2009 Fifth International Conference on Wireless and Mobile Communications

A Real-time Architecture for Automated Wireless Sensor and Actuator Networks

Yuanyuan Zeng
Department of Computer Science,
University College Cork,
Cork, Ireland
yz2@cs.ucc.ie

Cormac. J. Sreenan
Department of Computer Science
University College Cork,
Cork, Ireland
cjs@cs.ucc.ie

Guilin Zheng
School of Power and Mechanical Engineering,
Wuhan University,
Wuhan, China
glzheng@whu.edu.cn

Abstract—**Wireless sensor and actuator network (WSAN) are composed of a large number of heterogeneous sensors and actuators. In the automated WSAN, sensors are collaborated to monitor the physical phenomenon in the surveillance field, while the actuators are to collect sensing data, process the data and perform appropriate actions without the existence of central controller. Most WSANs are used in the real-time sensing and reaction systems towards physical environment. In this paper, we propose a real-time architecture for automated WSAN to bind the latency in applications. In the architecture, we present distributed mechanisms for sensor-actuator event reporting and self-aware coordination to maintain the delay bound sensor-actuator communication. And then we present mechanism for ordered multi-event task assignment, and the acting coordination mechanism to provide efficient reaction and execution of the event task in time. Preliminary simulation results are presented to demonstrate the advantages of our solutions.**

Keywords-real-time; Wireless sensor and actuator Networks; communication; coordination; automated architecture

I. INTRODUCTION

A wireless sensor and actuator network is a group of sensors and actuators that are geographically distributed and interconnected by wireless networks. Sensors gather information about the state of the physical world, and the actuators react to the information by performing appropriate actions. WSANs are widely used in applications such as home automation, environmental monitoring, microclimate control, fire handling system, and battlefield surveillance, etc. In automated WSANs, sensor-actuator and actuator-actuator communication and coordination may occur autonomously during data collection and the process of performing changes when detecting events. In many situations, sensors are usually low-cost, low-power and small devices equipped with limited sensing, data processing and wireless communication capabilities, while actuators typically have stronger computation and communication powers and more energy budget that allows longer battery life, and they could be mobile. Regardless, resource constraints apply to both of them. Depending on the application there may be a need to make a repaid respond towards the physical world, such as in fire handling systems, and other environment monitoring and handling systems etc. Therefore, the issue of real-time communication is very important in WSANs. Considering the design constraints of limited power, network dynamics, heterogeneity and scalability, protocols and algorithms proposed for WSNs may not well-suited for the unique features and application requirements of WSANs.

Our goal in this paper is to propose a real-time architecture which provides timely reporting, reaction and execution to the environments upon the detection of an event, autonomously. The remaining of this paper is organized as follows. Section 2 presents the related work. In section 3, we outline the network module and architecture. We present sensor-actuator event reporting and self-aware coordination in section 4. And in section 5, we present the actuator-actuator coordination mechanisms. The simulation is described in section 6. Finally, section 7 concludes this paper.

II. RELATED WORK

Real-time communication problems in WSNs are not new. He et al [1] proposed an outstanding real-time communication protocol binding the end-to-end communication delay by enforcing a uniform delivery velocity. Felemban et al proposed [2] a novel packet delivery mechanism called MMSPEED for probabilistic QoS guarantee. Chipara et al proposed [3] a real-time power aware routing protocol by dynamically adapting transmission power and routing decisions. Although a number of protocols are proposed for WSN, they may not work well when applying directly to WSAN [4, 5]. Stankovic et al [6] highlighted the state of the art and present many open research questions that must be solved. Shah et al [7] proposed a real-time coordination and routing framework that addresses the coordination of sensors and actuators and respects the delay bound for routing in a distributed manner. Durresi et al [8] presented a geometric broadcast protocol for WSAN. Hu et al [9] developed an anycast communication paradigm that can reduce both end-to-end latency and energy consumption. Boukerche et al [10] proposed a routing protocol with service differentiation for WSANs, which provides low latency and reliable delivery in the presence of failures. Melodia et al [11] presented the first distributed coordination framework for wireless sensor and actuator network to made tradeoff

978-0-7695-3750-4/09 $26.00 © 2009 IEEE
DOI 10.1109/ICWMC.2009.8

Authorized licensed use limited to: UNIVERSITY COLLEGE CORK. Downloaded on January 28, 2010 at 09:37 from IEEE Xplore. Restrictions apply.

between energy consumption and delay for data transmission; and discussed the sensor-actuator and actuator-actuator coordination problems. But the authors only focused on scenarios with immobile actuators that can act on a limited area defined by their action range. Nagi et al [12] designed a real-time communication framework that supports event detection, reporting, and actuator coordination.

