Real-Time Coordination and Routing in Wireless Sensor and Actor Networks

Ghalib A. Shah, Muslim Bozyiğit[1], Özgür B. Akan, and Buyurman Baykal[2]

[1] Department of Computer Engineering,
Middle East Technical University, Ankara, Turkey 06531
{e135333, bozyigit}@metu.edu.tr
[2] Department of Electrical and Electronics Engineering,
Middle East Technical University, Ankara, Turkey 06531
{akan, baykal}@eee.metu.edu.tr

Abstract. In Wireless Sensor Actor Networks (WSAN), sensor nodes perform the sensing task and actor nodes take action based on the sensed phenomena in the field. To ensure efficient and accurate operations of WSAN, new communication protocols are imperative to provide sensor-actor coordination in order to achieve energy-efficient and reliable communication. Moreover, the protocols must honor the application-specific real-time delay bounds for the effectiveness of the actors in WSAN.

In this paper, we propose a new real-time coordination and routing (*RCR*) framework for WSAN. It addresses the issues of coordination among sensors and actors and honors the delay bound for routing in distributed manner. *RCR* configures sensors to form hierarchical clusters and provides delay-constrained energy aware routing (DEAR) mechanism. It uses only cluster-heads to coordinate with sink/actors in order to save the precious energy resources. The DEAR algorithm integrates the forwardtracking and backtracking routing approaches to establish paths from source nodes to sink/actors. In the presence of the sink in WSAN, it implements the centralized version of DEAR (C-DEAR) to coordinate with the actors through the sink. In the absence of sink or ignoring its presence, there is a distributed DEAR (D-DEAR) to provide coordination among sensors and actors. Cluster-heads then select the path among multiple alternative paths to deliver the packets to the actors within the given delay bound in an efficient way. Simulation experiments prove that *RCR* achieves the goal to honor the realistic application-specific delay bound.

1 Introduction

Recent advances in the field of sensor networks have led to the realization of distributed wireless sensor networks (WSN). A WSN is composed of large number of sensor nodes, which are densely deployed in the sensor field in a random fashion with a sink node. The task of sensor nodes is to detect the events in the sensors field and route them to the sink node, which is responsible for the monitoring of the field.

Y. Koucheryavy, J. Harju, and V.B. Iversen (Eds.): NEW2AN 2006, LNCS 4003, pp. 365–383, 2006.

Recently, the capabilities of the WSN are extended to include the actor nodes responsible with taking action against the detected events [1]. Such architecture is called a wireless sensor and actor networks (WSAN), where a small numbers of actors, as compared to sensors, are spread in the sensor field as well. Actors are mostly mobile and resource-rich devices and can be thought to form a mobile ad hoc network of their own. This paradigm of WSAN is capable of observing the physical world, processing the data, making decisions based on the sensed observation and performing appropriate actions. Typically, the architecture of a WSAN consists of sensors which sense the phenomena, a sink that collects the data from the sensors to process and actors that act upon the command sent by the sink. In the literature, such architecture is known as *semi-automated architecture*. An architecture in which sensor nodes send information to the actor nodes directly without the involvement of sink node is called an *automated architecture* [1]. It is apparent that the communication path in a *semi-automated architecture* introduces significant delay, which is not acceptable for delay-sensitive applications. For example, consider a military application where sensors in the battlefield will detect the movement of red forces and send the information to the sink which is situated in a remote command and control station. The sink then triggers an action through an actor to counter in the threat area. In this case unnecessary delay is introduced due to sensor-sink communication which could be removed if the actors can take localized actions without the involvement of the sink; depending on the data sent by the sensors. Hence, the most challenging task in WSAN is the coordination between sensors and actors to provide real-time response.

In WSAN, the effective sensor-actor coordination requires the sensors to know the right actors and the routes to reach them. Moreover, it requires delay estimates for all possible routes. In addition to energy constraints as in WSN, WSAN also imposes timing constraints in the form of end-to-end deadlines. Clearly, there is a need for real-time communication protocols for WSAN, which provide effective sensor-actor coordination while consuming less energy.

There have been considerable efforts to solve the routing problem in wireless sensor networks [3], [5], [6], [10], [11], [12], [13], [14]. However, these protocols do not consider the heterogeneity of WSAN. Moreover, none of these protocols provide sensor-actor coordination and real-time routing. A coordination framework [2] for WSAN has been proposed that is an event-based reactive model of clustering. Cluster formation is triggered by an event so that clusters are created on-the-fly to optimally react to the event itself and provide the required reliability with minimum energy expenditure. Reactive cluster formation algorithms have the disadvantage that they consume precious time on event occurrence for cluster formation. Hence for real-time coordination such an architecture is not suitable. Moreover, cluster to actor routing in [2] is done using greedy geographical approach. A packet forwarding node finds the next hop node according to the greedy approach failing to do so results into a packet loss as the packets enters into a void region. Since the work assumes that the network is dense therefore it does not propose any void region prevention or recovery mode implementation.

Consequently, there exists no unified solution which addresses the real-time coordination as well as routing problem for the heterogeneous WSANs.

In this study, we propose a real-time coordination and routing (RCR) framework, which addresses the sensor-actor coordination with real-time packet delivery in the *semi-automated architecture* as well as *automated architecture*. RCR, incorporates the two components, namely DAWC and DEAR. DAWC is our heuristic clustering protocol used to dynamically configure the sensor nodes in the form of clusters to achieve energy efficiency. Whereas, RCR achieves the real-time demand τ of packet delivery through our delay-constrained energy aware routing (DEAR) protocol that is the first and foremost aim of the protocol. The DEAR protocol, described in Section 3, establishes a backbone network by integrating the forward tracking and backtracking mechanism that provides all the possible routes towards target nodes (sink/actors). The path selection criterion is based on the packet delay as well as the balance consumption of energy of sensor nodes. In the presence of the sink in WSAN, it implements the centralized version of DEAR (C-DEAR) to coordinate with the actors through the sink. On the other hand, when there is no sink or central node in WSAN, it provides the distributed version of DEAR (D-DEAR) for coordination among sensors and actors. Performance evaluation study reveals that RCR addresses the real-time coordination and routing requirements of WSANs.

