Journal of Network and Systems Management, Vol. 14, No. 3, September 2006 (© 2006)
DOI: 10.1007/s10922-006-9039-4

Clustering in WSN with Latency and Energy Consumption Constraints

Bassam Aoun[1,2] and Raouf Boutaba[1]

Published online: 12 July 2006

Sensor networks have emerged as a revolutionary technology for querying the physical world and hold promise in a wide variety of applications. However, the extremely energy constrained nature of these networks necessitate that their architecture be designed in an energy-aware manner. Clustering is the architecture of choice as it keeps the traffic local; sensor nodes would send only to nearby cluster-head within a fixed radius, independent of the network size.

In this paper we address the problem of clustering in WSNs, subject to upper bounds on the maximum latency, the energy consumed by intermediate nodes, and clusters size. Those constraints are necessary for the reliability of the system and for extending its lifetime. We propose a polynomial time algorithm consisting of recursively computing minimum weighted dominating sets, while respecting latency and energy consumption constraints. We compare our algorithm to other alternatives and show that it consistently outperforms them.

KEY WORDS: Wireless sensor network; gateway placement; clustering; quality of service (QoS); dominating set (DS).

1. INTRODUCTION

Recent advancements in MEMS and wireless communication technologies have fostered the emergence of wireless sensor networks (WSN) as a mean for querying the physical world. A WSN consists of a dense network of low cost and energy limited devices capable of sensing certain phenomena in the area of interest and reporting data through a short-range and low-power radio. Sensors relay streams of data, either periodically or based on events, to a remote command center where the data will be processed into relevant contextual information. In addition to

[1]David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1.

[2] To whom correspondence should be addressed at David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1. E-mail: baoun@cs.uwaterloo.ca.

sensing and reporting their own data, sensor nodes cooperatively forward each other packets which travels multihops before reaching their destination.

Smart Dust [1] and PicoRadio [2] projects at UC Berkley, and WINS [3] project at UCLA are three of the research projects attempting to build low-cost and small wireless sensors. WSNs are considered for a wide range of monitoring applications such as industrial control and diagnostic monitoring [4], security [5], environment observation [6] and habitat monitoring [7].

Different WSN architectures have been designed to address specific application requirements and environmental constraints. In this paper, we consider applications of WSN where strategic placement of gateways is possible such as national reserve and large public infrastructure monitoring. In addition, we consider data gathering sensor networks as opposed to event detection sensor networks. Sensor nodes are required to periodically report to the gateway, making the whole network active rather than a single flow resulting from a specific event.

Recent studies indicate that communication consumes energy by more than two orders of magnitude than computation, storage and sensing for a majority of common applications [8]. In order to conserve energy in communicating with a remote command center, a clustering architecture of a WSN is considered. Sensor nodes would send only to a nearby cluster-head within a fixed radius, independent of the network size. The cluster head, referred to as gateway, will perform data aggregation, and relay only relevant information through long haul radio communication to the command center, as shown in Fig. 1. For that purpose, gateways will have specialized processing capabilities, long haul communication, and less restrictive energy level.

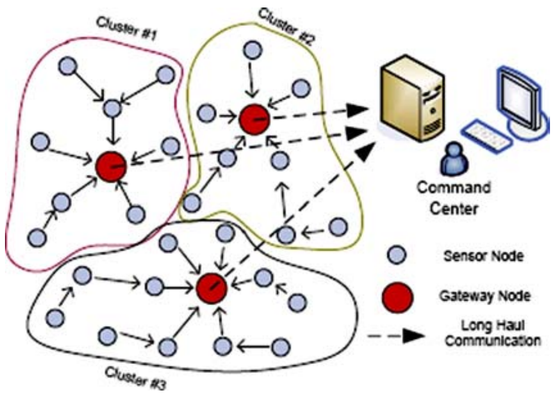

Fig. 1. Multi-gateway clustered wireless sensor network.

Various multi-hop and energy aware routing techniques have been suggested in the literature [9, 10]. However, these techniques have overhead due to route discovery and finding optimal path. These schemes are therefore more suitable for mobile ad-hoc networks or when the topology changes frequently.

In this paper we consider a tree-based proactive routing scheme which easily allows flows aggregation and minimize overhead [11]. A spanning tree rooted at the gateway can be used for traffic forwarding. Each node is mainly associated to one tree, and would attach to another tree as an alternative route in case of path failure.

For operational considerations, clustering in WSN should take into account the latency and energy consumption. Latency is an important factor for system reliability such as the case of emergency response, and accuracy of data reporting in case of high frequency periodical data updates. On the other hand, energy consumption is essential to ensure survivability of sensor nodes and hence the lifetime of the system.

In a multihop network, significant delay occurs at each hop due to contention for the wireless channel, packets processing and queuing delay. The latency is therefore a function of the number of communication hops between the source and the gateway. The latency requirement is translated into an upper bound R on the cluster size.

We consider the energy consumption at the level of gateways and sensor nodes. Recall that the gateway forwards the traffic of its cluster nodes to the command center through long haul radio communication. Although gateways are supplied with a significant amount of energy for that purpose, they are still subject to energy constraint and therefore to a limit of total forwarded traffic. In addition, the traffic forwarded by the gateway is bounded by its processing speed and its wireless throughput to the command center. Those constraints are translated into an upper bound on the cluster size S, assuming each sensor node generates an equal amount of one unit of traffic.

Recall that most of the traffic in a WSN is aggregated and forwarded by intermediate sensor nodes towards the gateways. We refer to the load on individual wireless links as relay load L in unit of traffic. We therefore enforce an upper bound on the relay load at each sensor node to mitigate the burden on the sensor nodes located close to the gateways. This limits the excessive energy spent by intermediate nodes forwarding others packets, ensuring survivability of sensor nodes and the lifetime of the system.

In this paper, we address the problem of clustering and placing gateways in WSN, subject to upper bounds on the maximum latency, power consumed by intermediate nodes, and clusters size. Those constraints are necessary for the reliability of the system and extending its lifetime. We present a polynomial time recursive algorithm to divide the WSN into clusters of bounded radius, while ensuring relay load and cluster size constraints.

The contribution of this work is the design of a novel algorithm consisting of recursively computing minimum weighted dominating sets for clustering and placing gateways in a WSN, while respecting latency and energy consumption constraints.

The rest of this paper is organized as follows. Section 2 presents an overview of related work in the literature. Section 3 describes the network model and presents the gateway placement problem and its ILP formulation. Section 4 presents a detailed description and analysis of the recursive dominating set algorithm with latency and energy consumption constraints. Experimental analysis and comparisons are presented in Section 5. Section 6 concludes this paper.

2. RELATED WORK

In addition to clustering in wireless sensor and ad hoc networks, our work inherits two other major aspects from the literature: on one hand, the capacity facility location problem (CFLP), and on the other hand, hierarchical routing in ad hoc networks.

The gateway placement problem could be considered as an instance of the more general CFLP problem which has been studied in the fields of operations research and approximation algorithms. In the past several years, a lot of work has been done in the design and analysis of approximation algorithms [12] for two facility location problems: the uncapacitated facility location problem [13], and the k-median problem [14]. In those techniques, distance is expressed in terms of Euclidean-distance, relying on the triangular inequality, rather than in terms of hop-count; consequently an upper bound on the relay load is not considered. In addition, there is no abstraction of a cluster or constraints on the cluster radius which is a necessary factor in placing gateways.

There have been numerous studies on designing hierarchical routing architecture for ad hoc networks. Routing, based on a Connected Dominating Set (CDS) forming a spine to relay routing information and data packets, is a typical technique in MANETs [15–17]. The approximation algorithms developed to solve the CDS problem are not suitable in our context: simply relaxing the problem of connecting cluster-heads leads to non-optimal solutions. In addition, the proposed schemes are concerned with 1-hop clustering, not convenient for WSNs.