Different from previous work, we aim to present a real-time architecture from the application aspect, and to bind the application time by several sub deadlines in each phase of the applications. Few of the relevant work considers the energy efficiency and reliability of the actuators when in coordination, which is also resource limited; as well as effective self-aware sensor-actuator and actuator-actuator coordination by using the actuator mobility.

III. AUTOMATED NETWORK ARCHITECTURE

Each sensor and actuator has a unique id number, and knows its own location, which can be obtained at a low cost from Global Positioning System or location discovery algorithms. Each sensor and actuator can know its neighbors' information by periodic message exchange. Each actuator knows its residual energy. We assume the network is synchronized by means of one of the existing synchronization protocols. The actuators are equipped with two radio transmitters, i.e., a low data rate transmitter to communicate with sensors, and a high rate wireless interface for actuator-actuator communication.

We adopt the deadline partition to divide the application deadline into multiple sub deadlines in different phases. As long as the individual sub deadlines are met, we can guarantee the delay of the whole application process. We define the application delay as bound *D* that is the maximum allowed time between the instant when the physical features of the event area sampled by the sensors till the actuators finish the destined event in the area to make a change. We try to generally divide application-specific bound *D* into several sub-deadlines: *D1* is the delay bound for end-to-end event reporting from sensor to actuator; and *D2* is the delay bound for self-aware sensor-actuator coordination. And then *D3* in the delay bound for actuator negotiation to make task assignment. *D4* is the delay bound for actuator-actuator coordination process to make reaction and finish the event. The bound *D*, *D1*, *D2*, *D3*, *D4* are all parameters dependent on the application requirements.

IV. SENSOR-ACTUATOR COORDINATION

A. Sensor-Actuator Event Reporting

The designed event reporting algorithm could be combined with the existing routing algorithms in sensor-actuator communication. Each actuator takes responsibility of collecting data from local sensors by dividing the surveillance field into clusters, i.e., in each cluster, the actuator acts as a cluster head and local sensors act as cluster members. Each actuator a_k is assigned with a weight function according to a specific local sensor as weight(s_i, a_k), which decided by the distance between s_i and a_k, i.e., *dis*(s_i, a_k), and the residual energy of a_k, i.e. *E*(a_k), as well as the load of a_k, i.e., *load*(a_k). The load can be calculated by the packets received within a certain recent interval. Each actuator periodically beacons its position, residual energy and load among its neighborhood. The higher weight of an actuator represents that it incurs less delay, better energy efficiency and more reliability. So the weight function is defined as:

$$weight(s_i, a_k) = \frac{\sqrt{E(a_k)}}{dis(s_i, a_k) * \sqrt{load(a_k)}} \quad (1)$$

So we could use weight function of actuators as a metric to make field partition to form each cluster *cluster*(a_k) associated with the actuator a_k. If there is a tie, the actuator with lower id wins, where: *cluster*(a_k)={for any sensor $s_i \in$ cluster(a_k)|*weight*(s_i, a_k)≤*weight*(s_i, $a_{k'}$), OR *weight*(s_i, a_k)=*weight*(s_i, $a_{k'}$), *id*(a_k)< *id*($a_{k'}$)} Initially, all actuators are with the same energy level and load (i.e., zero load), so the field partition is a Voronoi graph as shown in figure1.