The remainder of the paper is organized as follows. In Section 2, we present the cluster formation procedure. We discuss the route path computation and sensor-actor coordination in Section 3. Performance evaluation and results are considered in Section 4. Finally, the paper is concluded in Section 5.

2 Hierarchical Configuration of Sensor Nodes

To provide a real-time coordination among sensors and actors in WSAN, RCR configures the sensor nodes hierarchically. The operations of the routing protocols are discussed in Section 3. The configuration of sensor nodes to achieve real-time coordination is discussed in this section. We propose so called Dynamic Weighted Clustering Algorithm (DAWC). The operations of DAWC consist of cluster formation of sensor nodes, delay budget estimation for forwarding a packet from the cluster-heads and to guarantee the packet delivery within the given delay bound τ.

There are many clustering algorithms [4], [7], [8], [9] proposed in the literature but unlike these studies, DAWC is neither periodical clustering procedure nor the cluster size is fixed in terms of hops. It adapts according to the dynamic topology of the sensor and actor networks. Our work is motivated from the previous work "A Weight Based Distributed Clustering" [8]. However, unlike [8], DAWC provides the cluster formation procedure to cope with the dynamic number of hops in a cluster and provides support for real-time routing. Cluster formation is based on the weighting equation formulated in the Section 2.2, which sets weight to different application parameters according to the applications need. DAWC adapts to the variation in the sensors field and can be optimized accordingly.

Once the cluster has been formed, cluster-heads get estimates of *delay budget*[1] of their member nodes. The delay budgets of member nodes help to build the delay-constrained energy efficient path.

When the sensor nodes are not uniformly deployed in the sensor field, the density of nodes could be different in different zones of the field. Choosing an optimal number of clusters k_{opt}, which yield high throughput but incur latency as low as possible, is an important design goal of DAWC.

2.1 Optimal Clustering

In this section, we evaluate the optimal number of clusters k_{opt} and, hence, the optimal size of clusters. We assume the uniform deployment of sensor nodes and devise a formula to find out k_{opt}. Later, we see the implication of it to non-uniform deployment for optimal configuration in Section 2.1.

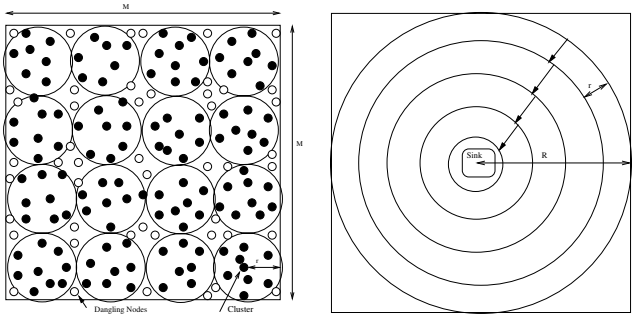


Fig. 1. Network model to formulate the optimal clusters. Fig 1(a) represents the model to find the probability of DN nodes. Fig 1(b) illustrates the model of routing packets from cluster-heads to the sink node.

Optimal Clusters in the Field. Each member node transmits its data packet to the cluster-head. Let r be the transmission radius of each node regardless of its functioning. In the clustering process, there is some probability that a number of dangling nodes[2] (DN) may exist due to the density of nodes or coverage of the elected cluster-head. Let us first find out the probability of such nodes. To do that, we map the sensor field ($M \times M$) to non-overlapping circles of radius r as shown in Fig 1(a) and assume that the nodes lying outside the boundary of the circle are DN nodes and the others are member nodes. These DN nodes require affiliating to cluster-head through the nodes insides the circle (member nodes). The squared field M^2 can be packed by $M^2/(2r)^2$ non-overlapping circles of radius r. Thus, the probability γ of a multi-hop member is

[1] *delay budget* is the time to deliver the data packet from the cluster-head to the member node.

[2] Nodes which have not joined any group or cluster are referred as *dangling nodes*.

$$\gamma = \frac{M^2}{(2r)^2} \times \frac{(2r)^2 - \pi r^2}{M^2} \approx 0.214$$

Let E_{elec} be the energy consumed by the electronic circuitry in coding, modulation, filtration and spreading of the signal. Whereas, $\epsilon_{amp} r^2$ is the energy consumed for signal amplification over a short distance r. Thus, the energy consumed by each member node in transmitting a packet of size l is

$$E_{Member} = l(E_{elec} + \epsilon_{amp} r^2(1 + \gamma))$$

The above equation can be simplified by taking the area as circle given in Eq. 16 of [15].

$$E_{Member} = l(E_{elec} + \epsilon_{amp} \frac{M^2(1 + \gamma)}{2\pi k})$$

Let us assume that the sensory field is covered by a circle of radius R, where the sink node lies at the center of this circle as shown in Fig 1(b). This assumption is made for routing packets from cluster-heads to the sink. The assumption is reasonable because it is less likely that a node lying outside the boundary of circle will be elected as cluster-head due to the low weight than the nodes inside the circle. Cluster-heads do not extend their transmission range to transmit packets directly to the sink node and, therefore, has the same radius r as member nodes. We adapt the multi-hop model proposed by [17] to route packets from cluster-head to the sink.

