An Efficient Event Handling Protocol for Wireless Sensor and Actor Networks

Charalampos Konstantopoulos
Department of Informatics
University of Piraeus, Greece
Email:konstant@unipi.gr

Grammati Pantziou
Department of Informatics
Technological Educational Institution of Athens, Greece
Email:pantziou@teiath.gr

Ioannis E. Venetis
Department of Informatics
Technological Educational Institution of Athens, Greece
Email:ivenetis@teiath.gr

Damianos Gavalas
Department of Cultural Technology and Communication
University of the Aegean, Greece
Email:dgavalas@aegean.gr

***Abstract*—A critical issue dealt with in Wireless Sensor and Actor Networks (WSANs) is the real time response of actors to events occurring in the network area. The fast notification of actors from Sensor Nodes (SNs) about the events as well as the effective coordination of actors for prompt event handling is most important in these networks. In this paper, we introduce a distributed protocol for effective event handling in a WSAN. Existing approaches mainly favoring actors near the current event for handling it, may lead to highly ineffective solutions for certain worst-case scenarios. Through a randomized approach, our solution also selects distant actors for handling events, and guarantees fast average responsiveness to events and a balanced energy distribution among actors. In addition, it features efficient distributed algorithms for sensor to actor and actor to actor coordination which are of independent interest.**

I. Introduction

WSANs is an important enhancement over the classical wireless sensor networks (WSNs) featuring a number of active elements (actors) that can directly act on events sensed by the SNs of the network. The most common situation arising in WSANs deployments, is the occurrence of critical events demanding quick response from actors. Thus, SNs triggered by events should be able to send prompt notification to one or more actors; thereafter the actors should quickly decide on the actor that will handle the current event. The most common practice is for nearby actors to handle the current event. Such a greedy reaction may be the most natural way of handling events but in some scenarios, this approach may not be the optimal one in terms of the total distance traveled by the actors. The total travel distance of actors is critical for their energy supply since the energy consumption due to physical movement is typically much higher than the energy consumed for communication with other actors or SNs. Also, this metric divided by the number of events occurred in a time interval is proportional to the average delay for handling these events, given a constant speed for actor movement.

This research has been co-financed by the European Union (European Social Fund ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) Research Funding Program: Archimedes III. Investing in knowledge society through the European Social Fund.

In this paper, we differentiate from existing approaches [1] where actors close to the current event are mainly selected for handling it, and we propose a solution where distant actors might be selected. This approach handles worst-case scenarios where the greedy approach of always selecting an actor close to the current event, may lead to excessive actor movement and hence, to solutions with high energy consumption and high event handling delay. A useful algorithmic paradigm for our problem is the on-line k-server problem [2]. This problem considers a metric space (e.g. euclidian) and the existence of k servers which can move and handle requests occurring in this space. After each request, a server is selected to relocate and serve the request. The on-line algorithm for the k-server problem should make decisions without knowing any of the future requests. The main objective is the minimization of the total distance traveled by servers handling a sequence of requests. Clearly, there is a direct analogy between the k-server problem and that of the on-line event handling in a WSAN.

We present a distributed solution for effective event handling in a WSAN which employs a randomized approach for selecting actors to handle events. Specifically, our approach proposes an efficient implementation of the harmonic algorithm (HA) [2] on WSANs. Note that the HA is an inherently centralized approach for solving the k-server problem, and not easily amenable to distributed implementation. Although, methods for emulating centralized on-line algorithms in a distributed setting exist (e.g. [3]), they are mainly of theoretical interest and their implementation on WSANs incurs high energy cost. Our solution guarantees fast event handling on average, and fairly distributes the workload among actors resulting in balanced energy consumption across actors. The experimental results confirm the above advantages of our approach over other approaches in the literature. In addition, our protocol features efficient distributed algorithms for actor to actor coordination which are of independent interest. Although, the actor coordination problem in WSANs has already been studied in the literature, most previous works provide a rather high level description of coordination protocols without many details about the distributed implementation on WSANs [4], [5]. In this work, we analytically describe the actor coordination protocol, highlighting the inherent difficulties that such