Many studies [18–20] considered clustering of WSN, but their study was limited to the theoretical aspect. They considered an abstraction of the sensor network where the cluster is a circular region and the cluster head is located at the center of that region. In addition, sensor nodes are considered to be uniformly distributed over the cluster region. In that context, they formulate an optimization problem in which they associate a cost function with each type of node and aim to determine the optimal number of clusters. In this paper, we propose concrete algorithms to cluster and place gateways in any given WSN topology. In addition, instead of considering the energy requirement of the whole system, we take into account constraints on individual sensor nodes. Our work complements previous works at the theoretical level, by considering the actual formation of clusters and the placement of their respective cluster-heads.

Other works have proposed k-hop clustering algorithms [21–23] but very few have considered a complete set of constraints on clusters formation. Some considered the cluster size [21, 22], others only the cluster radius [23]. None of the previous works satisfy all the requirements of our clustering problem and rarely presents a guarantee in comparison to the optimal performance. Our problem is therefore more challenging.

In [24], the authors addressed the gateway placement problem of WSN outside the scope of clustering. The proposed approach is close to the network flow model. Wong considered two instances of the problem: minimizing the maximum latency or minimizing the total energy consumption. Constraints in terms of bounds on the relay load and cluster size are not considered. Furthermore, the proposed approximation algorithm makes no guaranty on the optimality of the solution. The additional constraints and the strict clustering model considered in this paper make our problem more challenging.

To date and to the best of our knowledge, only the work in [25] has been proposed to cluster a WSN while satisfying a latency, relay load and cluster size constraints.

Bejerano [25] successfully adopts a clustered view of the WSN and used a spanning tree rooted at each clusterhead (i.e. gateway) for message delivery. Bejerano breaks the problem of clustering and enforcing the required constraints into two subproblems. The first one seeks to find a minimal number of disjoint clusters containing all the nodes subject to an upper bound on clusters radius. The second one considers placing a spanning tree in each cluster, and clusters that violate the relay load and cluster size constraints are further subdivided. In this paper, we consider the combined problem where the spanning tree and the cluster coverage evolve in parallel as long as latency and energy consumption constraints are satisfied.

3. SYSTEM DESCRIPTION

3.1. Network Model

We consider the problem of gateway placement in the context of Wireless Sensor Networks (WSN). A WSN is represented by an undirected graph $G(V, E)$, called a connectivity graph. Each node $v \in V$ represents an sensor node with a circular transmission range of 1 unit. The *neighborhood* of v, denoted by $N(v)$, is the set of nodes residing in its transmission range. A bidirectional wireless link exists between v and every neighbor $u \in N(v)$ and is represented by an edge $(u, v) \in E$. The number of neighbors of a vertex v is called the *degree* of v, denoted by $\delta(v)$. The maximum *degree* in a graph G is called the graph degree $\Delta(G) = \Delta$

Any subset of nodes $S \subseteq V$ forms a cluster and defines a cluster graph $G_S(S, E_S)$ along with the set of links E_S between them. In this study, we only consider clusters whose corresponding cluster graph is connected. Every cluster has a cluster-head $y \in S$ and the radius r of a cluster is the maximum distance between y and any other node in the cluster.

The *distance*, denoted $d(u, v)$, between two nodes u and v is the minimum number of hops between them. The *radius* of a node v in $G(V, E)$ is the maximum distance between v and any other node. The *radius* of G is hence defined as the minimum radius in the graph. On the other hand, the *diameter* of G is the maximum distance between two arbitrary nodes (or the maximum *radius*).

For computational purposes, we use an *adjacency matrix* to represent the connectivity graph. The adjacency matrix of $G(V, E)$ is a matrix with rows and columns labeled by the graph vertices V, with a 1 or 0 in position (m, n) according to whether v_m and v_n are directly connected or not. For the undirected graph G, the adjacency matrix is symmetric.

3.2. Problem Description

In this paper we address the problem of gateway placement in WSNs required to access remote command centers. This consists in logically dividing the WSN into a set of disjoint clusters, covering all the nodes in the network. In each cluster, a node would serve as a gateway connected to the command center, serving the nodes inside the cluster and performing data fusion.

In each cluster, a spanning tree rooted at the gateway is used for traffic aggregation and forwarding. Each node is mainly associated to one tree, and would attach to another tree as an alternative route in case of path failure.

An optimal placement subject to latency and energy consumption constraints is necessary to enhance system reliability and extend its lifetime. As discussed earlier, those constraints are translated into the following: an upper bound R on the cluster radius, an upper bound S on the cluster size, and an upper bound L on relay traffic. The gateway placement problem therefore consists in logically dividing the WSN into a minimum number of disjoint clusters that cover all nodes and satisfy all three constraints.

3.3. ILP Formulation

We formulate the placement problem as an integer linear program (ILP). Let $N = V$ be the set of sensor nodes and $G \subseteq V$ be the set of gateways. Depending on the technology considered, G might be a subset of V as is the case in this paper.

We introduce a binary variable y_i to indicate whether a gateway $i \in G$ is set up. To represent gateways allocation to sensor nodes, we define another binary variable $x_{i,\,j}$ which takes the value of 1 whenever node $j \in N$ is assigned to gateway $i \in G$. $h_{i,\,j}$ represents the number of hops along the shortest path from node $j \in N$ to gateway $i \in G$ and $z_{i,j}^k$ is a binary variable indicating whether the path from i to j passes through node k. Recall that L and S are upper bounds on the relay load and cluster size, respectively. Our objective function is formulated as follows:

$$\min \sum_{i \in G_i} y_i \tag{1}$$

subject to:

$$\begin{aligned}
&\text{(a)} \quad \forall j \in N : \sum_{i \in G} x_{i,j} = 1 \\
&\text{(b)} \quad \forall j \in N, i \in G : y_i \geq x_{i,j} \\
&\text{(c)} \quad \forall j \in N : \sum_{i \in G} h_{i,j} \cdot x_{i,j} \leq R \\
&\text{(d)} \quad \forall i \in G, k \in N : \sum_{j \in N} z_{i,j}^k \leq L \\
&\text{(e)} \quad \forall i \in G : \sum_{j \in N} x_{i,j} \leq S \\
&\text{(f)} \quad \forall i \in G : y_i \in \{0, 1\} \\
&\text{(g)} \quad \forall j \in N, i \in G : x_{i,j} \in \{0, 1\} \\
&\text{(h)} \quad \forall j \in N, k \in N, i \in G : z_{i,j}^k \in \{0, 1\}
\end{aligned} \tag{2}$$

Condition (2a) denotes that each sensor node is assigned to one and only one gateway. Inequality (2b) implies that a gateway has to be set up before being assigned sensor nodes. Inequality (2c) ensures that there exists a path with at most R hops between the sensor node and the assigned gateway. This constraint implies that a cluster of bounded radius can be formed. Inequalities (2d) and (2e) provide an upper bound on the relay load and cluster size, respectively. The last three conditions indicate that y_i, $x_{i,j}$, and $z_{i,j}^k$ are binary variables.

By reducing the minimum set cover problem to the gateway placement problem given by the ILP above, one can show that it is NP-hard to find a minimum number of gateways. In practice an LP solver, such as Matlab or CPLEX, can only handle small-sized networks under the proposed model due to the fast increase in the number of variables and constraints with the network size. It will not be possible to solve the ILP for large networks due to memory constraints.

In the next section, we present a polynomial time near-optimal approximation algorithm to solve the placement problem that satisfies the latency and energy consumption constraints.

