A Novel Self Organizing Framework for SANETs

Muhammad Farukh Munir and Fethi Filali
Institut Eurécom, Department of Mobile Communications, Sophia Antipolis, France
Email: {munir,filali}@eurecom.fr

Abstract - **Sensor application requirements vary tremendously, ranging from densely deployed habitat-monitoring setup to the real-time constrained military applications. This paper presents the design of a novel self-organizing framework for SANETs (Sensor and Actuator Networks). We assume two wireless interfaces at the actuator nodes, one to communicate with sensor nodes, and the other, for the neighboring actuator nodes for fast and effective actuation process. We design a new protocol called ADP (Actuator Discovery protocol) which provides an optimal attachment for a sensor node to the nearest actuator. For the sensor-actuator coordination, the proposed ADP can guarantee ordering, synchronization and eliminates the redundancy of actions. It also renders the sensor and actuator nodes with the ability to self-organize and to exploit the spatial and temporal correlations in an efficient way. A new node-addressing scheme is proposed to make the routing more powerful especially in case of node failure and mobility.**

Keywords: SANETs (Sensor and Actuator Networks), Self-Organization, addressing, Spatial and Temporal correlation.

1 Introduction and Related Work

The advent in technology led to the emergence of SANETs giving a distributed control to the management, communication and coordination aspects of the network functioning formerly referred to as WSNs (Wireless Sensor Networks). A SANET consists of a group of sensors and actors (actuators), that are deployed to perform distributed sensing and actuation tasks, linked up by a wireless medium. The sensor nodes (small, cheap devices with limited computation) are deployed for the collection of data through a sensing mechanism, while actors (resource rich, better processing capabilities and stronger transmission power) take decisions and then perform appropriate actions upon the environment.

Current applications for WSNs are mostly research prototypes or are tailored toward a specific purpose. There is no typical WSN structure and architecture, and the basic goals of a WSN rely mostly on the considered application. Most of the sensor network research tends to focus on specific hardware with efficient, ingenious communication protocol algorithms and system control architectures that address the specific resource constraints, which did not lead to the emergence of a generic architecture for the sensor networks [1, 2, 3].

Despite of the sufficient work done in SANETs [4, 5, 6, 7], an effective coordination mechanism for efficient sensing, dissemination of information, and to perform right and timely actions (actuated by the actors) is required. These networks can be the integral part of systems such as: disaster/crime prevention, real-time military applications, environmental and health monitoring to smart spaces [8].

This paper investigates a new self-organizing framework for the SANETs. In this regard, we design a new ADP (Actuator Discovery protocol) which provides optimal attachment to any sensor node in the network with the nearest and most relevant actuator node. A sensor node anywhere in the network finds the nearest actuator using the proposed ADP during the initial deployment phase. Afterwords the data is relayed to the attached-actuator through a multi-hop discrete path unless the network undergoes a remarkable topology change, which covers the basics of sensor-actuator coordination. The ADP is very simple to accommodate the limitations of small sensor nodes and is energy efficient. The functionality of ADP is independent of the routing protocol and is distributed to cope with large scale deployments. For the actuator-actuator coordination, we employ the use of the assumed separate wireless interface to communicate with the neighboring actors so that they can perform long-range data communication without any involvement of sensor nodes to effectively coordinate and to perform the actuation process. The examples of actuation covers a wide range of possibilities and is typically application dependent.

For example, Targeting an enemy holding a snipper in a battle field can be an interesting case to consider. The actuation process has to localize the position of the enemy and actuate the destruction process. But the important constraint in this case is the latency because the sensor data can be no more valid at the time of actuation in case of increased latency. Which makes the assumption of a separate wireless interface for actuator-actuator coordination quite relevant.

Normally the two nodes for the communication are bounded to each other at the transport layer and as a result the efficiency of the routing protocols is highly degraded especially in the case of mobility. We have also proposed a new addressing scheme for sensor/actuator nodes for the efficient routing of data, exploiting correlations (both spatial and temporal) and to deal with node-failures in a proactive fashion. This helps us to manage data dissemination reliably and implicitly, independent of the node locations also giving an efficient distributed control for data relaying in the case of node failures, mobility, and to ex-

ploit correlations to a remarkable extent.