In the model, a circle is divided into concentric rings with the thickness r. The energy spent to relay the packet from outside ring towards inside ring is $l(2E_{elec} + \epsilon_{amp} r^2)$. The number of hops Γ require to route packet from cluster-head to sink node can be calculated by $\frac{R}{r}(1 - \hbar)$. Where $\hbar$ is the probability that the cluster-head is close enough to the sink to directly transmit packets. This probability can be calculated by using the nodes distribution in the rings given in [17].

$$\hbar = \frac{r}{R} \sum_{i=1}^{R/r} \frac{R^2 - (ir)^2}{M^2}$$

Since packets from the cluster-heads far from the sink node are relayed through intermediate nodes. Therefore, if $\Lambda(i)$ is the number of neighbors of a node i then $\Lambda(i) \times E_{elec}$ is the energy consumed by the electronic circuitry of the neighbors of forwarding node i during the propagation of packet. The number of neighbors (Λ) of any node can be found as

$$\Lambda = n \frac{\pi r^2}{M^2}$$

Hence, the energy consumed in routing data from cluster-head to sink is measured as

$$\begin{aligned} E_{CH-Sink} = l(&\Lambda E_{elec} + E_{elec} \\ &+ (2E_{elec} + \epsilon_{amp} r^2 + \Lambda E_{elec})\Gamma). \end{aligned}$$

The total energy dissipated by the network is

$$\begin{aligned} E_{total} = l((n + n\Lambda)E_{elec} \\ + k(2E_{elec} + \epsilon_{amp}r^2 + \Lambda E_{elec})\Gamma \\ + n\epsilon_{amp}\frac{M^2(1+\gamma)}{2\pi k}) \end{aligned}$$

For $r < R$, the optimum value of k can be found by taking the derivative of above equation with respect to k and equating to zero

$$k_{opt} \approx \sqrt{\frac{n(1+\gamma)}{(2\pi(1 + \frac{2E_{elec}}{\epsilon_{amp}r^2} + \frac{\Lambda E_{elec}}{\epsilon_{amp}r^2}))\Gamma}} \times \frac{M}{r} \tag{1}$$

It is noteworthy that the optimal value depends on the transmission range r of the nodes. For long range of transmission, the value of optimal clusters k_{opt} is small. This is in contrast to the optimal clustering in SEP [16] that is independent of range parameter. For example, Let us assume that $E_{elec} = 50nj/bit$ and $\epsilon_{amp} = 10pj/bit/m^2$ for experiments and $n = 100$, $M = 100$ with the sink at center of the field $(x = 50, y = 50)$. Then the value of radius R is obtained by drawing a circle at $x = 50, y = 50$ to cover the field. The estimated value is $R = 60$ and let set the range r of individual nodes to 25. In this scenario, we obtain the value of $k_{opt} \approx 10$. By increasing the range of nodes to 40 meters, we obtain $k_{opt} \approx 7$. Whereas, the value of k_{opt} in SEP [16] is 10 regardless of the transmission coverage of individual nodes.

Optimal Cluster Size. Besides choosing the optimal value k_{opt} for number of clusters, the number of member nodes in a cluster is as important as the number of clusters. The optimal value of member nodes M_{opt} helps in load balancing of clusters and ensures efficient MAC functioning. Head nodes use more energy than the member nodes. Since the sensors are energy-constrained devices and a cluster-head is selected from the homogeneous nodes, the number of member nodes in a cluster should ensure the longevity of the cluster-head as long as possible.

When the deployment is uniform then the M_{opt} can be easily found by n/k_{opt}. However, for non-uniform deployment, the number of member nodes depends on the density in a particular zone of the sensor field. Therefore, we put the maximum and minimum limits M_{Min} and M_{Max} respectively on the size of cluster such that we still achieve k_{opt} clusters in non-uniform deployment. Suppose M_i is the number of neighboring nodes of any ith node and $Max(M_i)$ is the maximum number of neighboring nodes that any of the ith neighbor node have. We measure density of nodes in a particular zone by comparing the neighbor nodes M_i with M_{opt}. We can conclude that the deployment is:

$$\begin{aligned} M_i/M_{opt} > 1, dense \\ M_i/M_{opt} \approx 1, uniform \\ M_i/M_{opt} < 1, sparse \end{aligned}$$

We set the limits M_{Min} and M_{Max} as:

$$M_{Max} = Max(M_{opt}, Max(M_i))$$

That is, the maximum of M_{opt} and maximum number of neighbors of any cluster-head at the time of cluster formation.

$$M_{Min} = M_{opt} \times Min(M_{opt}, Max(M_i))/M_{Max}$$

These limits allow the configuration to manage the dense as well as sparse deployment of nodes.

2.2 Cluster Formation

The first phase of DAWC is to form k_{opt} number of clusters. During the formation of clusters, each cluster-head gets the *delay budget* of each of its member node. The delay budget is used to identify an appropriate node to send delay-constrained data packet. The cluster election procedure is based on calculating weight for each sensor node in the sensor field and it chooses the head that has the maximum weight. The weighting equation is given in cluster-head election procedure. We define weight threshold of the cluster-head to rotate the cluster-heads responsibility among all the potential nodes. A cluster is not strictly organized to 1-hop but it accepts the membership of a node that could not reach any cluster in the first phase of cluster formation. Therefore, a cluster can include n-hop members, for n $\geq$ 1. Although the operations of the protocol starts after the first phase of cluster formation, there may still exist some DN nodes.

We assume that the nodes are aware of their geographical locations through some localization devices like GPS. In the next section, we describe the details of computing the *delay budget* and cluster formation procedure is presented in Section 2.2.

Delay Measurement. When nodes are initially deployed in the field, every node i broadcasts its ID, which is added in the neighbors list by all the nodes that receive this broadcast. A node that receives this broadcast, computes the delay $delay_s$ of the packet received from its neighbors along with the delay budget $delay_r$. $delay_s$ is the delay of the packet experienced and $delay_r$ is the delay that the sender estimated when some packet was received from the receiver i.e $delay_r : sender \leftarrow receiver, delay_s : sender \rightarrow receiver$.

The total delay in transmitting a packet from one node to a node in its neighbor is measured by the following factors: queue, MAC, propagation and receiving delay represented by T_q, T_{Mac}, T_{Prop} and T_{Rec} respectively. The wireless channel is asymmetric that does not imply any synchronization mechanism. Therefore, the delay is measured partially at both the sender and receiver. Sender measures the delay L_s until the start of transmission that includes the queue delay as well as the MAC contention delay. Whereas, the receiver adds the factor L_r as sum of propagation delay and receiving delay to get the total packet delay L_h.

$$L_s = T_q + T_{Mac},\ L_r = T_{Prop} + T_{Rec}$$

The hop latency L_h can be computed as sum of these factors:

$$L_h = L_s + L_r$$

Initially, the delay is measured by exchanging the *hello* beacons. Each node maintains this value in its neighborhood table that contains the fields $\{ID$, $delay_r$, $delay_s$, $energy$, $weight\}$. To get more close to the accurate measurement of packet delay, the delay value is updated when the events flow from member nodes to cluster-head. It is due to the fact that the size of data packet may differ than the *hello* beacon that may experience different delay.

