Gateway Placement Optimization in Wireless Mesh Networks with QoS Constraints

Bassam Aoun, Raouf Boutaba, Youssef Iraqi and Gary Kenward

***Abstract*—In a Wireless Mesh Network (WMN), the traffic is aggregated and forwarded towards the gateways. Strategically placing and connecting the gateways to the wired backbone is critical to the management and efficient operation of a WMN.**

In this paper, we address the problem of gateways placement, consisting in placing a minimum number of gateways such that QoS requirements are satisfied. We propose a polynomial time near-optimal algorithm which recursively computes minimum weighted Dominating Sets (DS), while consistently preserving QoS requirements across iterations. We evaluate the performance of our algorithm using both analysis and simulation, and show that it outperforms other alternative schemes by comparing the number of gateways placed in different scenarios.

***Index Terms*— Wireless mesh networks, gateways placement, clustering, approximation algorithms.**

I. Introduction

WIRELESS is well established for narrowband access systems, but its use for broadband access is relatively new. Wireless mesh architecture is a first step towards providing high-bandwidth network coverage. Mesh architecture sustains signal strength by breaking long distances into a series of shorter hops. Intermediate nodes not only boost the signal, but cooperatively make forwarding decisions based on their knowledge of the network. Such architecture provides high network coverage, spectral efficiency, and economic advantage.

Recently, interesting commercial applications of wireless mesh networks (WMN) have emerged. One example of such applications is “community wireless networks” [1] [2]. Several vendors have recently offered WMN products. Some of the most experienced in the business are Nortel [3], Tropos Networks [4], and BelAir Networks [5].

Manuscript received October 1, 2005. This research is partially supported by NORTEL, the Natural Sciences and Engineering Research Council (NSERC), and Communications and Information Technology Ontario (CITO).

B. A. author is with the Computer Science Department, University of Waterloo, Waterloo, ON N2L3G1 CANADA (phone: 519-725-4924; e-mail: baoun@uwaterloo.ca).

R. B. author is with the Computer Science Department, University of Waterloo, Waterloo, ON N2L3G1 CANADA (e-mail: rboutaba@bbcr.uwaterloo.ca).

Y. I. author is with the Computer Science Department, Dhofar University, Salalah, Sultanate of Oman (e-mail: y_iraqi@du.edu.om).

G. K. author is with Nortel, Brampton, ON L6T 5P6 CANADA (e-mail: gkenward@nortel.com).

WMNs have a relatively stable topology except for occasional node failures or additions. Practically all the traffic flows either to or from a gateway, as opposed to ad hoc networks where the traffic flows between arbitrary pairs of nodes. Gateways would be connected directly to the fixed network and therefore constitute traffic sinks and sources to WMNs. Therefore, strategically placing and connecting the gateways to the wired backbone is critical to the management and efficient operation of a WMN.

The analysis of WMN scalability is based on the following scaling relationships: traffic increases with the number of nodes, and traffic also increases with the distance over which each node wishes to communicate (i.e. due to packet forwarding). In [6], Li et al. showed that λ, the capacity available to each node (i.e. the rate at which packets are originated), is bounded by

$$\lambda < \frac{C/n}{\bar{L}/r}$$

where C is the total one-hop capacity of the network, n is the number of nodes, $\bar{L}$ is the expected path length and r is the fixed radio transmission range such that $\bar{L}/r$ is the minimum number of hops to deliver packets.

The above inequality shows that as the expected path length increases, the bandwidth available for each node to originate packets decreases. Therefore, the network scales better when the traffic pattern is *local*. That is, each node sends only to nearby gateways within a fixed radius, independent of the network size. The expected path length clearly remains constant as the network size grows. Hence, for optimal performance, the WMN should be divided into disjoint clusters, covering all nodes in the network. Within each cluster, the cluster-head would serve as the gateway, connected to the wired backbone.

A tree-based routing scheme would easily allow flows aggregation and would minimize overhead, ensuring an optimal utilization of bandwidth [7]. Hence, 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, the gateway placement problem should take into account the Quality of Service (QoS) requirements such as delay and bandwidth. In a multihop network, significant delay occurs at each hop due to contention for the wireless channel, packets processing and queuing delay. The delay is therefore a function of the number of communication hops between the source and the gateway. The delay constraint is translated into an upper bound *R* on the *cluster radius*, or a maximum depth *R* of the spanning tree rooted at the gateway.