an implementation commonly has. Specifically, our techniques are based on geometric characteristics of the event area for achieving effective arbitration among actors. In addition, due to the real-time constraints of event handling in WSANs, we mainly focus on the fast implementation of the proposed protocols. Since faithfully implementing the harmonic algorithm may incur high message (energy) cost and long execution delay or may not be feasible in some cases (e.g. disconnected actor network), we opted for fast protocols which, in some cases, may deviate from the typical harmonic algorithm. Fortunately, these situations are not very likely to appear as explained later, and when they do occur, the degradation in the protocol efficacy is acceptable.

The rest of the paper is organized as follows. In Section II, relevant work to our research is presented. In Section III, the proposed distributed protocol is described. In Section IV experimental results for the performance of the protocol are presented, while Section V concludes our work.

II. Related Work

The most critical issue in WSANs is the rapid response of actors to events occuring in the network area with minimum energy consumption on SNs and actors [1]. First, SNs sense the event and then should send their reports to an actor (*sensor-actor commumication*) with low delay and energy cost. However, the actor receiving event reports may not be the best actor to act for this event; thus, the actors should cooperate for deciding which actor will eventually handle the on-going event (*actor-actor coordination*). These two steps are the main focus of all works on WSANs [4]. In [6], a scheme for sensor-actor and actor-actor communication is presented. The actors are stationary and SNs in an event area are partitioned in different groups, each reporting to a different actor. The actor-actor coordination problem is formulated as a Mixed Integer Non-Linear Program and a localized auction protocol runs for deciding the handling actors for the current event. The same authors in [7] also present a scheme with mobile actors. Each SN predicts the current position of the closest actor using Kalman filtering. The actor coordination problem is again formulated as a Mixed Integer Non-Linear Program. In addition, the algorithm may postpone some low priority events in favour of a new high-priority event.

The authors in [8] study the assignment of actors to events such that the overall distance traveled by actors for handling events is minimized. The solution should also guarantee response time under a certain threshold and that each event is handled by a sufficient number of actors. In [9], SNs sensing an event are clustered around the node first detecting the event. Each of these nodes forms an event map from the reports received from their cluster members and then send the map to their closest actor. Then, actors positioned nearby the event decide on which of the actors will eventually handle the event. The authors in [10] propose a cluster-based approach for sensor-actor communication. Each cluster head gathers the reports from its members and then, based on these data, it selects the handling actors for the on-going event so that the action range of the actors involved has minimal overlap.

In [11], a dominating set is found in the sensor network such that SNs are at most k hops away from at least one dominator and any two dominators are at least 1-hop away of each other. Then, actors are positioned at the locations of these dominators. In [12], SNs are organized in clusters and a distributed protocol based on the Galey Shapley (GS) algorithm from stable matching theory is proposed for matching actors to cluster heads with minimum message overhead and minimum actor relocation. In [13], a sensor-actor communication scheme with reliable event-reporting is proposed. The event reports from SNs are sent to actors within a certain time bound and with different priorities according to the importance of each event. In [14], an auction based coordination protocol is proposed. The actor first notified for an event broadcasts a request for bids from nearby actors and the latter return their bids to that actor. The actor incurring the lowest cost for handling the event is selected for event handling. In [15], a framework for sensor-actor and actor-actor coordination is proposed where a number of deadlines are imposed for completing the different phases of event handling. Each SN sensing an event decides on the actor being notified based on the remaining energy, the distance from the SN and the load of nearby actors. Similar criteria are considered for deciding the actors which will handle the on-going events. In [5], a localization service is used for a SN sensing an event to find its nearest actor and then the actually nearest actor to the current event is found by a localized auction protocol from [14].

Fig. 1. Failure of greedy technique for handling events.