4. RECURSIVE DOMINATING SET ALGORITHM

4.1. Dominating Set Problem

The core algorithm consists of recursive approximations of the minimum Dominating Set (DS) problem. The corresponding decision problem of DS generalizes the NP-hard Vertex-Cover problem and is therefore also NP-hard [26].

Since the minimum DS problem is NP-hard, we rely on a *greedy* approach for approximation. Approximating a DS using the greedy approach was first proposed by Chvatal [26] for a more general model. The dominating set problem could be formulated as follows:

Definition 1: A dominating set of a graph $G = (V, E)$ is a subset $C \subset V$ of the nodes such that for all nodes $v \in V$, either $v \in C$ or a neighbor u of v is in C.

4.2. Algorithm Description

Our algorithm consists of recursively computing minimum dominating sets: at iteration[3] i, we compute a minimum dominating set V^i of the graph $G^{i-1} = \left(V^{i-1}, E^{i-1}\right)$ resulting from the previous iteration.

The proposed algorithm, **Recursive_DS** (V, i, R, L, S), performs recursive calls. At each iteration, V represents the dominating set of the previous iterations, i represents the iteration number, and R, L and S represent the upper bounds on cluster radius, relay load and cluster size respectively.

As shown at line **1**, we first compute the adjacency matrix of graph $G^i = (V^i, E^i)$, which is an internal representation of the connectivity graph G^i consisting of the dominating set V^i of the previous iteration $i - 1$. At iteration i, two nodes v and $u \in V^i$ are adjacent if they are i hops away. The rationale is presented in the next section.

The **While** loop from line 4 to 17 selects iteratively the node $v \in V^i$ that covers the greatest number of remaining nodes that are uncovered in G^i. The algorithm works as follows. The set U contains, at each stage, the set of remaining uncovered nodes. The set C contains the cover being constructed (i.e. the dominating nodes). Line 5 represents the greedy decision-making step. A node v is chosen that covers as many uncovered nodes as possible (with ties broken arbitrarily). Line 6 shows the resulting subset S'' composed of v and its neighbors. After v is selected, the nodes in S'' are removed from U, and v is placed in C (line 9 and 10).

[3] In this paper, iterations refer to recursive iterations. For example, iteration i refers to the ith recursive step, or recursion.

```
Algorithm 1: Recursive_DS$(V, i, R, L, S)$
1.  $Adj$ = Adjacency_matrix$(V, i)$
2.  $U \leftarrow V$
3.  $C \leftarrow [\ ]$
4.  While $U \neq [\ ]$
5.          $v$ = Greedy_selection$(Adj)$
6.          $S'' = v \cup$ Neighbors$(v, Adj)$
7.          $SP\_tree$ = Build_tree$(v, S'', Adj)$
8.          if  Satisfy_QoS$(SP\_tree, L, S)$
9.                  $C \leftarrow C \cup v$
10.                 $U \leftarrow U - S''$
11.                 $Adj$ = Adjacency_matrix$(U, i)$
12.         else
13.                 //Restricting the neighbors of $v$ in $G^i$
14.                 //such that $S''$ does not occur again
15.                 Modify_adjacency_matrix$(v, Adj)$
16.         end
17. end
18. if  Cluster_radius$(i+1) > R$
19.         return $C$
20. end
21. return  Recursive_DS$(C, i+1, R, L, S)$
```

Algorithm

When the algorithm terminates, the set C contains the set of dominating nodes at level i.

Lines 18–20 constitute the *stopping criteria* of the recursive calls. If the cluster radius of the next iteration is larger than R, we return the set C which constitutes the set of required gateways, satisfying the latency and energy consumption constraints. Otherwise, we call the function **Recursive-DS**(V, i, R, L, S) where C would represent V^{i+1} for the iteration $i + 1$.

However, before proceeding and adding v to the list of dominating nodes, we check whether a cluster rooted at v, including S'', is *feasible*. Recall that the original network is represented by $G^0(V^0, E^0)$ and clustering constraints in term of L and S should be applied to G^0. We note that each node $v \in V^i$ indexes (i.e. remembers) all the nodes in V^0 it covered in previous iterations, those nodes shall be referred to as cover(v); such that $\cup_{v^i \in V^i} \mathrm{cover}(v^i) = V^0 = V^0$.

We refer to a cluster as *feasible* if a spanning tree, rooted at v and covering all nodes in cover(S''), satisfies the relay load and cluster size constraints. At line 7, we build a spanning tree and we check if the constraints are satisfied, at line 8. If they are satisfied, we add v to the list of dominating nodes C, and remove S'' from U. Otherwise, the cover(S'') is too large and we remove an edge from E^i between v and another neighbor in V^i by modifying Adj such that the combination S'' of v does not occur again. This approach gives the chance to different feasible clusters to form before moving to the next iteration and increase the coverage of clusters. Hence, whenever the cluster radius reaches the upper bound R, all the clusters are guaranteed to satisfy the cluster size and relay load constraints.

Determining feasibility before reaching the maximum radius size, provides flexibility in term of reclustering with neighboring nodes at intermediate iterations. We will show in Section 5 that this approach leads to a much lower number of required gateways, compared to other schemes which check for the cluster size and relay load constraints after forming clusters of radius R.

4.3. Algorithm Illustration

In this section, we illustrate the above algorithm by showing its intermediate steps. We consider a random topology consisting of 93 nodes in an area of 15×15, as shown in Fig. 2. The algorithm is implemented in Matlab. The goal is to divide

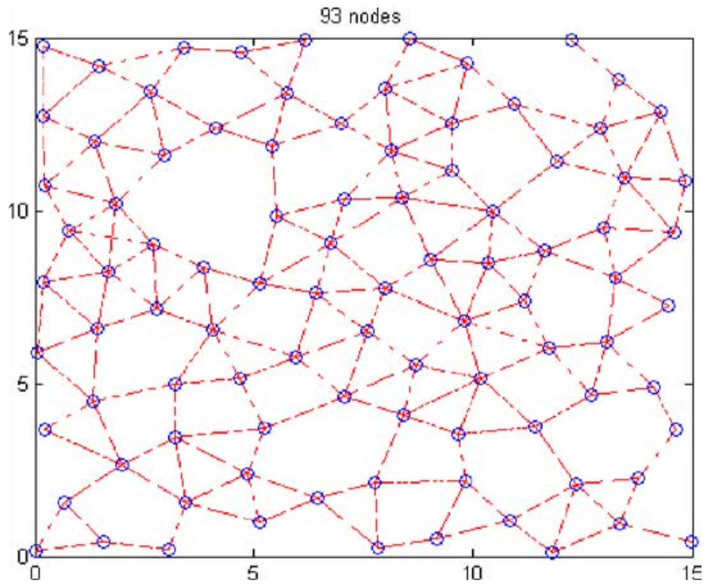

Fig. 2. Original network consisting of 93 nodes. We aim to place a minimum number of gateways satisfying the cluster radius $R = 6$ constraint.