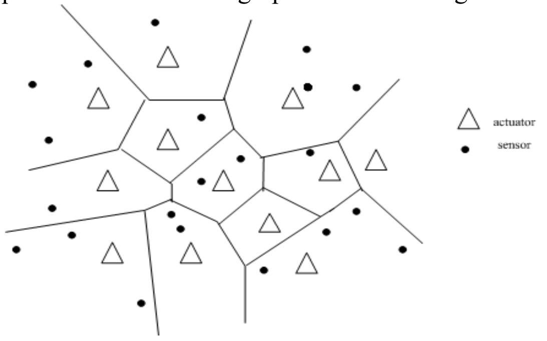

Figure 1. Initialized network clustering based on Voronoi graph

To bind the end-to-end delay in event reporting, the sensor adds a *slack1* filed recording the time left in the current phase to its data packet that is set to *D1* by the source sensors. The *slack1* filed will be updated in sensor-actuator communication hop by hop. We use *Δ*(*e*) to represent the delay on the outgoing link *e* in routing from sensor to actuator. We could estimate the *Δ*(*e*) as: $\Delta(e)=(delay_{tran}+delay_{cont})*R$, where $delay_{tran}$ is a constant determined by packet size and network bandwidth, and $delay_{cont}$ is contention time, and then *R* is transmission count. A source sensor s_i starts the event reporting algorithm by sending out a *DetectEvt*(*sensor_id*, *hop_num*, *slack1, ev_id*, *actuator_id*) message, in which the *actuator_id* is the id of the actuator chosen by the initialized Voronoi graph. Then the *DetectEvt* message is forwarded out by existing routing protocol, such as GPSR. During the forwarding, if the slack time is enough, i.e., *slack1>=Δ*(*e*), then the packet is forwarded to the next hop. When in packet forwarding process, if the relaying sensor finds another actuator with higher weight, then the packets will be routed towards the

Authorized licensed use limited to: UNIVERSITY COLLEGE CORK. Downloaded on January 28, 2010 at 09:37 from IEEE Xplore. Restrictions apply.

better actuator. When the destination actuator receives the data packets, it will reply a *ReplyEvt* message backwards. Otherwise, the packet will be expired, and then a *ReplyEvt* is notified backwards, and a *FindActuator* message will be broadcasted out by the current sensor to invoke the self-aware sensor-actuator coordination (in section 4.2).

B. *Self-Aware Sensor-Actuator Coordination*

When a sensor firstly sends out a *FindActuator* message, the self-aware sensor-actuator coordination is invoked to find an alternative actuator nearby to move and make data collection within the bound *D2*. We call the sensor as sensor-actuator coordination invoker. To provide delay-bound coordination, we add *slack2* into *FindActuator* message body to record the time left for coordination. The *slack2* will be updated by local timestamp and encapsulated into message during the *FindActuator* message forwarded hop by hop. When a sensor with a reporting actuator nearby receives a *FindActuator* message, it will calculate to make decision and notify the actuator nearby to move and make data collection, if the time left is enough. And then a *GetActuator* message will be sent out backwards by the sensor. Otherwise, if a nearby actuator receives the *FindActuator* message and the time left is enough to move to the invoker position, and then a *GetActuator* message will be forwarded backwards. The moving velocity will be dependent on the distance to specific collecting position and time left for coordination recorded in *slack2*.

V. ACTUATOR-ACTUATOR COORDINATION

A. *Event Task Assignment*

When an actuator gets the last data packet of an event from local sensors, it broadcasts out a *Negotiate*(*actuator_id*, *Event_id*, *Event_priority*) message, and we call this actuator as negotiation invoker. The *Event_id* is the id of the event that the actuator collects; and the *Event_priority* is an application-specific identity used to discern the urgent degree of each event, such as "very high", "high", "medium", "low", and "very low". The event priority is predefined according to the application requirements. The node that receives a *Negotiate* message will reply an *ActReply*(*actuator_id*, *energy*, *location*, *Event_id*, *Event_priority*) message with its own id, event id and priority. The negotiation is executed in the local area of the actuator invoking the process. Multiple sensors placed in the monitoring field could be used to detect the same event, so one or multiple actuators will get the sensing data of the exact event. Through the process of backward *ActReply* message, the data of the same event (with the same *Event_id*) will be aggregated during the backward process and forwarded back to the first actuator that invokes the negotiation. If there is more than one event sensed by an actuator, then after the data aggregation for the same event, a multi-event ordered task assignment should be involved. The task assignment on this aggregation actuator will be executed according to the event priority, i.e., the event with higher *Event_priority* rank will win. If there is a tie of the event priority, the event with lower *Event_id* will win. Beyond the above, the aggregation actuator will get to know the whole area and location of the detected event through multi-actuator communication. The event area could be represented by using the left upside coordinate (Ev_X_i, Ev_Y_i) and the right downside coordinates (Ev_X_j, Ev_Y_j) approximately. During negotiation, we could combine each sub event area gathered by each actuator into the whole event area.

