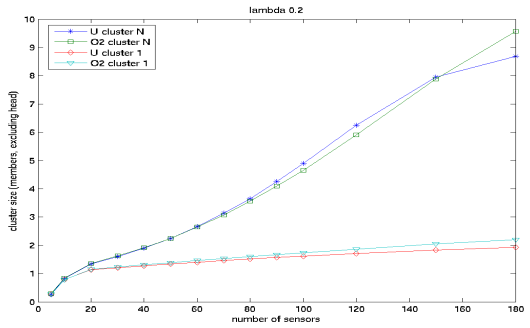

Fig. 10. Cluster size.

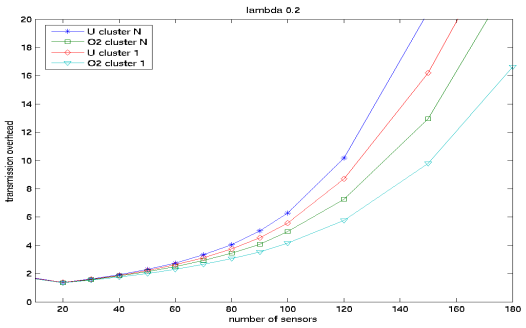

Fig. 12. Transmission overhead.

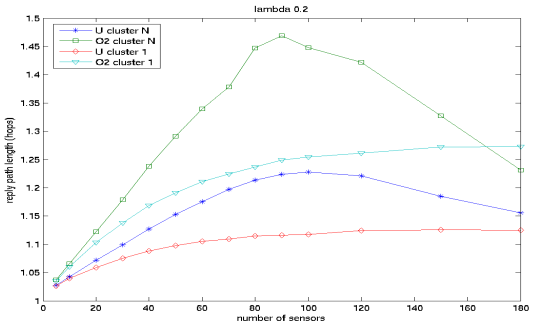

Fig. 11. Path length for reporting data.

We conducted a detailed simulation study of high statistical accuracy on a large computing cluster that demonstrated that sensor clustering can effectively reduce service waiting times and minimize data loss in wireless sensor-actuator networks as well as reducing environment exploration times. The simulations included wireless channel contention supported by a CSMA layer and the effects of obstructions in the environment and path planning to the travelling times of actuators. Our results have also suggested that it is advisable to restrict the size of clusters as they offer better tradeoffs to energy consumption and data loss.

ACKNOWLEDGMENT

The work presented in this paper was partially supported by the project CASCADAS (IST-027807) funded by the FET Program of the European Commission. The paper represents the work and contribution of an individual party involved in the project.

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] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

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

[15] 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.

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

[17] C.-Y. Wong and U. Qidwai, "Intelligent sensor network for obstacle avoidance strategy," in *Proc. of the IEEE Conference on Sensors*, Irvine, CA, USA, Nov 2005.

[18] R. Ghurchian, T. Takahashi, Z. Wang, and E. Nakano, "On robot self-negation in outdoor environments by color image processing," in *Proc. of the International Conference on Control, Automation, Robotics and Vision*, Singapore, Dec 2002.

[19] R. Lent, "Ines: Network simulations on virtual environments," in *Proceedings of International Conference on Simulation Tools and Techniques for Communications, Networks and Systems*, March 2008.