Note that in this work, we study the generic architecture which is robust and entirely distributed with a little varying-degree of centralized control for the actors on their local clusters. Instead of trying to achieve improvements only for the static configurations, we also addressed the node mobility, changes to the discrete paths, implications encountered during sleep and wakeup schedules and avoid using the GPS system for localizations.

The proposed solution consists of three phases; namely (i) The **Learning-phase** starts during the initial deployment, as depicted in Figure 1, when the sensor nodes try to locate the nearest actuator using a broadcast strategy through all the neighbors that are in their sensing radius.

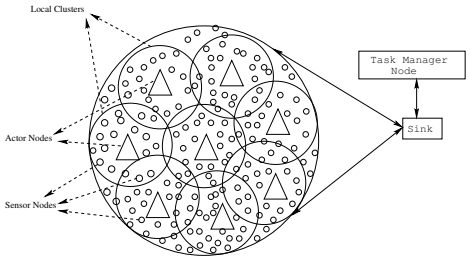

Figure 1: Architecture of SANETs

(ii) The **Coordination-phase** carries out the normal transmission of the data from any node to its attached actuator (sensor/actuator coordination) through a discrete path.

(iii) The **Failure and Recovery-phase** monitors the correlations properties of SANETs, which includes exploiting the local cluster for an alternate path to the attached actuator in a reactive approach.

The remainder of the paper is structured as follows. In Section 2, the self-organization framework and the dynamics of the proposal are explained in detail. Section 3 presents the experimental results. Section 4 terminates the paper with a brief conclusion and summarizes the future work.

2 The Self-Organization Framework

As we have already explained in the earlier section that there is no sensor/actuator network structure and architecture, and the basic goals of the previously done work relied mostly on the considered application. In this work, we propose an architecture which is tailored toward a standard behavior for most deployment scenarios, aiming to satiate the time-stringent requirements and effective energy utilization in a purely distributed fashion. The proposal consists of three phases: the learning-phase, the coordination-phase, and the failure and recovery-phase. In the following sub-section we detail the three phases.

2.1 The Learning-phase

The learning-phase starts when the sensors during the initial deployment stage locate the nearest actuator using a one-hop broadcast. The deployment of the sensor and actuator nodes can be randomly chosen or follows any predefined distributions. The finding of the "optimal-actuator attachment" for each sensor node is done through a novel protocol called ADP.

2.1.1 Actuator-discovery Protocol (ADP)

When a sensor node is turned on, it should first determine to which actuator node it has to be associated. For this end, a sensor node transmits a broadcast message named attach-request $AttachRequest\,(\,cost, M_j, C\,)$ to all its one-hop neighbors as shown in Figure 2. A neighboring node upon receiving an attach-request message checks that it has sent an attach-request in the period T_n (application specific), if it has already sent a broadcast to its neighbors, it has to wait for a reply until timeout. Otherwise it does the same unless the probe reaches the nearest actuator. The reply message named attach-reply $AttachReply_i(cost, M_j, A_i)$ from the actuator follows the probe and terminates at its origin defining a discrete path to the sensor node. If a node receives multiple paths then it chooses the shortest one (hop-count), and certainly if it receives more than one path containing the same hop-count, it chooses the earliest reply.

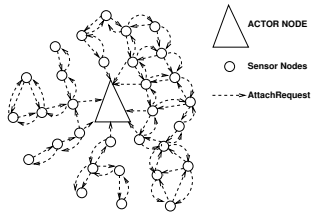

Figure 2: AttachRequest by sensors at the start of ADP

In Algorithm 1, we have induced a cost function (control procedure) to obtain a promised QoS in terms of delay and energy consumption. For this work, we have assigned the hop-count to this function to restrict the probe from reaching the other end of the deployment by limiting the probe packet not to cross a certain hop-count (referred as 'C'). This hop-count can be treated as a function of sensor-actuator node ratio in the network to limit unnecessary broadcast and also keeping the chances of actuator-discovery well alive (leaving the issue as implementation concern and not the design one, deployment specific). We don't take into account the distance between any node and its associated neighbors with the reason being that the energy required to transmit to a node in its sensing radius is a constant (no power control assumed for transmissions). ADP produces loop-free paths to the actuator nodes, as stated below.