B. *Action Mechanism*

We assume all actuators have the same type of communication and acting hardware, i.e., they have the same maximum acting range $ActR_a$. Let $G=\{g_1, g_2, \ldots, g_m\}$ be the set of all event grids in surveillance field. Let g_i be the location vector for an event grid g_i. Let $d_{i,k}$ be the distance between g_i and the actuator a_k. $d_{i,k} = \| g_i - a_k \|$. We use an acting capability metric to express the acting coverage over the event area in the surveillance field. The actuator only can act on the event within its acting range, and it has better acting capability to the closer event. Let $p(g_i, a_k)$ describe the acting capability of the actuator toward the event occurs at g_i. $p(g_i, a_k)$ can be illustrated as follows:

$$p(g_i, a_k) = \begin{cases} f(d_{i,ak}, E_{ak}) & d_{i,ak} \le ActR_{ak} \\ 0 & d_{i,k} > ActR_{ak} \end{cases} \quad (2)$$

The acting capability of the actuator will depend on its distance toward the event and also its residual energy level. We could use the following function to represent the capability of the actuators in performing actions towards the physical world.

$$f(d_{i,ak}, E_{ak}) = \frac{1}{1 + \frac{\sqrt{d_{i,ak}}}{E_{ak}}} \quad (3)$$

Then the acting capability of the actuator set AS in specific g_i is represented as follows:

$$p(g_i, AS) = 1 - \prod_{a_k \in AS} (1 - p(g_i, a_k)) \quad (4)$$

Let *Ev_Area* be the specific event area. The acting capability of local actuator node set *AS* toward the event area *Ev_Area* is defined as the summation of the acting coverage over the target of *Ev_Area* from (Ev_X_{min}, Ev_Y_{min}) to (Ev_X_{max}, Ev_Y_{max}).

$$p(Ev\_Area, AS) = \int_0^x \int_0^y p(g_{i,} AS)\,dxdy \quad (5)$$

For the acting coverage requirement in surveillance field, an application-related threshold p_{th} is given to determine the requirement of acting capability degree for an application. *AS'* is subset of *AS* (i.e., the set of local actuators that perform the task), so the nodes in *AS'* perform the acing task

Authorized licensed use limited to: UNIVERSITY COLLEGE CORK. Downloaded on January 28, 2010 at 09:37 from IEEE Xplore. Restrictions apply.

of event area with the acting capability should be no less than application desired p_{th}. So:

$$p(Ev\_Area, AS') \geq p_{th} \tag{6}$$

To make tradeoff between reliability and energy efficiency, we propose the concept of acting redundancy. For the actuator set *AS* that the acting range of actuators in *AS* can cover the whole event area, each actuator a_k when $p(Ev\_Area, a_k)>0$, will decide whether to attend the event execution by calculation of acting redundancy $\xi_{,k}$ as follows:

$$\xi_k = \frac{p(Ev\_Area, AS \setminus \{a_k\})}{p(Ev\_Area, a_k)} \tag{7}$$

When $\xi_k \geq 1$, it implies that the actuator a_k has no contribution to perform action at the specific event area. So this actuator will not attend the task execution for the event. Considering the reliability in the event reaction, we could invoke an application-specific ξ_{th} to make the existence of acting redundancy for an application.

The actuator will decide whether to attend task execution by the edibility rule as follows:

$$ACTING\_state(a_k) = \begin{cases} ture & \xi_k < \xi_{th} \\ false & \xi_k \geq \xi_{th} AND\, p_{AS'} > p_{th} \end{cases} \tag{8}$$

Otherwise, if local actuators cannot provide effective acting coverage because of distance or energy, then the negotiation invoker will sign to move the actuators into event area and select the velocity to perform the task in time. For *n* actuator in local *AS*, we divide the event area into *n* sub areas evenly. So each actuator moves into the centroid of each sub area. The sub-deadline for making actuator-actuator coordination is *D4* recorded by *slack4* and encapsulated in the message. Let $t_{k'}$ be the time the actuator $a_{k'}$ takes to arrive at the sub event area, and *r* be the rate of performing appropriate actions by actuators (*r* is defined by applications). The moving actuators should satisfy formula (9), so the moving velocity could be calculated.

$$r * \sum_{i=1}^{n} (B_4 - t_{k'}) = Ev\_Area \tag{9}$$

The maximal moving velocity is determined by the application. If the calculated velocity is larger than maximum, or the moving actuator set could not satisfy formula (9), the real-time event execution fails.