For the d-hop member of a cluster, packets are forwarded by the intermediate nodes to the cluster-head. Each intermediate nodes calculates the delay L_h of the packet and forwards the packet to next hop by adding its L_h in L_s. After following through some intermediate nodes, cluster-head gets the packet and adds its factor L_r as the receiver of the packet. Hence the cumulative delay $delay_s$ of a member node d hops away from its cluster-head is computed as:

$$delay_s = \sum_{i=1}^{d} L_{h_i} \tag{2}$$

Member nodes compute the delay estimate $delay_s$ of their cluster-head in this way through cluster-head *announcement* beacon. When member nodes broadcasts *hello* beacons, they put the $delay_s$ of cluster-head as $delay_r$ into the beacon. Cluster-head gets the delay budget $delay_r$ for its members and use this value in routing.

Cluster-Head Election Procedure. DAWC effectively combines the required system parameters with certain weighting factors to elect cluster-heads. Values of these factors can be chosen according to the application needs. For example power control is very important in CDMA-based networks. Thus, weight of the power factor can be made larger. In order to achieve the goal of energy saving, RCR minimizes the frequency of clusters reformations. It is achieved by encouraging the current cluster-heads to remain cluster-heads as long as possible. That is why we have included the time of being cluster-head in computing weight. Similarly, if resource-rich devices are deployed to work as cluster-heads then the weighting factors of distance can be made large and time of being head can be kept small. The operation of the cluster-head election procedure is outlined as follows:

Each node i maintains a list of its neighbors. Each entry of the neighbor list contains node ID and its weight W_i computed on the basis of the selected parameters. Once the neighbor list is ready, the cluster-head election procedure is initiated for the first time. Each node i does the following:

- Calculates D_i as the average distance to its neighbors, M_i as the total number of its neighbors, E_i as its energy and T_i as the time being head in the past.
- Computes weight $W_i = (c_1 D_i + c_2 E_i + c_3 M_i\ )\ /\ c_4 T_i$, where the coefficients c_1, c_2, c_3, c_4 are the weighting factors for the corresponding parameters.

- Elects the node i as the cluster-head if it has M_i in the range of its minimum M_{Min} and maximum M_{Max} threshold of nodes and has the highest weight among its neighbors.
- Sets its threshold $W_{Th} = \mathrm{c}W_i$, where c is the reduction factor to readjust the threshold.
- The cluster-head keeps computing its weight and when the weight goes down to its threshold W_{Th}, it triggers the cluster-head election procedure.

To save energy, we do not periodically reform clusters. In each round, cluster-head recomputes its weight and compares with its threshold value. If W_i of cluster-head i is higher than its W_{Th} value then it keeps functioning as head. if $W_i < W_{Th}$ then it checks whether its W_i is also lower than any of its member node weight. If so, it withdraws itself to function as cluster-head and cluster election procedure is initiated.

The pseudo-code of the operations executed by a sensor node in each round of cluster formation is reported in Algorithm 1.

Algorithm 1. Elect Cluster-head

```
1: Pseudo-code executed by each node N in each round
2: W_max = 0
3: for all i in Λ(n) do
4:    if W_max < W_i then
5:       W_max = W_i
6:    end if
7: end for
8: W_i = my-weight()
9: if status = NONE then
10:    if W_i > W_max then
11:       announce-head()
12:       W_th = W_i × c
13:       where c is the threshold factor
14:    else if status = HEAD then
15:       if W_i < W_th then
16:          if W_i < W_max then
17:             withdraw-head()
18:          else
19:             W_th = W_i × c
20:          end if
21:       end if
22:    end if
23: end if
```

If nodes could not join any cluster during the first phase then DAWC accommodates these DN nodes as follows. When a high weight node in a group of dangling nodes have the number of neighbor nodes smaller than the lower limit M_{Min}, it decreases this value locally by one and then retires three times. Each

try is made during the periodic *hello* beacon. It continues until its M_i becomes equal to M_{Min} or it joins any cluster. If M_i reaches to M_{Min} then it announces itself as cluster-head and the other dangling nodes have chance to join this head. In this way, M_{Min} is reduced to manage the sparse zone of sensors. While the M_{Max} is set to disallow the nodes to make the cluster-heads overloaded in dense zone.

2.3 Neighboring Cluster Discovery

The sink or actors can be multi-hop away from the source clusters, packets are then forwarded through intermediate clusters. Clusters are linked with each other to provide multi-hop cluster routing. Some member nodes within a cluster can hear the members of neighboring clusters or heads, such nodes act as *gateways*. It is also possible that there will be multiple *gateways* between two clusters. Cluster-heads keep record of all of these *gateways*.

We build a set of forwarding gateway nodes GS, for each cluster-head, for routing packets to neighboring clusters. Let SM_i be the set of members of cluster-head H_i and SM_j be the set of members of neighboring cluster-head H_j. H_i maintains a set of gateway nodes GS_i such that

$$GSi(Hi) = \{x \in SM_i / H_j \in \Lambda(x) \quad \forall y \in SM_j \wedge y \in \Lambda(x) \quad \forall i \neq j\}$$

Where $\Lambda(x)$ is the set of neighbors of node x. A member node x of cluster-head H_i belongs to the gateway set GS_i of head H_i if either H_j or some member y of H_j exists in the neighbors set of x. The attributes of the elements of GS_i are $\{AdjacentHead, Energy, Delay, Hops\}$. These attributes help the cluster-heads in selecting a particular item from the set GS. We will describe the selection criteria in detail in Section 3.3.

Once the cluster formation is complete, each cluster gets the neighbor clusters list along with the gateways to reach them. The route computation is discussed in the next section.