LEMMA 1. *The next-hop selected by a sensor with ADP has a defined optimal path to the actuator node.*[1]

[1]We omit the proof because of page constraints.

Algorithm 1 ADP

```
Pseudo-code executed by all the sensor
nodes N_i during initial deployment-phase.
Initially:
 cost = ∞
 attached-actuator = ∞
 C = constant (the trade-off is explained
in Section 2.1.1).
 A_i = Identity of the Actuator.
 For any sensor node N_i
 do ActorDiscovery() {
   if cost (N_i, A_i) = ∞ then
     for each neighbor M_j of N_i do
       Send AttachRequest(cost, M_j, C)
       Receive AttachReply_i(cost, M_j, A_i)
#Determine optimal Actuator, and the
next-hop among the neighbors to reach
it.
        for each AttachReply do
          if path(cost, M_j) <
path(cost, M_{j-1}) then
            for N_i MinCost = path(cost, M_j)
            AttachedActuator = A_i
            next_ho_to_actuator = M_j
          end-if
        end-for
     end-for
   end-if
}
After deciding the actuator, each node
sends a "JoinRequest" to its actuator.
  send JoinRequest(A_i)
The actuator sends a "JoinAck" back
to the sensor node confirming cluster
joining.
  send JoinAck(M_j,N_i)
The procedure attach-request is
implemented recursively as follows.
  AttachRequest(cost, M_j, C){
    if (cost != ∞)
      return (UpdateCost(cost), M_j, A_i )
    else if (C != 0) then
      for all neighbors M_j of N_i
        do AttachRequest(cost, M_j, C − 1)
      end-for
    end-if
}
Actuator Reply to the broadcast messages
from the one-hop away nodes contains the
following.
  AttachRequest(cost, M_j, C)← ActorReply(cost = 1,
A_i)
UpdateCost() is the part of the control
semantics, and for this specific case,
it is chosen to be hop-count
UpdateCost(cost) {
  return cost + 1
}
```

As depicted by Figure 3, now a sensor node has a specified path to route its sensed data to the actuator nodes by simply forwarding it to only one of its neighbors (immediate next node in the path to the actuator), and the actuator also keeps the defined path to the node (building its tree structure for the localized cluster).

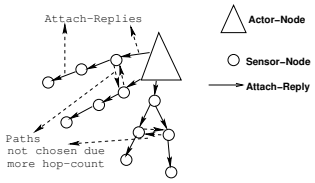

Figure 3: Actuator-replies (AttachReply) for corresponding AttachRequest messages

The background for this consideration takes us to the lack of a GPS system for position localizations in our study, which consumes a lot of sensor nodes' energy. In a similar fashion all the nodes reserve an optimal path to their nearest actors as shown in Figure 4, forming a local cluster, thus giving us the initial deployment in the form of well-distributed small clusters.