A common feature of the works above is that they act myopically, considering only the current events. However, low cost handling of the current event may subsequently lead to high handling cost for all future events. Usually, an event triggers additional events in the local area and a solution which gradually gathers more actors in the eventful area will surely pay off finally. So, in our solution, besides the actors near the current event, distant actors may also be selected with comparatively lower probability. This less greedy approach achieves lower event handling delay on average, as well as lower energy consumption due to actor movements. A main contribution of this work is the implementation of the inherently centralized HA on a WSAN with low energy and communication cost. Also, as mentioned in the introduction , we also give all the necessary details of this implementation.

III. The Distributed Event Handling Protocol

Let $N(S, A)$ be a WSAN with a set of sensors S and a set of actors A ($|A| = k$). Each time an event occurs, one of the k actors should be selected for handling the event. Then, the selected actor moves toward the location of the event, handles the event and remains at the new position. The objective is to minimize the total distance traveled by all actors over a sequence of events. The important constraint in the problem is that the future events are not known in advance, and therefore, the actor selection is made without such knowledge.

Note that the greedy approach where the actor closest to the current event is always selected, is not necessarily optimal. In some cases, selecting a more distant actor is more efficient in terms of the total distance travelled by actors in a series of events. For instance, in Fig. 1 there are two actors and events are happening alternatively at points X and Y. Then a greedy solution continuously moves actor 1 while actor 2 remains still at point Z. A non-greedy algorithm would bring the actor 1 to point X and actor 2 to Y. So, no actor movement would be needed for handling the events happening at points X and Y.

Now, we present a distributed algorithm for event handling which employs the HA for selecting the actors which will handle events. The HA is a well-known approach for solving the k-server problem [2]. Upon event occurrence, this algorithm selects a server for handling the event with a certain probability. Specifically, assume that d_i is the distance of the i-th server ($i = 1 \dots n$) from the on-going event. Then the probability of the i-th server being selected is $p_i = \frac{1}{d_i^\alpha} / \sum_{j=1}^{n} \frac{1}{d_j^\alpha}$ where α is a small positive constant ($\alpha \geq 1$). Clearly, the servers close to the event are favored against servers located away from it. In this way, the average travel distance of servers is relatively small. Indeed, the average travel distance for handling an event is equal to $\sum_{i=1}^{n} p_i d_i$. For $\alpha = 1$, this sum is written as $n / \sum_{j=1}^{n} \frac{1}{d_j^\alpha}$, that is the harmonic mean of actor distances. Now, it is well known fact that the harmonic mean strongly tends to the lowest of d_i. This is more so in the case when $\alpha > 1$. At the same time, servers located far from the event may still be selected even with a small probability; hence, the algorithm is not trapped in scenarios similar to that of Fig. 1. Note that in the original description of HA, $\alpha = 1$. However, in a practical setting, by properly setting the value of a, we can leverage the greedy behaviour of the algorithm since higher values of a increases the probability of selecting an actor near the event.

The implicit assumption in the HA is that there is a central entity being constantly aware of the server locations and able to instantly know the exact location of the on-going event. Based on this information it decides which server will handle the event. In WSANs, such an assumption is not applicable because it would entail high communication cost. Therefore, the proposed distributed implementation should cope with the absence of the above central entity and should also provide for inaccuracies in actor and event locations as well as for the inherent delay incurred for actor location updates.

A high level description of the proposed distributed event handling protocol is given in Fig. 2. Note that at the initialization phase of the protocol, each SN learns its *nearest* actor by having actors broadcast hello messages and SNs choose the actor being minimum number of hops away. Assuming dense sensor networks, the actor with minimum hop distance from a SN is also the geometrically closest to that node. In the sequel, the basic steps are described in detail.

Step A: Event notification of actors. Each SN around an event notifies its nearest actor possibly through multihop communication. Thus, actors are notified with low delay and energy cost. Essentially, the SN-actor communication is an anycast where SNs just need to contact any of the nearby actors. Should there be a concern about network congestion or reduced reliability, our shortest path based communication between SNs and actors could be easily replaced with other anycast protocols (e.g. those in [7], [13]) which consider these issues.