3 Delay-Constrained Energy Aware Routing (DEAR)

The main aim of RCR framework is to provide real-time routing in WSAN with least energy consumption. RCR achieves this by clustering sensors hierarchically and then selecting the path on the basis of end-to-end (E2E) deadline (τ) as well as balanced energy consumption. We propose a delay-constrained energy aware routing (DEAR) algorithm to deliver packets from the source clusters to the target nodes (Sink/Actors) in WSAN. A similar idea of delay-constrained least cost routing has been proposed in [18], [19]. Unlike these protocols, we have combined the forward-tracking and back-tracking approach to reduce the cost of path establishment. We establish a distributed single path, in which cluster-head selects the outgoing link such that the packet deadline is meet with efficient energy consumption. An energy efficient link does not merely mean the low cost link but a link that can satisfy the delay constraint and it balances the energy consumption on all the outgoing links.

3.1 Network Model

Before going into the details of the algorithm, we model the network as a connected directed graph $G = (V, E)$. The set of vertices V represents the sensor nodes, where $|V| = n$. E is the set of directed edges such that an edge $e(u \rightarrow v) \in E$ if $(u, v) \in V$. Two non-negative real value functions R(e), the available energy resource of node $v \in V$ on the outgoing link $e(u \rightarrow v) \in E$, and $\Delta(e)$, the delay experienced by the data packet on the corresponding link, are associated with the edges. These real values are used to compute the weight W(u,v) of the link $e(u \rightarrow v) \in E \vee (u, v) \in V$. The weight of an edge $e(u \rightarrow v) \in E$ can be defined as follows:

$$W(u, v) = R(e)/\Delta(e), \quad where \quad u, v \in V$$

Links are presumably asymmetrical because the R(e) and $\Delta(e)$ for the link $e(u \rightarrow v)$ may not be same while going in the opposite direction of this link $e(v \rightarrow v)$. The existence of alternative paths between a pair of vertices $u, v \in V$ provides the possibility of some paths being shorter than others in terms of their associated cost. We need to find out a minimum spanning acyclic subgraph of G having high total weight.

Let s be a source node and d be a destination node, a set of links $e_1 = (s, v_2), e_2 = (v_2, v_3), ..., e_j = (v_j, d)$ constitutes a directed path P(s,d) from $s \rightarrow d$. The weight of this path is given as follows:

$$W[P(s, d)] = \sum_{e \in P(s,d)} W(e)$$

Likewise, the E2E delay experienced by following the path P(s,d) is measured as:

$$\Delta[P(s, d)] = \sum_{e \in P(s,d)} \Delta(e)$$

After the formation of clusters, we can have a vertices subset H of the set V such that the elements in H are only the cluster-heads and has an associated integral function $hops[P(h \rightarrow target)], h \in H$. Similarly, we obtain the set $GS_h \quad \forall h \in H$ as the result of linking the clusters described in Section 2.3. Each element h of set H maintains a set of outgoing links OUT_h subset of GS_h to the single destination node either sink or actor. In the next section, we describe the way of building the set $OUT_h \quad \forall h \in H$.

3.2 Sensor-Actor Coordination

The main communication paradigm in WSANs is based on the effective sensor-actor coordination. Right actions against the detected events cannot be performed unless event information is transmitted from sensors to actors. Therefore, the ultimate goal of any routing protocol in WSANs is to relay the event readings to the actors within a certain delay limit. In the classical *semi-automated*

architecture, there is a central node that is responsible to collect the readings and issue action commands to the actors responsible for the action. Unlike this approach, *automated architecture* has also been realized due to the need of immediate action on the phenomena observed in the sensory field. In the former approach, sink is the destination of events reported by all the sources and is responsible to coordinate with actors. In the latter case, the mobile actors in *automated architecture* are the targets of the event readings observed by the sensor nodes and, hence, the coordination is local.

In order to compute the delay-constrained paths efficiently, we decompose G into a minimized acyclic subgraph $\bar{G} = (\bar{V}, \bar{E})$ constituting a large acyclic region within G. $\bar{V}$ is the set of nodes either in H or belong to the GS sets of cluster-heads i.e. $\bar{V} = H \cup GS_1 \cup GS_2 ... \cup GS_k$ for k number of clusters. $\bar{E}$ is the set of directed edges such that an edge $\bar{e}(u \rightarrow v) \in \bar{E}$ if $u, v \in \bar{V}$. The length of an edge $\bar{e}(u \rightarrow v) \in \bar{E}$ may be greater than one because the members in GS may be multi-hop far from heads. For instance, an edge $\bar{e}(u \rightarrow v) \in \bar{E}$ might exist due to some member node w such that $u \rightarrow w \rightarrow v), w \notin \bar{V}, (u, v) \in \bar{V}$. Here, $R(\bar{e})$ is the least available energy of any node visited while traversing the link $\bar{e}(u \rightarrow v)$ and $\Delta(\bar{e})$ is the cumulative delay experienced by the data packet on the corresponding link.

The decomposed minimized graph $\bar{G}$ is the backbone to establish the route from the source nodes to either the sink (*semi-automated architecture*) or the actor (*automated architecture*). In the next section, we look into the formation of the graph $\bar{G}$.

Centralized DEAR (C-DEAR). In this section, we deal with the centralized *semi-automated architecture*. The sink node is stationary like sensor nodes and the path from cluster-heads to sink is built in proactive way. Sink is the destination for all the source nodes in *semi-automated architecture*. Source to sink path is divided into two phases; source to cluster head and cluster-head to sink. The first phase builds the path from source nodes to cluster-head that is done during the cluster formation in a forward tracking manner. The next phase deals with finding the path from cluster-heads to the sink using backtracking. It is activated initially by the sink during the network configuration phase and is updated periodically. To achieve this, the algorithm visits the graph G and marks all the vertices $h \in H$. A mark is associated with the life of the node, which is deleted as that vertice(node) expires. A vertex can be marked if $h \in H$ has not been already marked or the current path delay $\Delta[P(sink, h)]$ is less than the previously observed path delay. Once all the elements h of set H are marked, we build a path $P(sink, h) \quad \forall h \in H$ in proactive fashion and each element $h \in H$ set its $hops[P(sink, h)] = |P(sink, h)|$.