VI. Simulations

We report the simulation performance results when compared with the framework presented in [10]. The simulations were developed within the OMNET++ Simulator. We simulate the network scenario when 200 sensor nodes are randomly deployed in a square area of 200mX200m. The other simulation parameters are chosen as: the transmission range of sensors is set to 40m. The sensing range is set to 20m and the bandwidth to 250kbit/s. Sensors send 56byte long packets, and the size of the queues is set to 20 packets. The transmission range of actuator is sent to 80m, and the acting range of actuator is sent to 100m. The initial energy of all sensors is 1J, and all actuators with initial energy 2J. The communication power is set to 50nJ/bit, the moving power is set to 0.001W/m, and the action power is set to 0.001W/m^2. The event area is rectangle with varying ranging from 10mX10m till 100mX100m in different simulations. The centroid of the event area is randomly selected such that the event area completely falls into the terrain. According to our framework, the application completion bound *B* is set to 50s, and then each sub-deadline *D1*, *D2*, *D3*, *D4* in sub-phase is set to 10s, 10s, 10s, and 20s. The Routing is using GPSR protocol, and the MAC layer is based on CSMA/CA. We perform terminating simulations that last 200s, average over different random topologies. The phenomenon is generated by 5 events at 50s, 100s, and 150s, respectively.

A. Sensor-Actuator event Reporting

The event area is randomly deployed in a square area of 25mX25m in the surveillance field, while the number of actuators is increased from 4 to 12 in different simulations. In figure2, the residual energy of actuators of our algorithm is better than the closest-actuator event reporting. The improvement of energy efficiency of our algorithm is slightly higher than the closest-actuator event reporting as the number of actuators increase. In figure3, the results show that the packet delivery ratio with varying number of randomly deployed actuators, which is shown that our algorithm selecting actuators with less load will help to improve successful the sensor-actuator data packet delivery.

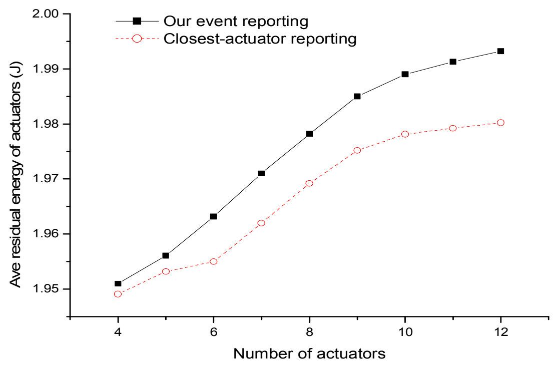

Figure2. Average residual energy of actuators with different number of actuators

Authorized licensed use limited to: UNIVERSITY COLLEGE CORK. Downloaded on January 28, 2010 at 09:37 from IEEE Xplore. Restrictions apply.

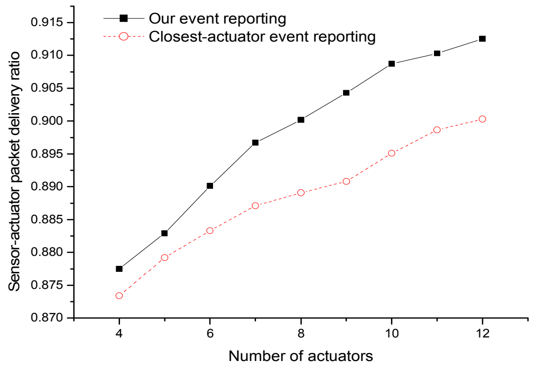

Figure3. Sensor-actuator packet delivery ratio with different number of actuators