A. Each SN sensing an event sends a report to its nearest actor

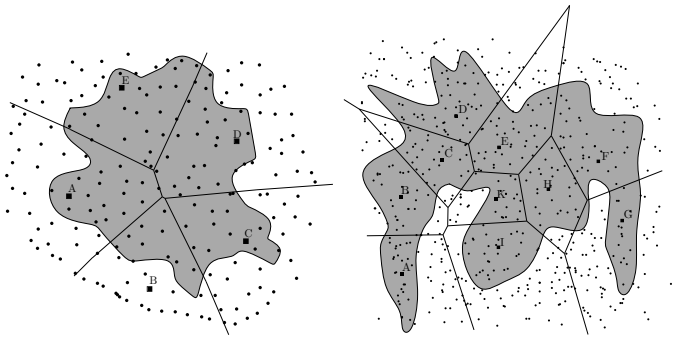
B. The actors notified about the event decide which of them will execute the HA

C. The selected actor runs the HA and then notifies the actor that should handle the event

D. The actor which receives the notification:

- **i.** broadcasts a 'leave' message to all SNs nearest to that actor. Each of these SNs finds the nearest of the remaining actors in the local area and thus all future event reports from these SNs are sent to the newly selected actors.
- **ii.** moves to the location of the on-going event.
- **iii.** updates all actors by broadcasting its new position.
- **iv.** asks all neighboring actors at its new position to send it their local cache with actor locations and then updates its own local cache
- **v.** sends a 'hello' message to all SNs around its new position so that SNs nearest to it will send their future event reports toward the newly arrived actor.
- **vi.** handles the on-going event and then stays at its new position until it is selected again for handling a new event.

Fig. 2. The Distributed Event Handling Algorithm



(a) Single HA invocation. (b) Multiple HA invocations.

Fig. 3. Different actors notified about an event.

Step B: Actor selection for the HA execution.

In large-scale events, the SNs inside the event area may notify more than one nearby actors. Next, if all these actors run the HA, each of them may assign a different actor for event handling. Apart from the likely inconsistencies in actor locations held by each actor, the different decisions are mainly due to the probabilistic nature of the algorithm. In that case, more than one actors will relocate to handle the on-going event. Thus, a coordination procedure among these actors is necessary to ensure that only one of the them will run the HA. The coordination protocol is based on the geometric information that each actor acquires from the SNs that notified it about the event. For instance, Fig. 3(a) depicts an event spread over the grey area, with five actors located within this area. The illustrated cells of the Voronoi diagram partition all SNs in the grey area according to their nearest actor. Note that each actor can approximately compose the event area inside its Voronoi cell from the location information received from its SNs.

Now, an easily implemented criterion for choosing the actor that will run the HA, is to select the actor whose area includes the southmost point of the event area. In a straightforward approach, each actor should exchange its southmost point (local minimum) with all actors located within the event area for finding the actor with the southmost point. This is essentially a leader election protocol with relatively high message (energy) cost and possibly long execution delay. Note also that when the ad-hoc network of actors is disconnected in the event area, the selection of more than one actors for running the HA is unavoidable regardless of the leader election protocol in use.

However, in most cases we can avoid the execution of a "heavy" leader election protocol if we can afford the selection of more than one actors for running the HA with small probability. Specifically, if the event area is convex, each actor needs to exchange its local minimum only with its neighboring actors (being in the transmission range of each other) since for convex areas, a local minimum point in the area is also a global minimum point for the whole area. Even in the case of a non convex event area, the above technique may as well be successful. Since the actors are typically far less than the SNs, the area supervised by each actor is relatively large. As a result, ignoring local concave segments, the whole event area may again be viewed as convex on the large scale. For instance, the event area of Fig. 3(a) is not convex. Nevertheless, in order to find the southmost point, the actors should only compare their local minimum against those of their neighboring actors since the small 'gulfs' existing in the area do not affect convexity on a large scale.