When h is marked, h adds the incoming link $in(x \rightarrow h), x \in V$ to the set OUT_h in reverse-topological order $out(x \rightarrow sink)$. The incoming link in may be associated with the last marked element $g \in H$ in the marking process or *null* if h is the first marked item and represents link to the root (sink node). This helps h to extend the set OUT_h by using the pre-determined set GS_h. The attributes of the elements of GS set contains the $AdjacentHead$ ID that corresponds to g.

For each element $o(m \rightarrow g) \in GS_h$, it searches for the match of g with the attribute $AdjacentHead$ of o. If there exists such element $o(m \rightarrow g)$ then h adds the link as $o(m \rightarrow g)$ to OUT_h and associate an integral value $H(o)$ apart from the other two real value functions $R(o)$ and $\Delta(o)$. Hence, the edges set $\bar{E}$ of $\bar{G}$ can be obtained as $OUT_1 \cup OUT_2, ..., \cup OUT_k$ for k number of clusters. Fig 2 illustrates the decomposed subgraph $\bar{G}$ with all the possible links to the sink node. We use the term link for set $\bar{E}$ rather than edge because vertices of set $\bar{V}$ may be connected by some intermediate vertices in V. The set OUT_h provides all the possible routes to the sink node and we exploit the multiple entries in OUT_h to provide delay-constrained energy aware routes and implicit congestion control. We describe the criteria of selecting the outgoing link in Section 3.3. The cost of marking process is O(n) and, in fact, it is the actual cost of building route from source nodes to the sink node.

Implementation. The marking process is implemented by broadcasting sink *presence* beacon in the network. That is, sink initiates the connection with the sensor nodes by broadcasting its *presence* beacon periodically, where the length of period (life of mark) is larger than the *hello* beacon. This periodic beacon helps to refresh the path because the topology of the sensor nodes is dynamic. A receiving node accepts this beacon if it meets one of the following conditions:

1. It has not already received this beacon or beacon has expired.
2. Delay of this beacon is smaller than the last received beacon.
3. The number of hops traversed by this beacon are small.

When a node receives a packet it calculates the delay and forwards the request in the direction of cluster-head. While the cluster-head forwards it to its neighboring cluster-head. Hence, each cluster-head learns the loop free path to the sink node and gets the delay value and number of hops so far.

Distributed DEAR (D-DEAR). The distributed event routing approach is imperative due to the non-existence of central controller. Events detected by the sensor nodes are directly routed to the actor nodes without the intervention of the sink node. To provide the distributed routing in the *automated architecture*, RCR proposes the distributed flavor of DEAR. In D-DEAR, we decompose the graph G into the m number of $\bar{G}$ subgraphs for each of m mobile actors. The idea is similar to C-DEAR described in detail in Section 3.2 except that we have m possible destinations. The marking process is triggered independently by all the m actors to construct m number of $\bar{G}$ representing the paths $P(h, actor_1), (h, actor_2), ..., (h, actor_m) \quad \forall h \in H$. The cost of D-DEAR is O(mn).

In order to optimize the sensor-actor coordination in the distributed environment, the marking process also propagates the current *load* factor of the actor. The *load* represents the number of sources the actor is serving at the moment. The marking criteria in D-DEAR is modified such that h accepts the mark of an actor on the basis of its Eculidian distance. The nearest one is the best candidate for marking the element h of the set H. There might be the possibility that two

or more actors reside at the same distance to h. In such case, *load* factor breaks the ties among such candidates and less-loaded actor is the winner.

Actors are location aware mobile nodes. Whenever an actor moves, it triggers the construction of graph $\bar{G}$ in addition to the periodic reconstruction of graphs. The periodic update of graphs is required due to the highly dynamic topology of the wireless sensor and actor networks because sensor nodes may be deployed at any time or their energy deplete. Hence, the algorithm updates the path proactively to reduce the chances of path failure like the path establishment.

3.3 Alternative Path Selection

Power efficiency has always been an important consideration in sensor networks. Whereas, E2E deadline τ is another constraint for real-time applications in wireless sensor and actor networks. Real-time event delivery is the main aim of our distributed routing protocol. We have described the process of building the set of outgoing links OUT. The selection of a particular link $o(m \rightarrow g) \in OUT_h, g \in H$ by the cluster-head $h \in H$ is based on the criteria to balance the load in terms of delay and energy of its member nodes. The operations of the alternative gateway selection are outlined in the Algorithm 2.

Algorithm 2. Select Outgoing Link

Ensure: Delay-constrained energy aware outgoing link $out \in OUT_h$

1: Pseudo-code executed by source cluster-head h to select an outgoing link from the set OUT_h.
2: $P = \infty$
3: **for all** $o(m \rightarrow g) \in OUT_h, g \in H$ **do**
4: **if** $time_{left}/hops[P(sink, s)] < \Delta(o)$ **then**
5: **if** $R(o) < P$ **then**
6: $P = R(n)$
7: $out = o$
8: **end if**
9: **end if**
10: **end for**

Cluster-head adds the $time_{left}$ field to its data packet that is set to τ by the source cluster-head. Each intermediate cluster-head looks for this $time_{left}$ field and selects the outgoing link accordingly by executing the above procedure. If the delay constraint can be meet through multiple links then it selects the one according to the criteria as described below:

"*The link, along which the minimum power available (PA) of any node is larger than the minimum PA of a node in any other links, is preferred*".

Every receiving node then updates the $time_{left}$ field as $time_{left} = time_{left} - delay_s$. It can be seen that the link selection criteria implicitly eliminates the congestion by alternating the links towards destination. Whenever a link is congested, the packet delay is increased and this delay is reported to the cluster-head

in successive *hello* beacon. The weight of this link is reduced and, eventually, the cluster-head reacts to it by selecting the other available link. Hence, the congestion is avoided in addition to the energy efficiency.