B. *Self-Aware Sensor-Actuator Coordination*

We consider a scenario with homogeneous actuators randomly deployed in the 200mX200m, where event area is randomly deployed in a square area of 25mX25m. We set the coordination delay to 1s. Moreover, the maximal moving velocities of actuator is set to 12 m/s. Figure4 illustrates the deadline miss ratio in the sensor to actuator communication procedure as the number of actuators increases from 2 to 12. It is shown that the miss ratio of our algorithm is much less than the other two algorithms, because the alternative actuator could move and help to maintain an effective real-time sensor to actuator communication by self-aware sensor-actuator coordination process. The results in figure5 show our sensor-actuator communication and coordination is slightly better on residual actuator energy. Because the actuator will move and consume energy, the advantage of communication with self-aware sensor-actuator coordination is not obvious when the actuator number is relatively small. And then as the actuator number increases enough, our communication algorithm has better energy efficiency for actuators because we consider the energy in clustering process by hybrid clustering.

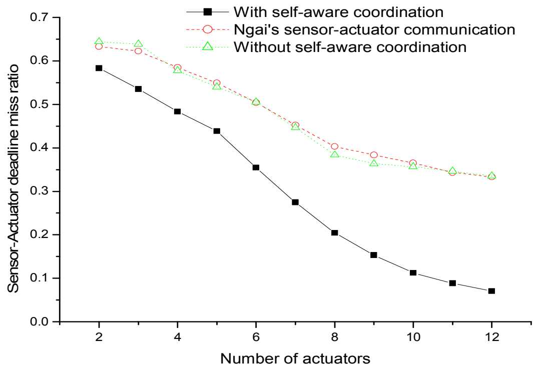

Figure4. Sensor-actuator communication miss ratio with different number of actuators

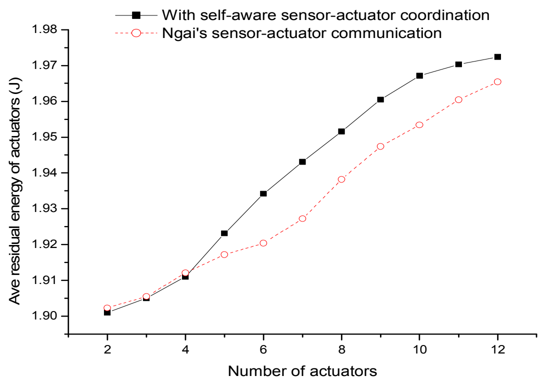

Figure5. Average residual energy of actuators with different number of actuators

C. *Actuator -Actuator Communication and Coordination*

In Ngai's framework, the coordination involved actuators determine which of them will perform the appropriate actions, and they simply select the actuators located closer to the event to move to the event area and react with the same speed (we set the speed to 6m/s in simulations) for moving and reacting to the event. The simulation results in figure6 show our paradigm has less miss ratio than Ngai's real-time framework, because we consider the effective acting range and acting capability for each actuator and then to make coordination decision toward the event area to provide enough acting coverage in time. Figure7 shows the residual energy of actuators of our paradigm and Ngai's framework as the number of actuators increases. The results illustrate that our coordination has better energy efficiency for actuators, because we consider both acting range and energy level that actuators could provide in the multi-actuator coordination process.

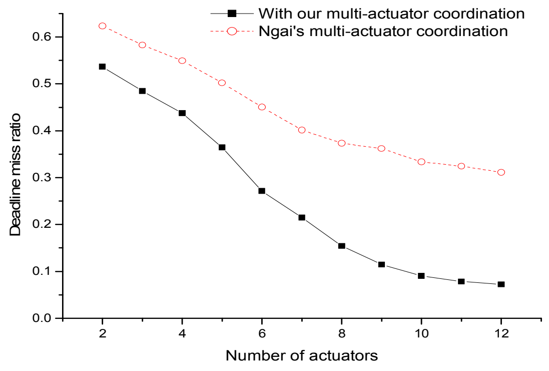

Figure6. Deadline miss ratio with different number of actuators

Authorized licensed use limited to: UNIVERSITY COLLEGE CORK. Downloaded on January 28, 2010 at 09:37 from IEEE Xplore. Restrictions apply.