Now, Fig. 3(b) depicts a scenario where there may be multiple invocations of HA. Here, we assume that actors A and G cannot communicate. As a result, actor G falsely assume that holds the southmost point and thus the HA will run on actor G in addition to actor A. However, due to relatively large transmission range of actors, most actors in the local event area can communicate directly with one another and thus the case of two actors not able to communicate directly is rather uncommon. Also, large-scale irregularities as those in Fig. 3(b) mainly arise when the event has spread over a large area. However, the SNs continuously monitor their area and the actors are notified in time before the event is spread over a large area. Thus, long segments in the event area are not common. The following lemma precisely provides the necessary conditions for multiple HA invocations:

Lemma 1: Let A_i, $i = 1, \ldots, m$, be the actors notified about the current event, p_i be the southmost point of the area supervised by A_i, $i = 1, \ldots, m$, and p be the southmost of all points p_i, $i = 1, \ldots, m$. Consider the digraph $G(P, E)$ where $P = \{p_i : i = 1, \ldots, m\}$ and $(p_i, p_j) \in E$ if A_i and A_j can directly communicate, and p_i is south of p_j. Then the number of HA invocations equals the number of components created by a Depth First Search (DFS) traversal of G rooted at p.

Proof: The fact that DFS from p cannot reach all nodes of G implies that the G is disconnected. Each time DFS restarts, we select again the southmost of all remaining points of the graph and run the DFS from this point. Each of these points from which DFS restarts dominates over all other points in its connected component. Also, these points are not connected with one another in G. As a result, the corresponding actors falsely assume that have the southmost point of the entire event area. Therefore, each of these actors will run the HA. ■

Note that we can easily reduce the probability of multiple HA invocations, by having each actor exchange its southmost point with all actors within k hops of it instead of exchanging only with its immediate neighbors. Apparently, by increasing k, the above probability is reducing and for large value of k with finally get the general leader election protocol discussed above.

Step C: Notification of the event handling actor. For convenience, assume that HA runs on a single actor, say, actor A. The HA will decide the handling actor, say, actor B, and then A should send a notification to actor B about this selection. For routing the notification from A to B, a geographical routing protocol [16] is employed over the ad-hoc network of actors. In this protocol, at each step, the next node to send the message is the neighbor closest to the final destination. We have adopted the basic greedy approach without the relatively complex fallback technique of perimeter routing in the case of network voids. Thus, when the greedy approach gets stuck at a void, the last node holding the notification packet takes over the event handling from the actor B. Note that the positions of B and its substitute are expected to be close together. This is because the node A running the HA is close to the event, the HA mostly favours the actor located near the event and the substitute of B is met on the way to B. So, due to the relatively nearby positions of these two actors, this change in the handling actor only marginally deviates from the "typical" execution of the HA.

Finally, if we consider the ad hoc network N of the actors where an arc between two actors exists when both are within transmission range of each other, then we can easily see that for deciding the handling actor, the actor A needs to consider only the actors belonging to the same connected component of N as A. Also, as discussed in the following step, each actor is at least aware of the locations of all actors in the same connected component.

Step D: Event handling and actor location update. Apparently, the successful implementation of the HA depends on the accuracy of actor positions stored in the actor running the algorithm. Since the only change in the network is the movement of a single actor toward the current event, the network of actors can be considered relatively stationary. Furthermore due to the relatively small number of actors in a WSAN, a simple broadcasting protocol is employed for updating the actors of the network with the new location of the handling actor. Specifically, before moving to the event location, the handling actor checks if it is going to be disconnected from its current connected component by the completion of its relocation[1]. In that case, it broadcasts its new location before leaving its current position so that all actors in its former component are informed about its new position. After its movement, the handling actor broadcasts its new location and asks from all neighboring actors at its new location to send it their local cache of other actors' positions.