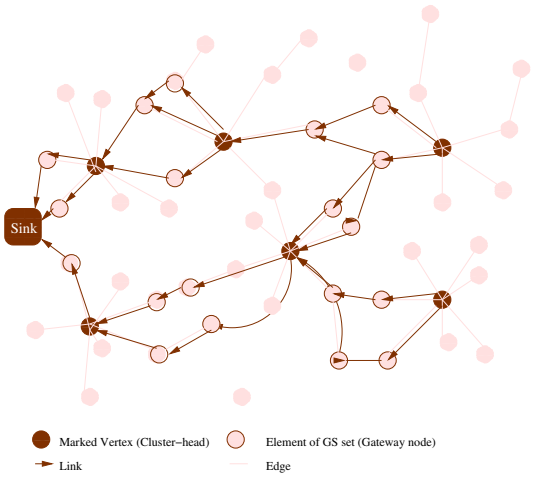


Fig. 2. Decomposition of graph G into the minimized acyclic subgraph $\bar{G}$ within the region G

4 Performance Evaluation

The performance of *RCR* is evaluated by using the network simulator ns-2 [20]. The example scenario consists of a sink node, three actors and 100 sensors randomly deployed in 200×200 meter square area. Three sensor agents developed by NRL are also placed to trigger phenomenon at the rate of 2 events per second each.

The main aim of our proposed framework is to deliver the events triggered in the sensors field to the actors within the given delay bound τ. Fig. 3 illustrates the average delay against the application deadline for the four different configurations; *direct semi-automated*, *indirect semi-automated*, *static-actors automated* and *mobile-actors automated* architecture. The mobility pattern of mobile actors is random walk. The major factor in missed-deadline is mobility of actors as clear by the mobile actors graph in Fig. 3.

Deadlines miss-ratio is an important metric in real-time systems. We measure the miss-ratio for different values of τ. Fig. 4 represents the evaluation of deadlines miss-ratio for all the configurations. The miss-ratio reaches to 0 when $\tau \approx 50ms$ for *direct semi-automated* and *static-actors automated*. This value of τ can not be set for the other configurations because there is still a significant miss-ratio.

Besides the mobility of actors, network configuration and events congestion are also the factors of missed-deadlines. It happens when the neighbor cluster of actor itself has detected events and at the same time the other clusters

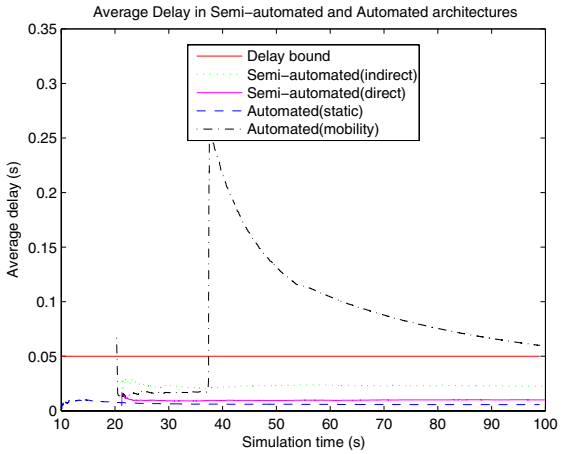


Fig. 3. Average delay in *semi-automated architecture* vs *automated architecture*

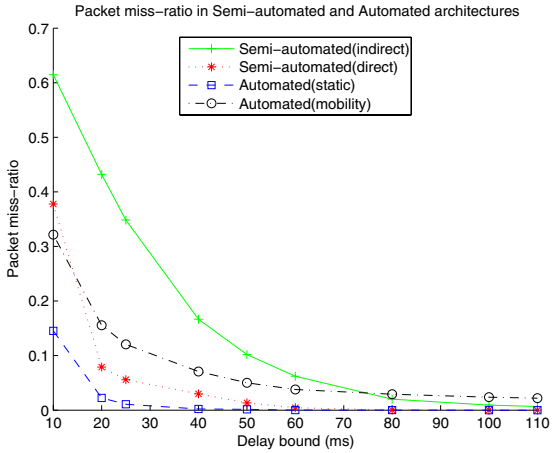
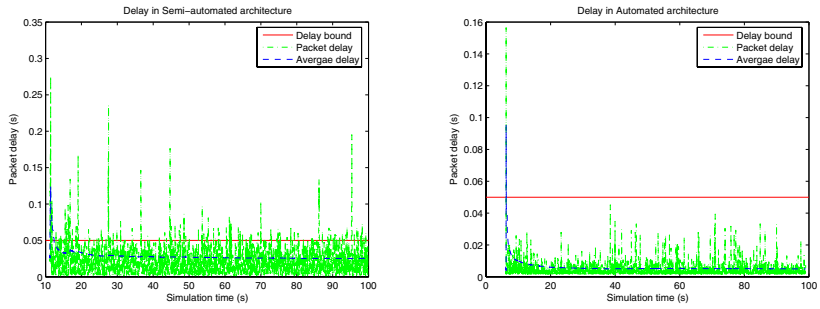


Fig. 4. Deadlines Miss-ratio in *semi-automated architecture* and *automated architecture* for $\tau = 10 - 110ms$

are also sending their readings for the same actor through this neighbor cluster. It is possible that the sensors start detecting events while the network is in the configuration state. This scenario not only causes the loss of packets but missed-deadlines as well. Fig. 5(a) represents the delay graph of *semi-automated architecture*. The events start occurring before the network is configured. There are missed-deadlines due to the non-configured network at the beginning of the graph. The same scenario for the *automated architecture* is shown in Fig. 5(b).