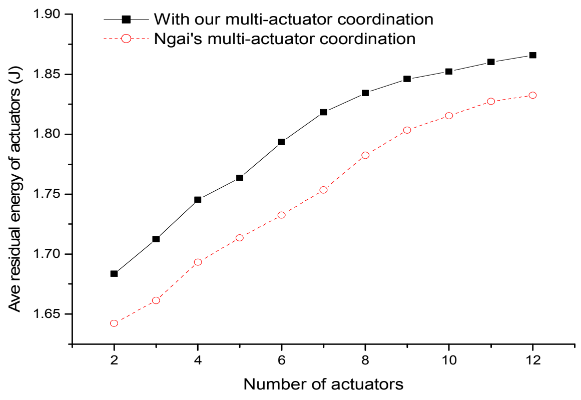

Figure7. Average residual energy of actuators with different number of actuators

VII. CONCLUSION S

We present a soft real-time architecture for automated wireless sensor and actuator networks. We consider the heterogeneous characteristics and functionalities of sensor and actuators, and offer a distributed, self-organized and comprehensive solution for real-time energy-efficient communication and coordination mechanisms in WSAN. It provides efficient sensor-actuator event reporting algorithm considering the delay, energy and load for actuators. Then a self-aware sensor-actuator coordination requirement mechanism is presented to find the actuator nearby to move and collect sensing data in time. The actuator-actuator negotiation is designed to make ordered task assignment for multiple events with different priorities. And then the multi-actuator makes coordination to select actuator and finish the task in time.

ACKNOWLEDGMENT

The work is supported by the NEMBES project funded by the Irish Higher Education Authority under the PRTLI-IV programme.

REFERENCES

[1] T. He, J. Stankovic, C. Lu, and T. Abdelzaher, “SPEED: A real-time routing protocol for sensor networks,” IEEE ICDCS, May 2003, pp. 46-55.

[2] E. Felemban, C.-G. Lee, E. Ekici, R. Boder, and S. Vural, “Probabilistic QoS Guarantee in Reliability and Timeliness Domains in Wireless Sensor Networks,” IEEE InfoCom, May 2005.

[3] O. Chipara, X. H. Zhimin, X. Guoliang, C. Qin, W. Xiaorui, et al, “Real-time power-aware routing in wireless sensor networks,” 14th IEEE International Workshop on Quality of Service, June 2006.

[4] X. Feng, “QoS Challenges and Opportunities in Wireless Sensor/Actuator Networks,” Sensors, 2008, 8(2), pp. 1099-1110.

[5] D. V. Dinh, M. D. Vuong, H. P. Nguyen, H. X. Nguyen, “Wireless sensor actor networks and routing performance analysis,” International workshop on Wireless Ad-hoc Networks, May 2005.

[6] J. A. Stankovic, T. Abdelzaher, C. Lu, L. Sha, and J. Hou, “Real-time Communication and Coordination in Embeded Sensor Networks,” IEEE, 2003,91(7), pp. 1002-1022.

[7] G. A. Shah, M. Bozyigit, O.B. Akan, B. Baykal, “Real-Time Coordination and Routing in Wireless Sensor and Actor Networks,” 6th International Conference on next Generation Teletraffic and Wired/Wireless Advanced networking (NEW2AN), Lecture Notes in Computer Sciences, 2006, 4003, pp. 365-383.

[8] A. Durresi and V. Paruchuri, “Geometric broadcast protocol for sensor and actor networks,” International Conference on Advanced Information Networking and Applications, mar 2005.

[9] W. Hu, N. Bulusu, S. A. Jha, “Communication Paradigm for Hybrid Sensor/Actuator Networks,” International Journal of Wireless Information Networks, 2005, 12(1), pp.47-59.

[10] A. Boukerche, R. B. Araujo, L. A. Villas, “Wireless Actor and Sensor Networks QoS-Aware Routing Protocol for the Emergency Preparedness Class of Applications,” 31st IEEE Conference on Local Computer Networks, 2006.

[11] T. Melodia, P. Dario, V. C. Gungor, and I. F. Akyildiz, “A Distributed Coordination Framework for Wireless Sensor and Actor Networks,” MobiHoc’05, May 2005.

[12] H. C. E. Ngai, R. M. Lyu, J. Liu, “A Real-time Communication Framework for Wireless Sensor-Actuator Networks,” IEEE Aerospace Conference, March 2006.

Authorized licensed use limited to: UNIVERSITY COLLEGE CORK. Downloaded on January 28, 2010 at 09:37 from IEEE Xplore. Restrictions apply.