[1]Provided that the transmission range of all actors is the same and if no obstacles prevent the communication of nearby actors, this can be easily done based on the locations of actors and checking if the new actor location is out of transmission range of all actors in the current connected component.

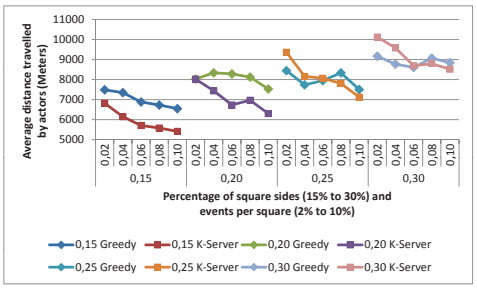


Fig. 4. Average distance travelled by each actor to serve events.

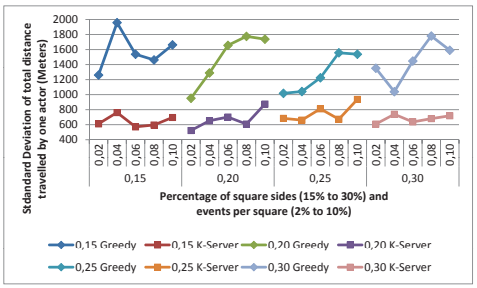


Fig. 5. Standard deviation of the distance travelled by each actor to serve events.

After receiving that information, the handling actor keeps the most recent information in its local cache. Obviously, when the handling actor is still connected to its original connected component, the first broadcast and the cache update protocol are not actually necessary and only the second broadcast needs to be performed. Thus, the above update scheme not only ensures that the new location of the event-handling actor is made known to the maximum possible number of actors, but the handling actor is made aware of the most recent locations of other actors in the network too.

Besides the actors, SNs should also be notified about the relocation of the event-handling actor. Thus, when this actor leaves an area (step D.i), it broadcasts a leave message to all SNs in its surrounding area. Each SN receiving this message first checks if the departing actor is the nearest one. In that case, it discards all local information kept for that actor and then relays the message to all its neighbors except the one it received the message from. When a SN receives a leave message not originated from its nearest actor, this SN discards the received message and replies back to the message sender with the ID of its nearest actor and the number of hops for reaching that actor. Each time, an "orphan" SN receives such a reply, it checks whether the new route to an actor is the shortest so far. In that case, it updates its local information and then forwards the newly updated data to its neighbors except the one it received the reply from. Thus, the SNs formerly closest to the departing actor are now associated with the nearest among the remaining actors in the local area. When the event-handling actor arrives at the event area, it brodcasts a 'hello' message to the SNs of the area (step D.v). Each recipient SN updates its local information if the newly arrived actor is the nearest one in the area. Each SN which associates with the new actor, it then sends the updated information to its neighbors.

Finally, the newly arrived actor at the event area handles the on-going event. However, due to possible multiple HA invocations, more than one actors may arrive at that area. Each of these actors sends beacons to nearby actors at the event location and the actor with the smallest ID handles the event eventually. Thus, apart from the unnecessary actor movement, the selection of more than one actors for event handling does not affect the efficacy and the correctness of the protocol.

IV. Experimental Results

In order to evaluate the performance of our algorithm we have implemented it in the Castalia simulator [17]. We also implemented the greedy algorithm, where every event is served by the actor nearest to the event. As has been mentioned, the involvement only of actors near the event is typical of other works on WSANs. The field where events occur has been defined to be a square $500m \times 500m$. Inside the square, 500 sensors have been randomly deployed. The number of actors used to serve the events has been set to 20. At the beginning of each simulation run, the actors are positioned on a 5×4 grid. A total number of 5000 events were simulated, with events occuring at a 500 second time interval. After a number of experiments, we also set the parameter α of HA to 3, as HA achieves the best overall performance for this particular value. Also, at step B (Fig. 2) we implemented the simple scheme where each actor contact only its neighbors for finding out whether it holds the southmost point of the event area. We conducted a number of experiments where the employed scenarios follow a certain pattern. Specifically, we consider that the sequence of events is partitioned into $1/p$ subsequences of $5000 \cdot p$ events each, where p is a percentage value varying in the range of 2% to 10%, in steps of 2%. The events of a specific subsequence all take place randomly within a smaller square region of the deployment area and these squares are different for each subsequence. The position of each square is also selected randomly. This kind of scenarios is common since when an initial event occurs in a region a lot of other related events will likely occur at nearby locations.