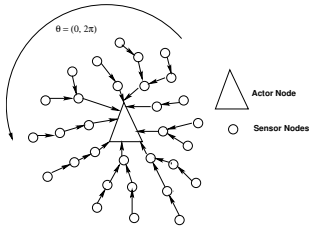

Figure 4: The Local Cluster formulated at the termination of ADP

2.1.2 Correlation Trees

Once all the nodes have defined paths to their attached actuator, the actuator rearranges all the paths in the following way to exploit correlation properties of the SANETs. As shown in Figure 5, the actuator rearranges all the paths between the direction Θ= [0,2π) in the depth-first arrangement order. In this way we have all the one-hop sensor nodes as the first children of the actuator node, so on and so forth. Which gives a depth-first search tree structure for the reason that the nodes are scanned in between Θ= [0,2π) and all the children of any one-hop node from the actuator needs to be managed first. All the sensor-nodes have defined identities (names, address, etc). But when a cluster is created and organized into the tree form by the actuator, it assigns temporary addresses to the sensor nodes and keeps the mapping with itself. As depicted in Algorithm 2 once the tree structure is maintained we define the temporary addresses of nodes by addressing all the nodes on the same hop-count first, following their descendants aiming toward a breadth-first addressing scheme. The mapping between the actual node-address and temporary-address is managed by the actuator ($N_{add}^{(i)} \rightarrow T_{add}^{(i)}$) in every cluster. This strategy helps in optimizing the search to the attached neighbors in case of node-mobility and failure, and exploiting the correlation properties (explained in Section 2.4).

Algorithm 2 SOT

```
Pseudo-code executed by the actuator-node
scan the local-cluster Θ= [0,2π)for all
nodes
    do SOT(node n)
      visit(n)
      for each child (next-hop) w of n
        do SOT(w) (Initially, first
scanned node one-hop away from actuator)
        add edge nw to the Tree SOT
      end for
A temporary-address is assigned to each
node by the actuator.
Unmark all the vertices
    choose some starting vertex n
(actuator-node)
      mark(n)
      list L = n
      Tree SOT = n
        while L non-empty
choose some vertex v from the front of
list
        visit(v)
        AssignAdd(v)
        for each unmarked neighbor w
          mark(w)
          AssignAdd(W)
          add it to the end of list
          add edge vw to SOT.
        end-for
```

LEMMA 2. *All the sensor-nodes are attached to the actuator with increasing hop-count in a depth-first order (Algorithm 2).*

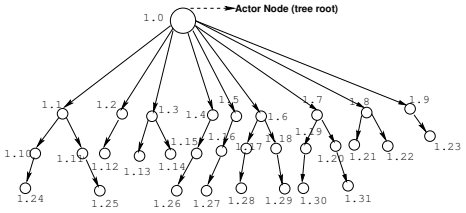

Figure 5: A Self-Organized Tree (SOT)

2.2 The Coordination-phase

The deployed sensor nodes start sensing the distributed environment, and transmit their data through the defined path to the attached actuator. For the sensor-actuator coordination the actuator-attachment and the paths obtained to route data to the actors provide effective energy optimization for the sensor nodes. For the actuator-actuator coordination we have utilized another wireless transmission interface to keep information about the neighboring actors and operating at a higher degree than sensor/actuator coordination. The actuator-actuator coordination has to be fast enough for the timely-actuation process. In this fashion, the selection of single or multiple actors for the actuation, in-sequence delivery of different events detected in a region along with a varying degree of challenges [4] are handled quite effectively. The actors route their data to the sinks using AODV (Ad Hoc On-Demand Distance Vector routing) to confront with the issues related to the validity of sensor data at the time of actuation. The motivation behind choosing AODV as a routing protocol for the actuator/actuator coordination is that the network of actors could be seen as a Mobile Ad-Hoc Network (MANET) which in general has a low mobility. This whole strategy helps to actuate the required actions in an energy efficient and timely-coordinated manner with a guaranteed QoS (application specific approach). There can only be two deployment configurations for the SANETs:

Static Deployment: In this case, both the sensor and actuator nodes are static. The gain with the static distribution is the maximum due to efficient dissemination of routing data to the acquired actors also depending on the environment friendliness of the deployment and no accidental node failures. The gain in the efficiency can be intuitively seen by the trade-off between the time required in network-learning and then adapting it. Monitoring the penetration of the discovery-probe gives a good balance for energy-efficiency and a predefined guarantee for the actuator-discovery.

Mobile Deployment: For mobile deployment, we have four different types of configurations (detailed in Section 2.3). The learning phase for mobile-case is exactly the same as for the static-deployment. But at any point in time, the discrete path to the actuator nodes may change due to the mobility of the nodes. The purpose behind organizing the cluster in the above-explained behavior is to exploit the correlation properties (explained Section 2.4) of the SANETs not only the data-centric level but also at the node-centric level (direct-addressing).