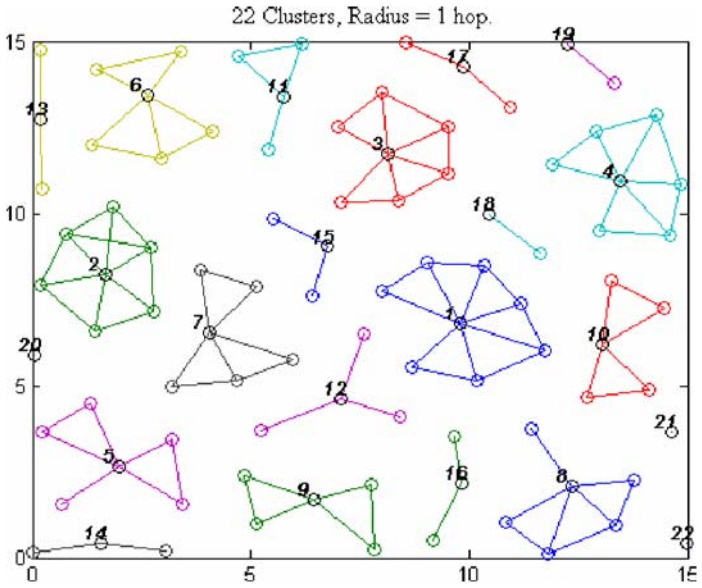

Fig. 3. The resulting cluster-heads constitute V^1, as a result of the first iteration. It consists of a minimal dominating set over $G(V, E)$) of Fig. 2.

the network into a minimum number of disjoint clusters subject to an upper bound on the radius $R = 6$. Relay load and cluster size constraints are relaxed for the sake of simplicity.

The first iteration consists in finding a minimal dominating set over *G(V, E)*. Figure 3 shows the 22 clusters, V^1, resulting from the first iteration. The index at each cluster-head represents the order chosen by the greedy algorithm at line 5. The order reflects the idea of selecting the nodes which can cover a maximum number of uncovered nodes first. We note that the number of nodes $|V^1| = 22$ moving to the next iteration is considerably lower than the original $|V^0| = 93$.

Figure 4 shows the graph $G^1(V^1, E^1)$. Recall that two vertices in G^1 are connected if they are 2-hops away in the original network G. The indices at each $v^1 \in V^1$ in Fig. 4 represent the weight computed by the greedy algorithm, at line 5, which is the degree of the node observed in Fig. 3 and consequently shows the order in which they were selected.

Figure 5 consists of finding a minimal dominating set V^2 over $G^1(V^1, E^1)$ shown in Fig. 4. The index at the left of each cluster-head shows the order by which $v^2 \in V^2$ were selected. Figure 6 shows the resulting graph $G^2(V^2, E^2)$. Since any two nodes in G^2 are at least 3-hops away, an edge $(u^2, v^2) \in E^2$ exists if u^2 and $v^2 \in V^2$ are 3-hops away. Finally, Fig. 7 shows the resulting V^3 at the

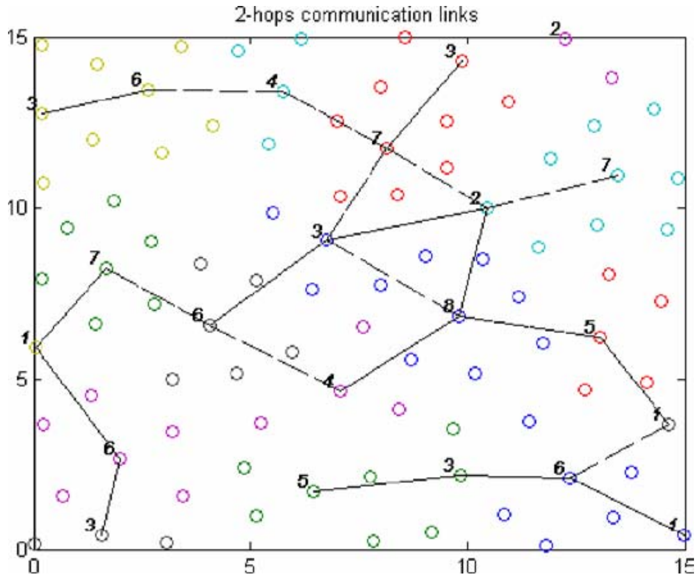

Fig. 4. $G^1(V^1, E^1)$.

third iteration. The algorithm stops since the cluster radius of the next iteration exceeds the upper bound R. The next section will present an analytical analysis of the algorithm, formulating the relation between the maximum cluster radius r_i and iteration i; the analysis will therefore justify the reason why the algorithm stops at iteration 3, given $R = 6$.

4.4. Algorithm Analysis

In this section we will denote $G^0(V^0, E^0)$ as simply $G(V, E)$. From the definition of dominating sets, we get the following corollary where V^1 denotes the dominating set of the first iteration:

Corollary 1: For any node $v \in V$, there exists a node $v^1 \in V^1$ such that $d(v, v^1) = 1$ or $v = v^1$.

The second iteration consists of finding a dominating set over V^1. The vertices $v \in V^1$ are at least 2 hops away, therefore two vertices in V^1 are adjacent if they are 2 hops away. In other word, the graph $G^1(V^1, E^1)$ sets up a connection between two vertices in V^1 if they are 2 hops away in the original graph G. At iteration $i = 2$, *Adj* will be a square matrix of size $|V^1| \times |V^1|$ and reflects the connectivity graph of G^1. At the end of iteration 2, a set V^2 will result and forms a dominating set

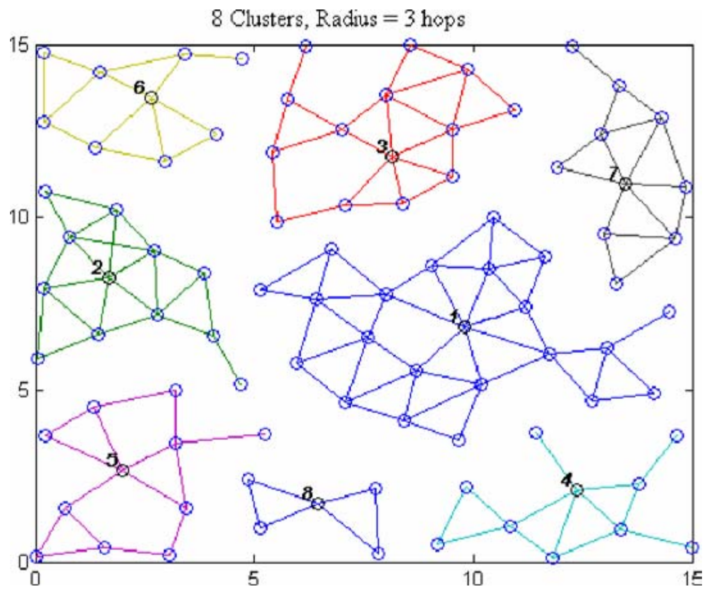

Fig. 5. The cluster-heads constitute V^2 as a result of the second iteration. It consists of a minimal dominating set over $G^1(V^1, E^1)$ of Fig. 3.

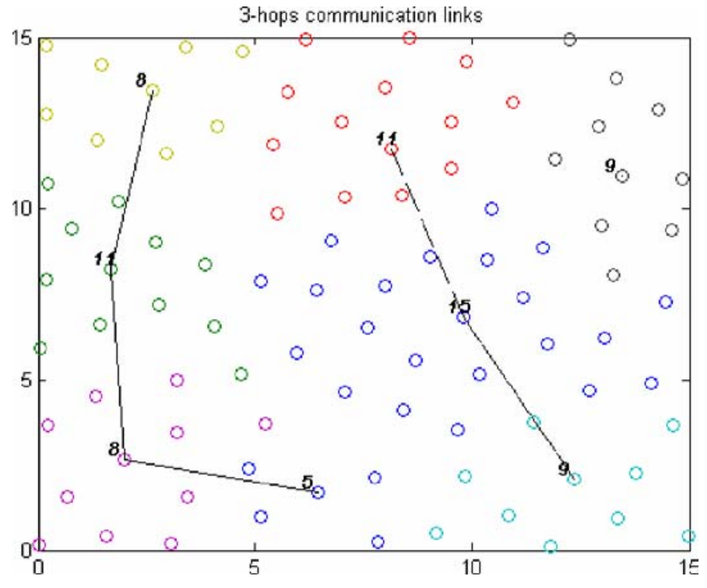

Fig. 6. $G^2(V^2, E^2)$.