We have focused on both the energy required by SNs and actors for communication as well as on the mechanical energy required for moving actors to handle events. The first is measured directly by Castalia while mechanical energy consumption is inferred from the travelling distance covered by the actors. Fig. 4 presents the average distance travelled by each actor to serve the events. Besides being an indicator of the mechanical energy spent by each actor, the average distance, if multiplied by the factor (number of actors/number of events), also indicates the average time required to serve an event, assuming that actors move with a constant speed. Note also that the delay due to actor movement is much higher than the delay due to network communication. Figure 4 shows that our algorithm performs better when considering smaller square areas. For larger square sides, our algorithm is worse for small numbers of events in each square, but outperforms again the greedy algorithm for more events. This behavior is expected. As more events happen in a specific area, the cost of moving more actors initially further away into that area pays off eventually. These actors now have to travel shorter distances to handle future events. Next, we study the standard deviation of the distance travelled by each actor (Fig. 5). This metric shows how balanced the load of actors for serving events is.

Fig. 6. Average consumed energy per sensor. The side of the event square is 20% of the field side. The number of events per square is 10% of the total number of events.

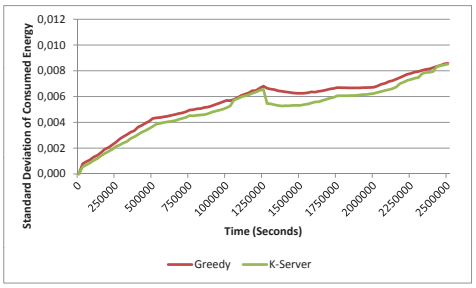


Fig. 7. Standard Deviation of average consumed energy per sensor. Same parameters as in Fig. 6.

Our algorithm clearly outperforms the greedy algorithm in all cases. This is due to the random nature of our algorithm, which might select an actor to serve an event, even if that actor is further away from the event location. However, this brings one more actor to the neighbourhood where events happen and improves the behaviour of our algorithm when serving the subsequent events. Note also that the graphs for our algorithm are smoother compared to the ones of the greedy algorithm. This is due to the tendency of the greedy algorithm to leave many actors inactive. That depends on the initial position of the actors and the coordinates of the first events in each square; an actor may move to a specific area to serve an event and the next events may be closer to that actor. In such case, this actor will serve more events, covering a much larger distance compared to other actors. Once again the random nature of our algorithm prevents most of these cases.

Regarding the energy consumption of SNs due to communication, the algorithms have similar performance (Fig. 6), since both algorithms use the same method for sensor-actor communication, for determining which actor will run the selection algorithm, for routing the notification packet to the selected actor and for actor location update. Fig. 7 present the standard deviation of the average consumed energy per SN. Clearly, both algorithms have comparable performance and follow the same trend. Summing up, our algorithm requires much less energy than the greedy approach, when considering both the communication and the mechanical energy.

V. Conclusions and Future Work

We presented a protocol for event handling in WSANs. By randomization, the protocol efficiently handles worst cases in the spatial distribution of future events. In contrast, most previous works act greedily and consider only local actors around the current event for event handling. Thus, in practice, our protocol guarantees low handling delay and fair distribution of actor workload, resulting in balanced energy consumption across actors. As a future work, we will study a different actor location update protocol where only actors near the trajectory of the moving actor will receive location updates. That change will not much affect the overall performance of our approach.