2.3 Failure and Recovery-phase

We assumed that every sensor node has a pre-defined maximum battery life-time with a minimum threshold indicating failure in near future. The Failure and Recovery-phase monitors this time line and inform the actuator before the actual failure to take a few precautionary measures which can include: (i) Exploiting the local cluster for an alternate path to nodes that lost their routes to the actuator. (ii) Do nothing if there was no further attached node. (iii) And update the cluster information of the attached-actuator for local management. But if the failure is due to any sort of accident then the correlations properties are utilized to their full for the recovery procedures in the following way as shown in Figure 6.

We have taken into consideration four possible deployment scenarios.

(i) Both sensors/actuators are static: For the static-case, if there is a node failure in the middle of the path, the farther nodes can do a one-hop broadcast to all the neighbors in their sensing range, which can reply with their respective hop-counts toward the actuator. The sensor node selects the shortest path to the actuator and an update message is sent (piggybacked along with the data-packet) to the actuator for updating the correlation tree.

(ii) Static actuators and mobile sensors: In mobile-

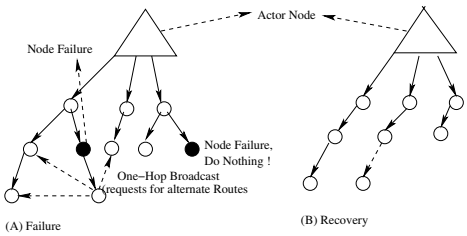

Figure 6: The occurrence of Failure and steps required for Recovery Procedure.

sensor configuration, the path to the attached-actuator can be no more valid at any instant in time due to the mobility pattern of the nodes, and can also be the result of an accidental-failure. The nodes that have no more defined paths sends a one-hop broadcast. If a reply is received, the same procedure will follow as for the static case. Otherwise the one-hop neighbors send a path-search probe to their one hop neighbors, and the procedures can terminate after acquiring a path to the actuator. In-fact, if this is the case, the neighboring nodes that don't have a defined path has already sent a path-update request after time-out in most practical scenarios. All the nodes have to authenticate the validity of their paths after a certain defined interval T_o which can be a trade-off between the mobility model and the path-update procedure. In the mobile-sensor case the constant dissemination of energy to keep the paths updated can be a direct mapping to the applications' requirements. The dry analysis for the proposed scheme intuitively satisfies the efficient consumption of energy and meeting the required constraints for real-time traffic.

(iii) Static sensors and mobile actuators: Now we consider the case where actuator nodes are mobile in the deployment. During the learning phase the sensor nodes attain their initial attachment to an actuator node. But the defined path again can be no more valid at any point in time during the lifetime due to actuator-mobility. In this case, it is the actuator nodes that are experiencing mobility and thus it is the duty of the actuator nodes to periodically update the actuator-attachment procedure for the sensor nodes based on the mobility pattern. Upon receiving the update-request from the actuator the one-hop away sensor nodes forward it to their neighbors. Again the penetration of the probe is limited in terms of number of hops. The actuator nodes periodically update the neighboring actors' information to have a minimal bound on the latency.

(iv) Both sensors/actuators are mobile: Typically this is one deployment which is most energy consuming due to obvious reasons. After the initial learning phase, and in the event of node movements both the actors and sensors keep periodically updating their cluster and actuator-attachment informations respectively. As in the previous case, the actors send a broadcast to the one-hop sensor nodes, and these sensor nodes in return do the same unless the probe reaches its predefined premises. Similarly a sensor node during its life time, if failed to receive a reply before timeout from its one-hop neighbor, sends an update broadcast request to all of its one-hop neighbors and works further with already defined mechanism. In all the cases, recovery-procedure is dependent on the mobility of the nodes and also on the speed of mobility. But we can always limit the energy utilization of the sensor nodes by restricting the update probe within defined boundaries with the maximum possibility of finding an optimum actuator-attachment. Also note that the sensor nodes are kept simple enough in hardware terms because we have utilized only message passing between nodes quite simply and efficiently. And the sensor nodes are nowhere involved in complex computations to attain shortest paths to sinks (actuators' job in our study) and to obtain correlation features.