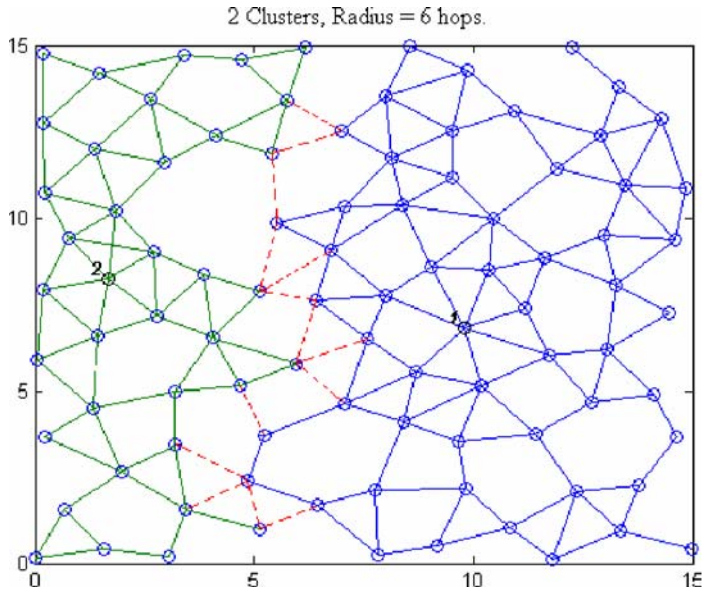

Fig. 7. This is the last iteration, as the clusters radius equals the upper bound R = *6*. $|V^2| = 2$, hence the recursive algorithm places 2 gateways at the following positions.

over V^1. V^2 will constitute a cover that can reach any $v^1 \in V^1$ in 2-hops, leading to the following corollary:

Corollary 2: For any node $v^1 \in V^1$, there exists a node $v^2 \in V^2$ such that $d(v^1, v^2) = 2$ or $v^1 = v^2$.

From *Corollary 1* and *2*, we can derive a bound for the distance of any node $v \in V$ to $v^2 \in V^2$. Since $d(v_i, v_j)$ represents the shortest distance between v_i and v_j, we can write $d(v, v^2) \leq d(v, v^1) + d(v^1, \mathrm{v}^2)$. Given that the distance between a node and itself is zero, we get $d(v, v^1) \leq 1$ and $d(v^1, v^2) \leq 2$ for all $v \in V$, $v^1 \in V^1$, and $v^2 \in V^2$. Hence, $d(v, v^2) = 3$.

Corollary 3: For any node $v \in V$, there exists a node $v^2 \in V^2$ such that $d(v, v^2) = 3$.

We now present a generalization. Since the distance between any two nodes belonging to the dominating set of iteration i is at least $i + 1$ hops, two nodes will be considered connected at iteration $i + 1$ if they are $i + 1$ hops away. Recalling that $v = v^0$, we get the following theorem by recursion:

Theorem 1: For any node $v \in V$, there exists a node $v^i \in V^i$ such that $d(v, v^i) \leq d(v, v^{\mathrm{i}-1}) + i$. Reducing the term $d(v, v^{\mathrm{i}-1})$ further recursively and given that $d(v, v^0) = 0$, we obtain the expanded form: $d(v, v^i) \leq (i \times (i + 1) \times \frac{1}{2})$.

Consequently, our algorithm will be able to guaranty an upper bound on the cluster radius of r_i at iteration i. Recall that r_i is the maximum distance $d(v, v^i)$ at iteration i. In order to hit a target radius size, one should set the initial value for r_1 adequately. Hence a general formula for r_i can be written as

$$r_i = \left(i \times (i+1) \times \frac{1}{2} \right) + (r_1 - 1) \times i. \tag{3}$$

For example, if we set the value of r_1 to 1, r_i would take the following values at consecutive iterations:

$$i = 1, 2, 3, 4, \ldots$$
$$r_i = 1, 2, 3, 6, 10, \ldots$$

Similarly, for $r_1 = 2$, we get

$$i = 1, 2, 3, 4, \ldots$$
$$r_i = 2, 5, 9, 14, \ldots$$

Assuming a spanning tree would be built in each cluster rooted at the gateway, r_i would serve as a guaranty on the upper bound on the depth of the tree.

The gateway placement algorithm can be implemented to run in time less then $\sqrt{2R} \times O\left(|V|^2\right)$, where R is the required upper bound on the cluster's radius. Each recursive iteration, line 1 to 20, of the **Recursive_DS**(V, i, R, L, S) runs in $O(|V|^2)$. That is, the recursive algorithm runs for a total of $i_{\max} \times O(|V|^2)$. From Equation (3), we can rewrite $i_{\max} \times O(|V|^2)$ as $\sqrt{2R} \times O\left(|V|^2\right))$ which expresses the run time of the gateway placement algorithm in term of required cluster radius R.

In addition to running in polynomial time, our algorithm provides a near-optimal solution, in terms of the number of gateways placed. The greedy approximation of the minimum DS problem is shown in [26] to provide a near-optimal polynomial approximation. Our recursive algorithm therefore provides a near-optimal solution at each iteration. At iteration i, we have a $\rho_i(n)$-approximation factor where $\rho_i(n) = H(\Delta)$ and Δ_i is the graph degree of the connectivity graph G^i. Therefore, the recursive gateway placement algorithm provides a $\rho(n)$ approximation where $\rho(n) = \prod \rho_i(n) = \prod H(\Delta_i)$.

4.5. Algorithm Enhancement

In this section, we refine our gateway placement algorithm by introducing weights W_m^i associated to each node v_m^i in order to effectively select dominating nodes at each iteration. The original algorithm uses a binary adjacency matrix, making the node's degree the major contributor to its weight in the greedy selection step, line 5.

We introduce the notion of distance and size of cover(u) to the greedy decision making step by considering a *weighted adjacency matrix* for the graph $G^i\left(V^i, E^i\right)$ The connection state between two vertices v_m^i and v_n^i would have a weight, $w_{m,n}^i$ The weight $w_{m,n}^i$ exists (i.e. $w_{m,n}^i \neq 0$) if v_m^i and v_n^i are adjacent in G^i The *weighted adjacency matrix* is not symmetrical because the vertices might have dominance relationship over each other. The weights are calculated as follows:

$$w_{m,n}^i = \begin{cases} 0, \text{ if } d\left(v_m^i, v_n^i\right) > i \text{ hope} \\ \frac{1}{i} \times W_n^{i-1}, \text{ if } m \neq n \\ W_n^{i-1}, \text{ if } m = n \end{cases}$$

and

$$W_m^i = \sum_{\forall n} w_{m,n}^i$$

where the initial conditions are

$$W_{m,n}^0 = \begin{cases} 0, \text{ if } d\left(v_m^i, v_n^i\right) > 1 \text{ hope} \\ 1, \text{ otherwise} \end{cases}$$

$$w_m^0 = \sum_{\forall n} w_{m,n}^0 = \delta\left(v_m\right), \text{ the node degree}$$

w_m^i corresponds to the weight used and calculated by the greedy step to select a dominated node v_m The weight w_m^i is stored with v_m in C, and used in the next section to populate the weighted adjacency matrix $w_{m,n}^{i+1}$, which is then later used in the greedy step. The idea is challenging but the implementation is very simple as no further calculations are required other than the weight originally calculated in the greedy step and carried forward by w_m^i.