References

[1] H. Salarian, K.-W. Chin, and F. Naghdy, "Coordination in wireless sensor actuator networks: A survey," *Journal of Parallel and Distributed Computing*, vol. 72, no. 7, 2012.

[2] A. Borodin and R. El-Yaniv, *Online computation and competitive analysis*. Cambridge University Press Cambridge, 1998.

[3] Y. Bartal and A. Rosn, "The distributed k-server problem - a competitive distributed translator for k-server algorithms," *Journal of Algorithms*, vol. 23, no. 2, pp. 241 – 264, 1997.

[4] E. Ruiz-Ibarra and L. Villasenor-Gonzalez, "Cooperation mechanism taxonomy for wireless sensor and actor networks," *Ad Hoc & Sensor Wireless Networks, An International Journal*, vol. 7, no. 1–2, pp. 91–113, 2009.

[5] I. Mezei, M. Lukic, V. Malbasa, and I. Stojmenovic, "Auctions and imesh based task assignment in wireless sensor and actuator networks," *Computer Communications*, vol. 36, no. 9, pp. 979–987, 2013.

[6] T. Melodia, D. Pompili, V. C. Gungor, and I. F. Akyildiz, "Communication and coordination in wireless sensor and actor networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 10, pp. 1116–1129, 2007.

[7] T. Melodia, D. Pompili, and I. F. Akyldiz, "Handling mobility in wireless sensor and actor networks," *IEEE Transactions on Mobile Computing*, vol. 9, no. 2, pp. 160–173, 2010.

[8] K. Selvaradjou and C. Murthy, "On maximizing residual energy of actors in wireless sensor and actor networks," in *Proc. 8th International Conference on Distributed Computing and Networking (ICDCN'2006)*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, vol. 4308, pp. 227–238.

[9] E. C. Ngai, M. R. Lyu, and J. Liu, "A real-time communication framework for wireless sensor-actuator networks," in *Proc. IEEE Aerospace Conference*, 2006, pp. 9 pp.–.

[10] G. A. Shah, M. Bozyigit, and F. B. Hussain, "Cluster-based coordination and routing framework for wireless sensor and actor networks," *Wireless Communications and Mobile Computing*, vol. 11, no. 8, pp. 1140–1154, 2009.

[11] K. Akkaya, F. Senel, and B. McLaughlan, "Clustering of wireless sensor and actor networks based on sensor distribution and connectivity," *Journal of Parallel and Distributed Computing*, vol. 69, no. 6, pp. 573–587, 2009.

[12] K. Akkaya, I. Guneydas, and A. Bicak, "Autonomous actor positioning in wireless sensor and actor networks using stable-matching," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 25, no. 6, pp. 439–464, 2010.

[13] E. Ngai, Y. Zhou, M. R. Lyu, and J. Liu, "A delay-aware reliable event reporting framework for wireless sensor–actuator networks," *Ad Hoc Networks*, vol. 8, no. 7, pp. 694–707, 2010.

[14] I. Mezei, V. Malbasa, and I. Stojmenovic, "Auction aggregation protocols for wireless robot-robot coordination," in *Proc. Ad-Hoc, Mobile and Wireless Networks (ADHOC-NOW'2009)*, ser. Lecture Notes in Computer Science, vol. 5793. Springer Berlin Heidelberg, 2009, pp. 180–193.

[15] Y. Zeng, D. Li, and A. V. Vasilakos, "Real-time data report and task execution in wireless sensor and actuator networks using self-aware mobile actuators," *Computer Communications*, vol. 36, no. 9, pp. 988–997, 2013.

[16] F. Cadger, K. Curran, J. Santos, and S. Moffett, "A survey of geographical routing in wireless ad-hoc networks," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 2, pp. 621–653, 2013.

[17] The Castalia simulator for Wireless Sensor Networks: http://castalia.research.nicta.com.au.