LEMMA 3. *The alternate-path obtained is also a loop-free shortest path to the actuator-node (Section 2.3).*

2.4 Node Correlations and Sleep-Awake management

The nodes can be addressed explicitly by the actors for any depth of deployment sensing and information transfer using their addresses attributed during the learning phase. The two-nodes for transmission are addressed by their node identities/addresses at the transport layer, but the actual communications are based on their temporary addresses so as to monitor the transfer in the same fashion by exploiting the correlation from the neighboring nodes. For the static-deployment, if we look at the local-tree created by the actuator-node, any node transmitting some sensed data can be approximately acquired by its one-hop neighbors on both sides at the same hop-count from the actuator (node-centric spatial correlation), which can be validated by the applying the data-centric spatial correlation techniques available in the state-of-the-art for the sensor/actuator networks at the actuator node. This hypothesis is validated by analyzing the work done in [13, 14, 15] to our proposed approach and we can also acquire a high degree of effective node-energy utilization. If we look again at the Figure 6-a, in case of a node failure, the two neighbor nodes (if the failed-node has any in its unit disk of sensing area) at the same hop count from the actuator can be utilized to acquire approximately the same sensed-information (spatial correlation). Which provides us with an extra-degree of sleep management procedures, by obtaining the approximately same information in case of insufficient-energy nodes. In this fashion as shown in Figure 7, we can carry out seamless transport of data from sensor nodes toward the actors and vice-versa.

For mobile-case, the additional feature available is the explicit monitoring of the sensor-nodes. If the sensor nodes moves according to any defined mobility model, it is only the temporary address of the node that changes and there can be a continuous relaying of data toward interested actuator nodes (intra-actuator mobility). During the course of mobility if a particular sensor nodes enters the sensing range of any of the neighbor actors, they perform the path-update request in the same-old way. Once actuator identified, the path for the data routing is kept by the sensor node and the local-cluster of the actuator is mod-

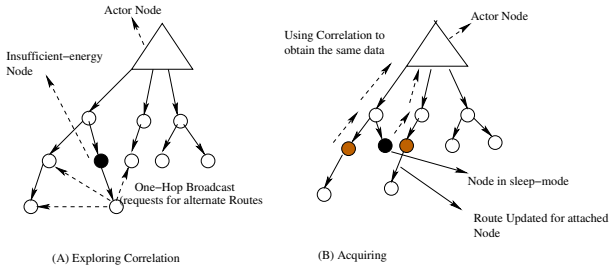

Figure 7: Exploring and Acquiring Correlation Procedure (EACP)

ified, which in turn broadcasts this information to other clusters in its sensing-range (inter-actuator mobility). In this very fashion we can explicitly monitor any desired level of deployment monitoring, define mobility patterns (application dependent), reduce the energy dissipation for data-routing, and achieve a defined QoS for real-time applications at the cost of minimal management overhead. This management task is performed by actuator nodes, so potentially, we can't even consider it a minimal overhead as these are not the constraints for actuator nodes.

LEMMA 4.*The two-neighboring nodes at the same hop-count from the actuator node forms the best spatial-nodes pair (Section 2.4).*

The nodes that are not involved in any sensing process and are certainly not in the data relaying path can be sent to sleep by the actuator nodes to conserve energy in the following fashion. At the actuator node (static-mode), we have defined paths to reach every sensor node in the network. When the data arrives at the actuator node from a particular node (through a defined path), it can mark the path in its local-tree knowing the approximate coordinates of the sensor node (s) considering the probability of its future use (we are considering a general monitoring of sensor data, that can be required by multiple actors to initiate the actuation process), exploiting the temporal correlation again at the node-centric level. In this fashion the actuator can send a sleep schedule to all the other nodes (can keep the spatially correlated nodes alive for high performance deployments) on different routes from the connected-actuator node. For the static-configuration it can acquire a trade-off between any level of desired energy-efficiency and a guaranteed QoS for real-time applications. For mobile configuration, we follow the sleep procedures in the same fashion, but the wake up is initiated by the path validation followed by a path-update procedure, if required.