5. EXPERIMENTAL ANALYSIS

5.1. Alternative Algorithms

In this study, we compare the performance of the basic and weighted recursive algorithms to two other alternatives: Iterative Greedy Dominating Set and Augmenting Placement.

5.1.1. Iterative Greedy Dominating Set

We compare our placement algorithms to the scheme proposed by Bejerano in [25]. The proposed idea in [25] is to break the problem into two subproblems and solving each one separately. The first one seeks to find minimal number of disjoint clusters that contains all the nodes and satisfy the radius constraints. In the second one, each cluster is further divided into sub-clusters if either the relay load or cluster size constraints are violated.

We use the iterative greedy dominating set heuristic [25, 26] for clustering, in the first sub-problem. This approach looks for the minimum dominating set of the power graph $G^R\left(V^R, E^R\right)$. It consists of picking iteratively the node whose R - neighborhood contains the greatest number of remaining nodes that are uncovered.

5.1.2. Augmenting Placement

Similar to [24], the *augmenting* placement represents an additional alternative for clustering. The algorithm is similar to the iterative greedy placement with respect to its internal procedures; however, it does not make *greedy* decisions regarding the next placement of additional gateways. Any placement providing subsequent coverage to uncovered nodes is considered.

5.2. Performance Evaluation

In this section, we evaluate the performance of the four different placement algorithms using various cluster radius R, relay load L and cluster size S constraints. The algorithms are evaluated according to the number of required gateways or clusters they produce.

For each setup, we generate 25 different random topologies and use the average to report performance results. Each topology consists of 175 sensor nodes placed in an area of 10×10. The communication radius CR is set to 1, and the minimum distance $d_{\min}$ separating any pair of nodes is set to 0.6. Transmission pattern is assumed to be circular and Euclidean distance is used to decide whether two nodes are in communication range. After the random topology is generated, a post processing step is performed to ensure that the resulting graph is connected. We set a threshold for that purpose. If the ratio of disconnected nodes is less than the threshold, the disconnected nodes are removed. Otherwise, we generate a new topology.

5.2.1. Effects of Relay Load

We start by studying the effect of *relay load* constraint on gateways placement. We fix the upper bound on *cluster radius* to 6 and relax the constraint on the *cluster size*. As shown in Fig. 8, the effect of *relay load* constraints is mainly pronounced when it is very limited; for $L = 4$, the *iterative greedy* and *augmenting* algorithms place *twice* the number of required gateways by the *recursive*algorithms.

On the other hand, as L increases and aggregate traffic is no longer the bottleneck, the differences in performance shrink. In addition, when the upper bound L on the *relay load* exceeds 20, the number of required gateways by each algorithm remains constant; the network is then clustered according to the limit imposed by the upper bound on *cluster radius*. We can clearly see that the

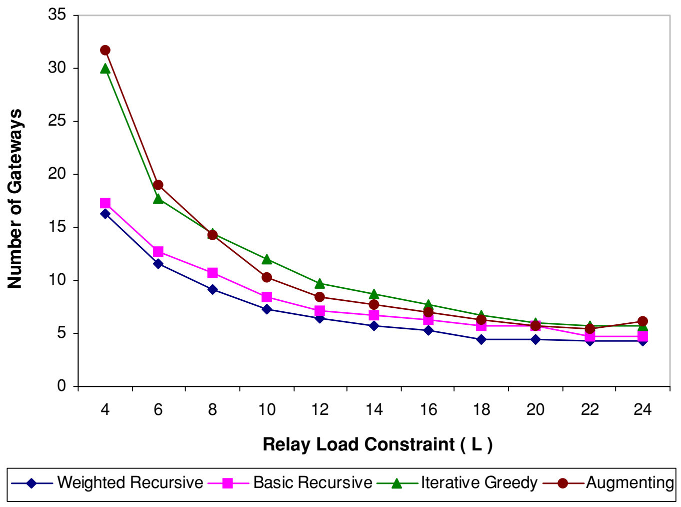

Fig. 8. Effects of the relay load constraints. Cluster radius = 6, cluster size = NaN.

weighted recursive algorithm performs best for all values of L, followed by the *basic recursive* algorithm.

5.2.2. *Effects of Cluster Size*

Next we study the impact of *cluster size* on the number of required gateways by each algorithm. The relative performance of the four algorithms is consistent with the previous analysis.

We can clearly see in Fig. 9 that the *iterative greedy* and *augmenting placement* are heavily penalized when the *cluster size* constraint is strict (i.e. S is small). As S decreases, the number of required gateways increases exponentially since each cluster is subdivided further as long as the *cluster size* constraint is violated. This results into smaller subclusters without the possibility to merge with neighboring clusters.

On the other hand, both *recursive* algorithms place much less gateways. For example, when $S = 6$, they require only 50% the number of gateways placed by the *iterative* algorithms. The reason is that clusters have the chance to merge with others forming *feasible* clusters during earlier iterations, as opposed to forming large clusters first and then subdivide them into *feasible* subclusters.

5.2.3. *Effects of Cluster Radius*

Next we compare the performance of the four algorithms as a function of *cluster radius* R. We fix the upper bound on *relay load* to 6 and relax the constraint

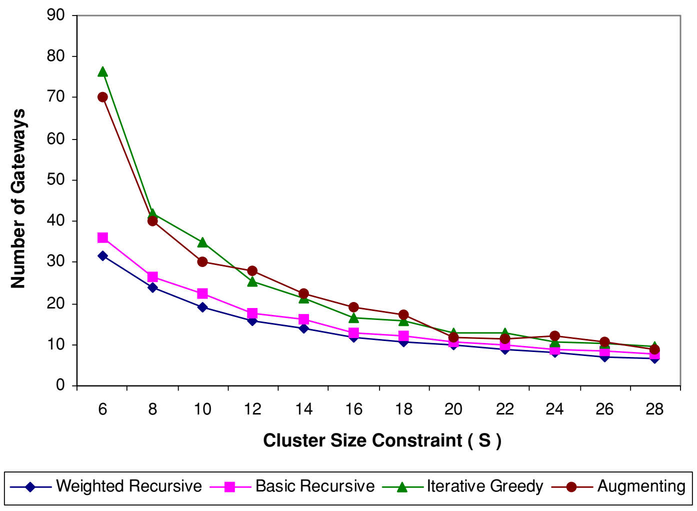

Fig. 9. Effects of the cluster size constraints. Cluster radius = 6, relay load = NaN.

on the *cluster size*. As shown in Fig. 10, the performance of the four algorithms differs significantly. We observe that the *recursive* algorithms react smoothly and consistently, placing fewer gateways as the constraint on the cluster radius get looser. Intuitively, for a given *relay load L*, as we increase the upper bound on

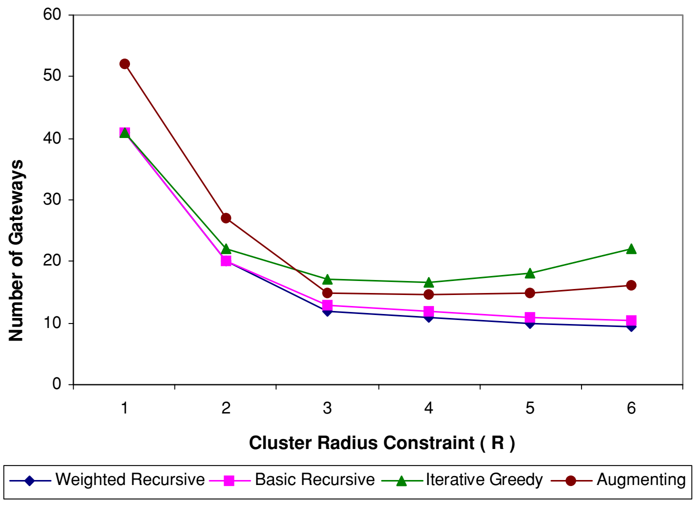

Fig. 10. Effects of the cluster radius constraints. Relay Load = 6, cluster size = NaN.

cluster radius R, we expect the number of clusters to decrease and then finally to converge.