Although the missed-deadlines are very small in Fig. 5(b) but the peaks are initially higher than the peaks in stable configuration and causes source of jitter



(a) (b)

Fig. 5. Event to actor routing starting before configuration of 100 sensor and 3 actor nodes ($\tau = 0.05sec$)

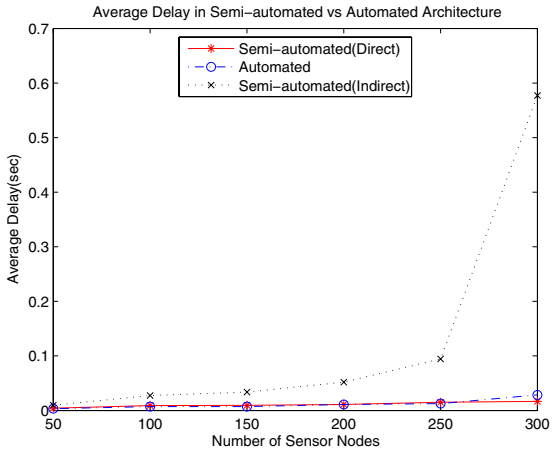


Fig. 6. Average delay by increasing number of nodes

in the traffic. The *automated architecture* requires lesser configuration time [3] than the *semi-automated architecture* and therefore, the packet delays are not much affected during the network configuration state. In Fig. 5(b), actors start receiving events at about 6 sec. This value is 12 sec. in the *semi-automated architecture*.

The scalability of the proposed framework is evaluated by increasing the number of sensor nodes in the field. Fig. 6 shows the results up to 300 nodes. The average packet delay in *indirect semi-automated architecture* increases significantly with the number of nodes as compared to the other configurations. The other two configurations (*direct semi-automated architecture* and *static-actors automated architecture*) are not much affected by deploying more sensors. The

[3] Configuration time is the time to form clusters, link them and to find out the actors for sending event readings. The network is stable after the configuration.

reason is apparent because the sink node is the only node that provides coordination among clusters and actors in indirect configuration. Whereas, the other two configurations provide direct coordination among sensors and actors.

5 Conclusion

There have been a number of routing protocols developed for WSN that claim to provide real-time routing and congestion control. However, none of them has considered the presence of actors. *RCR* addresses the issues in such heterogeneous network and provides a coordination framework for wireless sensor and actor networks. It clusters the sensors and selects a head in each cluster to minimize the energy consumption, estimates the delay budget and makes routing decisions. The communication framework works fine in *semi-automated architecture* as well as *automated architecture*. Simulation results show that *RCR* meets the E2E deadlines for real-time applications with a small value of miss-ratio.

References

1. I. F. Akyildiz and I. H. Kasimoglu, "Wireless Sensor and Actor Networks: Research Challenges," *Ad Hoc Networks, Vol. 2, Issue 4, pp. 351-367, October 2004.*
2. T. Melodia, D. Pompili, V. C. Gungor and I. F. Akyildiz, "A Distributed Coordination Framework for Wireless Sensor and Actor Networks," *ACM MobiHoc'05, May 2005.*
3. K. Akkaya, M. Younis, "An Energy-Aware QoS Routing Protocol for Wireless Sensor Networks," *23rd ICDCSW, pp. 710-715, 2003.*
4. S. Basagni, "Distributed Clustering for Ad Hoc Networks," *IEEE Proc. Vehicular Technology Conference, 1999.*
5. D. Tian and N.D. Georganas, "Energy Efficient Routing with Guaranteed Delivery in Wireless Sensor Networks," *Mobile Computing and Communications Review, 2001.*
6. Y. Yu, R. Govindan and D. Estrin, "Geographical and Energy Aware Routing (GEAR): a recursive dissemination protocol for wireless sensor networks," *UCLA/CSD-TR-01-0023, Tech. Rep., 2001.*
7. M. J. Handy, M. Haase and D. Timmermann, "Low Energy Adaptive Clustering Hierarchy with Deterministic Cluster-Head Selection," *IEEE International Conference on Mobile and Wireless, 2002.*
8. M. Chatterjee, S.K. Das and D. Turgut, "A Weight Based Distributed Clustering Algorithm for Mobile Ad hoc Networks," *HiPC 2000: pp. 511-521.*
9. A.D. Amis, R. Prakash, T.H.P. Vuong and D.T. Huynh, "Max-Min-D-Cluster Formation in Wireless Ad hoc Networks," *Proc. IEEE Infocom 2000, Tel Aviv, Israel, March 2000.*
10. C. Intanagonwiwat, R. Govindan and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," *Proc. ACM MobiCom, March 2000.*
11. T. He, J.A. Stankovic, C. Lu and T. Abdelzaher, "A Real-Time Routing Protocol for Sensor Networks," *Proc. IEEE International Conference on Distributed Computing Systems, May 2003.*

12. V. Rodoplu and T.H. meng, "Minimum Energy Mobile Wireless Networks," *IEEE JSAC, August 1999.*
13. W. R. Heinzelman, A. Chandrakasan and H. Balakrishnan, "Energy Efficient Communication Protocol for Wireless Microsensor Networks," *IEEE Proc. Hawaii International Conf. Sys. Sci. January 2000.*
14. W. R. Heinzelman, J. Kulik and H. Balakrishnan, "Adaptive Protocol for Information Dissemination in Wireless Sensor Networks," *Proc. ACM MobiCom, 1999.*
15. W. R. Heinzelman, A. Chandrakasan and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications, vol. 1, no. 4, pp. 660-670, October 2002.*
16. G. Smaragdakis, I. Matta and A. Bestavros "SEP: A Stable Election Protocol for clustered hetrogenous wireless sensor networks," *2nd International Workshop on Sensor and Actor Network Protocols and Applications (SANPA 2004).*
17. V. Mhatre and C. Rosenberg, "Homogeneous vs. heterogeneous clustered sensor networks: A comparative study," *In Proc. of 2004 IEEE International Conference on Communications (ICC 2004), June 2004.*
18. Q. Sun and H. Langendörfer, "A new distributed routing algorithm for supporting delay-sensitive applications," *Journal of Computer Communications vol. 9 no. 6 May 1998.*
19. R. Gawlick, A Kamath, S. Poltkin and K. G. ramkrishnan "Routing and admission control in general topology networks," *Tech. Report STAN-CS-TR-95-1548, Stanford University, CA, 1995.*
20. UC Berkeley, LBL, USC/ISI and Xerox PARC, "The Network Simulator ns-2," *The VINT Project, http://www.isi.edu/nsnam/ns/.*