3 Experimental analysis of ADP

we finally implemented the ADP in ns-2 as an application layer protocol. We have considered a wide range of network topologies and test-validated the performance of ADP with all considered topologies. Due to the posed space problem, we presented our results for a complex random network. We have considered complex random network configuration with 95 sensors and 5 actuators, where both sensor and actuator nodes are static. We have also evaluated the average number of sensors per actuator (local cluster) for complex random network topology in Figure 8. The local clusters have an equilibrium in terms of number of sensors, and the calculated mean turns out to be 19 sensors per actuator. which proves that the paths chosen by the sensors to their respective actuators are the shortest paths. The CDF for the delay distribution shows that over 65% of the nodes lie in the average-delay (according to our findings) range as shown in Figure 9 Along with the mean number of nodes per cluster we have also shown, in Figure 10, the mean path length as a function of network size. The mean path length, which is related to the end-to-end latency for both cases: Fixed number of actuators, and secondly actuators increasing by 5% with network size. However, the increase is more gradual with the increasing number of actuator nodes, which would be a more practical assumption for the SANETs.

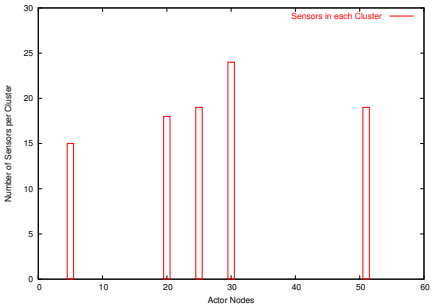

Figure 8: Number of Sensors per Actuator in a Complex Random Network Topology

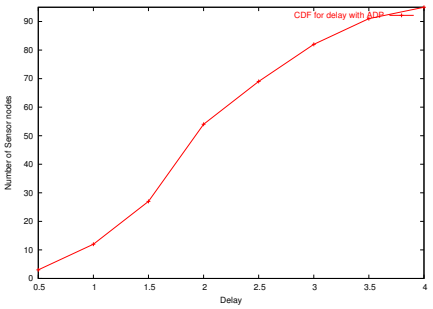

Figure 9: CDF for Attachment-Delay in Complex-Random Network Configuration

4 Conclusions and Future Work

For sensor-actuator coordination, the proposed ADP can guarantee ordering, synchronization and eliminates the redundancy of actions. For actuator-actuator coordination, the proposed unified framework using AODV can be exploited by different applications to always select the best networking paradigm that is possibly available according to the sensed information and to perform the necessary operations in an efficient way. The issues

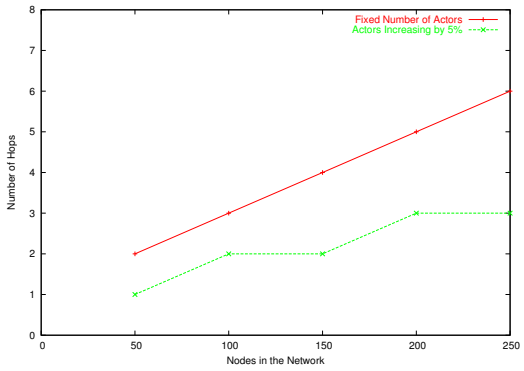

Figure 10: The Mean Path Length for ADP with Increasing Network Size

related to node-failures and mobility are well handled by the proposal in all possible deployment scenarios. Further the new addressing-scheme improves exploiting correlation behaviors especially in the case of mobility and node-failure. Mobility of the sensor nodes coordinated by the actuator further improves the results obtained in [19]. Localization and node positioning can be performed by improved MDS [10, 11] using actuator nodes as coordinators for effective energy utilization and fast localization of events for relevant actuation procedures. A major contribution to the state-of-the-art [9 - 19] in this paradigm can be directly applied to our proposal with minor modifications for efficient energy-utilization of the sensor nodes and to cope with the real-time constraints posed by many applications.