However, this is not the case for the iterative algorithms. Not only they place more gateways, but surprisingly, they behave inconsistently when the upper bound on the cluster radius increases. Referring to Fig. 10, the *iterative greedy* algorithm places more gateways for a *cluster radius* constraint of 6 than a radius constraints of 3, 4 or 5. This problem occurs whenever the *cluster radius* is big enough to accommodate a large number of cluster nodes in the initial clustering process. However, at a later stage, whenever various constraints are imposed, the *iterative* algorithm subdivides the clusters excessively to satisfy those constraints.

5.2.4. Effects of the Communication Radius

Next, we examine the effect of communication radius *CR* of sensor nodes on the placement algorithms. We vary *CR* from 0.8 to 2 while keeping the minimum distance d_{min} between any pair of sensor nodes to 0.6. Since d_{min} is constant, varying *CR* changes the network density λ; as *CR* increases, λ increases since a sensor node can communicate with a larger set of neighboring sensor nodes.

To study the impact of *cluster size* and *relay load* individually when changing *CR*, we perform two different experiments, keeping either constraint relaxed each time. Figure 11 illustrates the number of gateways required by varying *CR* while setting the upper bounds on *cluster radius R* to 3 and *cluster size S* to 15. As

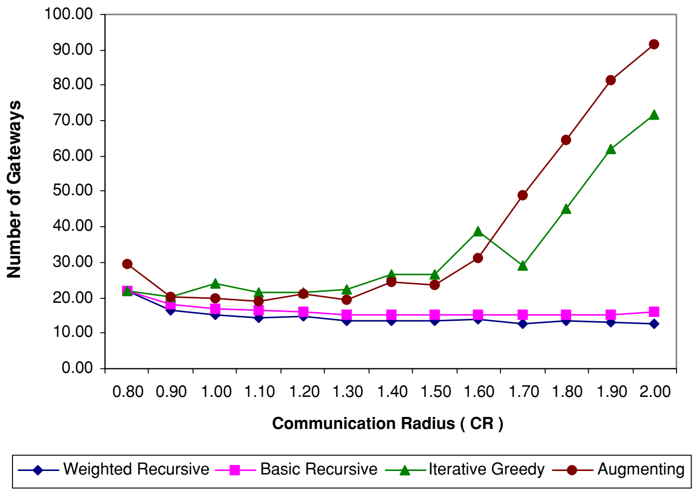

Fig. 11. Effects of the communication radius CR. Cluster radius = 3, cluster size = 15, and relay load = NaN.

shown in Fig. 11, an increase in communication radius reduces the number of gateways required by the *recursive* algorithms to cover the WSN. Such performance is expected because an increase in *CR* makes the connectivity graph of the network richer, hence the possibility to place less gateways. However, the *iterative* algorithms perform unexpectedly and place an increasing number of required gateways when the density increases. Such inconsistent performance is similar to the one observed in the previous section.

Once again, the *iterative* algorithm is penalized by forming initially very large clusters, mainly when the network is dense (i.e. large *CR*). Then, each cluster is subdivided further until the *cluster size S* constraint is satisfied, creating a large number of small clusters.

Figure 12 shows the consistent performance of all four algorithms when subject only to a *relay load* constraint *L*, instead of a cluster size constraint. Such improvement in performance is due to the fact that more sensor nodes can access the cluster head with less hops given an increase in *CR*. Therefore, forwarding traffic is reduced in the network and consequently the *relay load* constraint gets less strict. Unfortunately, placing fewer gateways when the *CR* increases comes at the expense of an increase in transmission energy for communication. We can clearly see that there exists a trade-off between the number of gateways and energy consumption for communication. A systematic cost-based analysis, similar to [28], is required to serve as guidelines to determining WSN design parameters.

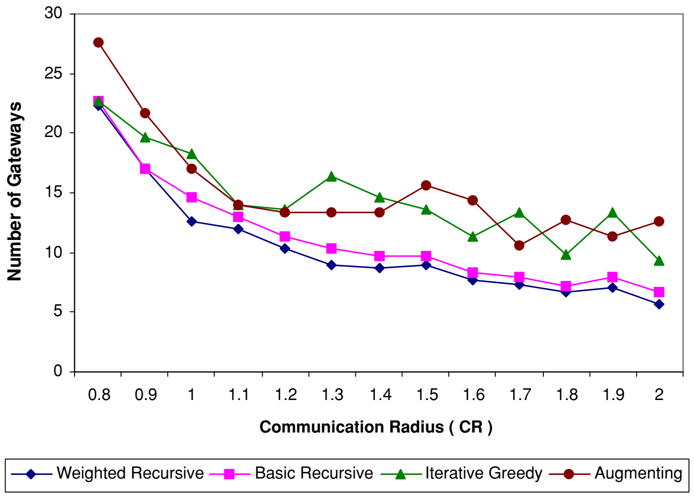

Fig. 12. Effects of the communication radius CR. Cluster radius = 3, relay load = 6, and cluster size = NaN.

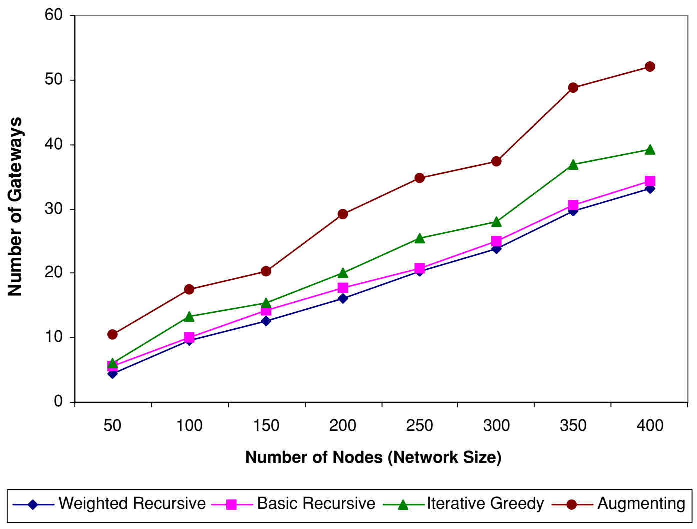

Fig. 13. Effects of the network size. Cluster radius = 3, relay load = 6, and cluster size = 15.

5.2.5. *Effects of Network Size*

Finally, we study the impact of network size on the placement algorithms. The communication radius is set back to 1 and we increase the number of sensor nodes and the respective area such that the density remains constant. As expected, an increase in the number of sensor nodes leads to a larger number of gateways required to cover the larger area.

It is very interesting to see in Fig. 13 that the number of required gateways increases *linearly* with the number of sensor nodes. This observation shows that clustering in WSN increases the scalability of the system. All placement algorithms scale similarly, however, the *weighted recursive* algorithm requires the least number of gateways to be placed, followed by the *basic recursive* algorithm, then the *greedy iterative* algorithm and finally by the *augmenting* placement.

6. CONCLUSION AND FUTURE WORK

In this paper we addressed the problem of gateways placement in WSNs required to access remote command centers. It consists in logically dividing the WSN into a set of disjoint clusters, covering all the nodes in the network. For operational considerations, the clustering problem is subject to upper bounds on the maximum latency, the energy consumed by intermediate nodes, and clusters size. First, we presented an ILP formulation for the gateway placement problem and showed that it is NP-hard. Then, we proposed a novel recursive algorithm for clustering the WSN within a bounded radius, while ensuring relay load and cluster size constraints. We showed that our algorithm runs in polynomial time and yields near-optimal results. Next, we presented a refinement to our gateway placement algorithm by introducing weights associated to each node, in order to effectively select dominating nodes at each iteration.

We compared the performance of the recursive algorithms to other alternatives and showed that they place less gateways and perform consistently when subject to different constraints and network parameters. We have shown that the recursive algorithms are robust with respect to dense networks and very strict constraints. In contrast, the iterative algorithms failed to perform consistently, placing an unreasonable high number of gateways in some circumstances and behaving unexpectedly in others. We recall that the originality of the recursive approach is to check for relay load and cluster size constraints at an early stage during clusters formation; intermediate clusters merge with neighboring clusters only when the resulting cluster satisfies the constraints, giving the choice for a wide range of possible combinations. On the other hand, by separating cluster formation and satisfying the constraints into two successive processes, the iterative algorithms require placing a large number of gateways by subdividing clusters into smaller ones without the chance to recombine with others.

The followings are possible directions for future work. First, it would be interesting to consider a decentralized implementation of the algorithm, given the locality of the computations. Second, additional constraints such as maximum distance between gateways could be imposed to allow multihop communication between the gateways and the command center. Finally, a *multi-tier* architecture could be considered to allow regional data fusion of multiple neighboring gateways recursively in a large WSN, reducing the cost of direct long haul communications. The recursive nature of our algorithm makes a *multi-tier* architecture easily computable by considering any specific intermediate dominating set as a tier.

ACKNOWLEDGMENTS

This research is partially supported by Communications and Information Technology Ontario (CITO), NORTEL, and the Natural Sciences and Engineering Research Council (NSERC).

REFERENCES

1. J. M. Kahn, R. H. Katz, and K. S. J. Pister, Next Century Challenges: Mobile Networking for Smart Dust, Proceedings MOBICOM, Seattle, 1999.
2. A. Burnstein, K. Bult, D. Chang, and F. Chang, Wireless Integrated Microsensors, Proceedings Sensors EXPO 1996, Anaheim, CA, 1996.

3. J. Rabaey, J. Ammer, J. L. da Silva and D. Patel, PicoRadio: Adhoc Wireless Networking of Ubiquitous Low-Energy Sensor/Monitor Nodes, IEEE Computer Society Workshop on VLSI, Orlando, FL, pp. 9–12, 2000.
4. Ember Corp., Reliable Wireless Networks for Industrial Systems, Technical White Paper, Published by Ember Corp.
5. C. Chiasserini, E. Magli, and D. di Elettronica, Energy Consumption and Image Quality in Wireless Video Surveillance Networks, Proceedings 13th IEEE PIMRC, 2002.
6. ALERT Systems, http://www.alertsystems.org.
7. A. Mainwaring, J. Polastre, R. Szewczyk, and D. Culler, Wireless Sensor Networks for Habitat Monitoring, In First ACM International Workshop on Wireless Workshop in Wireless Sensor Networks and Applications, 2002.
8. V. Raghunathan, C. Schurgers, S. Park, and M. B. Srivastava, Energy-Aware Wireless Microsensor Networks, in *IEEE Signal Processing Magazine*, 2002.
9. S. Singh, M Woo, and C. Raghavendra, Power-Aware Routing in Mobile Ad Hoc Networks, Proceedings of the 4th annual ACM/IEEE International Conference on Mobile computing and Networking, 1998.
10. D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, Next Century Challenges: Scalable Coordination in Sensor Networks, MOBICOM, 1999.
11. P. H. Hsiao, A. Hwang, H. T. Kung, and D. Vlah, Load Balancing Routing for Wireless Access Networks, In *IEEE INFOCOM*, 1999.
12. D. Shmoys, Approximation Algorithms for Facility Location Problems, APPROX 2000, pp. 27–32, 2000.v.
13. A. Kuehn and M. J. Hamburger, A Heuristic Program for Locating Warehouses, Management Sci., Vol. 9, pp. 643–666, 1963.
14. S. Arora, P. Raghavan, and S. Rao, Approximation Schemes for Euclidean k-Medians and Related Problems, ACM Symposium on Theory of Computing, pp. 106–113, 1998.
15. V. Bharghavan and B. Das, Routing in Ad Hoc Networks Using Minimum Connected Dominating Sets, International Conference on Communications, 1997.
16. J. Wu and H. Li, On Calculating Connected Dominating Set For Efficient Routing in Ad Hoc Wireless Networks, Workshop on Discrete Algorithms and Methods for mobile Computing and Communications, 1999.
17. Y. Chen and A. Liestman, Approximating Minimum Size Weakly-Connected Dominating Sets for Clustering Mobile Ad Hoc Networks, ACM International Symposium on Mobile Ad Hoc Networking and Computing, 2002.
18. V. Mhatre, C. Rosenberg, D. Kofman, R. Mazumdar, and N. Shroff, A Minimum Cost Heterogeneous Sensor Network with a Lifetime Constraint, IEEE Transactions on Mobile Computing, 2005.
19. J. Pan, Y. Hou, L. Cai, Y. Shi, and S. Shen, Topology Control for Wireless Sensor Networks, MobiCom, 2003.
20. S. Bandyopadhyay and E. Coyle, An Energy Efficient Hierarchical Clustering Algorithm for Wireless Sensor Networks, Proceedings of IEEE INFOCOM, 2003.
21. G. Gupta and M. Younis, Load-Balanced Clustering in Wireless Sensor Networks, IEEE International conference on communications, Anchorage, Alaska, 2003.
22. S. Banerjee and S. Khuller, A Clustering Scheme for Hierarchical Control in Multi-hop Wireless Networks, INFOCOM, 2001.
23. A. Antis, R. Prakash, T. Vuong, and D. Huynh, Max-Min d-Cluster Formation in Wireless Ad Hoc Networks, IEEE INFOCOM, 2000.
24. J. Wong, R. Jafari, and M. Potkonjak, Gateway Placement for Latency and Energy Efficient Data Aggregation, IEEE International Conference on Local Computer Networks, 2004.
25. Y. Bejerano, Efficient Integration of Multihop Wireless and Wired Networks with QoS Constraints, IEEE/ACM Transactions on Networking, 2004.

26. V. Chvatal, A Greedy Heuristic for the Set-Covering Problem, Mathematics of Operation Research, Vol. 4, No. 3, pp. 233–235, 1979.
27. D. Karger, P. Klein, and R. Tarjan, A Randomized Linear-Time Algorithm to Find Minimum Spanning Trees, J. ACM, Vol. 42, pp. 321–328, 1995.
28. V. Mhatre and C. Rosenberg, Design Guidelines for Wireless Sensor Networks: Communication, Clustering and Aggregation, Ad Hoc Networks, 2004

Bassam Aoun received his B. Eng. degree from the American University of Beirut (Lebanon) in 2004. He is pursuing his M. Math degree at the School of Computer Science, University of Waterloo (Canada). His research focuses on topology optimization and resource management in wireless mesh networks (WMN). Recent work has involved gateway placement, capacity analysis, and channel assignment in WMN.

Raouf Boutaba is an Associate Professor in the School of Computer Science of the University of Waterloo. He conducts research in network management and published more than 200 papers in refereed journals and conference proceedings. He is the Chair of the IFIP WG on Network Management, of IEEE ComSoc Information Infrastructure committee, and the Director of the IEEE ComSoc Related Societies Board. He is the editor in Chief of the IEEE eTransactions on Network and Service Management and on the editorial board of several journals. He is the recipient of several awards including the Premier's Research Excellence Award.