We are working on the performance study of the proposed framework using NS-2 and TOSSIM simulators. We have not presented the detailed analytical analysis for ADP due to the posed space problems. The issues concerning the choice of routing-protocol, correlation, localization, position optimization and object tracking will be presented in a later work. We are also working on MAC optimization based on the proposal for the efficient utilization of node-energy and to satiate the real-time problems experienced so far in the SANETs.

References

[1] V. Paruchuri, A. Durresi, and L. Barolli. “Energy Aware Routing Protocol for Heterogeneous WSNs”, in Proc. of DEXA, August 2005.

[2] Q. Fang, J. Gao, and L. J.Guibas. “Locating and Bypassing Routing Holes in Sensor Networks” in proc. of IEEE Infocom March 2004.

[3] J-H. Chang, and L. Tassiulas. “Maximum Lifetime Routing in WSNs”, IEEE/ACM Transactions on Networking, Vol. 12, No. 4, August 2004.

[4] Akyildiz, I.F.; Kasimoglu, I.H. Wireless “sensor and actor networks: research challenges”. Article Ad Hoc Networks, Vol. 2, No. 4, pp 351-367, October 2004.

[5] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci. “WSNs: a survey**”.** Article Computer Networks, Vol. 38, No. 4, pp 393-422, March 2002.

[6] T. Melodia, D. Popili, V. C. Gungor and I. F. Akyildiz. “A distributed coordination framework for WSANs: in Proc. of MobiHoc, May 2005.

[7] W. Hu, N. Bulusu, and S. Jha. “A Communication Paradigm for Hybrid SANs”, International Journal of Wireless Information Networks, Vol. 12, No. 1, pp 47-59, January 2005.

[8] E. Yoneki, J. Bacon: A survey of WSN technologies: research trends and middleware’s role, Technical Report, No. 646, September 2005.

[9] V. P. Mhatre, C. Rosenberg, D. Kofman, R. Mazumdar, N. Shroff. “A Minimum Cost Heterogeneous Sensor Network with a Lifetime Constraint”, IEEE Transactions on Mobile Computing, Vol. 4, No. 1, pp. 4-15, January 2005.

[10] Xiang Ji, and Hongyuan Zha. “Sensor Positioning in Ad-hoc WSNs Using Multidimensional Scaling”, in Proc. of IEEE Infocom, March 2004.

[11] Yi Shang, and Wheeler Ruml. “Improved MDS-Based Localization”, in Proc. of IEEE Infocom March 2004.

[12] G. Sarma, and R. Mazumdar. “Hybrid SNs: A Small World”, in proc. of MobiHoc, May 2005.

[13] A. Cerpa, J. L. Wong, M. Potkonjak, and D. Estrin. “Temporal Properties of Low Power Wireless Links”, in Proc. of MobiHoc, May 2005.

[14] I. F. Akyildiz, M. C. Vuran and O. B. Akan.” On Exploiting Spatial and Temporal Correlation in WSNs”, in Proc. of IWWAN, May 2004.

[15] H. Gupta, V. Navda, S. R. Das, and V. Chowdhary. “Efficient Gathering of Correlated Data in Sensor Networks” in Proc. of MobiHoc, May 2005.

[16] G. Xing, C. Lu, Y. Zhang, Q. Huang, and R. Pless. “Minimum Power Configuration in WSNs”, in Proc. of MobiHoc, May 2005.

[17] F. Kuhn, T. Moscibroda, and R. Wattenhofer. “Initializing Newly Deployed Ad Hoc and Sensor Networks”, in Proc. of MobiHoc, May 2004.

[18] J. L. Bredin, E. D. Demaine, M. T. Hajiaghayi, and D. Rus. “Deploying Sensor Networks with Guaranteed Capacity and Fault Tolerance”, in Proc. of MobiHoc, May 2005.

[19] B. Liu, P. Brass, O. Dousse, P. Nain, and D. Towsley. “Mobility Improves Coverage of Sensor Networks”, in Proc. of MobiHoc, May 2005.