Bandwidth requirements are of two forms. First, the total traffic inside each cluster is bounded by the capacity of the gateway, based on its connectivity to the internet and its processing speed. This requirement is translated into an upper bound on the *cluster size S*, assuming each AP generates an equal amount of one unit of traffic.

Guaranteeing a throughput for individual flows in a multihop wireless network is more challenging. For convenience, we assume a multi-channel WMN where interfering wireless links operate on different channels, enabling multiple parallel transmissions. The bottleneck on throughput is therefore reduced to the load of congested intermediate wireless links. Since traffic is aggregated and forwarded by intermediate APs, we refer to the load on individual wireless links as *relay load L* in unit of traffic. Therefore, the throughput requirement is translated to an upper bound on the *relay load* equal to the capacity of individual wireless links in unit of traffic.

In this paper, we address the problem of gateway placement in a WMN, aiming at placing a minimum number of gateways while ensuring the QoS requirements discussed above. We present a polynomial time near-optimal algorithm to divide the WMN into clusters of bounded *radius* under *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 placing gateways in a WMN, while ensuring the above QoS requirements.

The rest of this paper is organized as follows. Section 2 presents an overview of related works. Section 3 describes the network model, presents the gateway placement problem and provides its ILP formulation. Section 4 presents a detailed description and analysis of the recursive dominating set algorithm with QoS constraints. Experimental analyses and comparison to alternative approaches are performed in Section 5. Section 6 concludes this paper.

II. Related Work

Our work inherits two major concepts from the literature: the capacity facility location problem (CFLP); and clustering and 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 on the design and analysis of approximation algorithms [8] for two facility location problems: the uncapacitated facility location problem [9], and the k-median problem [10]. In those techniques, distance is expressed in terms of Euclidean-distance (relying on the triangular inequality) rather than in terms of hop-count, and 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 architectures 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 [11] [12] [13]. 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, which defeats the purpose of WMN.

Other works have proposed *k*-hop clustering algorithms [14] [15] but none of them satisfy all the requirements of our clustering problem and rarely present a guarantee in comparison to the optimal performance.

To date and to the best of our knowledge, very few schemes have been proposed to integrate the WMN with the wired backbone.

In [16], Wong et al. addressed the gateway placement problem in two separate settings: either minimizing communication delay or minimizing communication cost. For each setting, they propose different statistically tuned heuristics, using the same strategy: at each step they decide which of the candidate gateways will be eliminated from further consideration. QoS constraints in terms of bounds on the *relay load* and *cluster size* are not considered. Furthermore, the proposed approximation algorithm gives no guarantee on the optimality of the solution. The additional QoS constraints considered in this paper make the problem more challenging.

In [17], Chandra et al. addressed the problem of gateway placement, aiming at minimizing the number of gateways while guaranteeing AP's bandwidth requirements. They considered the problem as an instance of the network flow problem, allowing multipath routing. However, when constraints on communication path length are imposed, the proposed greedy heuristics leads to non-optimal solutions and hence no guarantee on performance. In addition, the iterative greedy approach makes the load of the gateways unbalanced, since gateways are placed whenever others are fully served. Finally, a clustered view of the WMN is not considered, making the design less suitable to our context.

The most relevant work to ours is the one in [18]. Bejerano successfully adopts a clustered view of the WMN and used a spanning tree rooted at each clusterhead (i.e. gateway) for message delivery. Bejerano breaks the problem of clustering and ensuring QoS 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* or *cluster size* constraints are further subdivided. In this paper, we consider the combined problem where the spanning tree and cluster coverage evolve in parallel as long as QoS requirements are satisfied. We show that the number of gateways required by our algorithm, subject to the same QoS requirements, is reduced by almost half in some cases, thus leading to a significant saving in deployment cost.

III. System Description

A. *Network Model*

We consider the problem of gateway placement in the context of Wireless Mesh Networks (WMN). A WMN is represented by an undirected graph $G(V,E)$, called a connectivity graph. Each node $v \in V$ represents an Access Point (AP) 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 $\Delta(G) = \Delta$.

The distance, denoted by $d(u,v)$, 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.

B. *Problem Description*

In this paper we address the efficient integration of the WMN with the wired network for Internet access, while ensuring QoS requirements. This consists in logically dividing the WMN into a set of disjoint clusters, covering all the nodes in the network. In each cluster, a node would serve as a gateway, connected directly to the wired network, and serving the nodes inside the cluster.

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.

For operational reasons, the gateway placement or clustering problem is subject to QoS constraints. As discussed earlier, the QoS 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 WMN into a minimum number of disjoint clusters that cover all nodes and satisfy all three QoS constraints.

C. *ILP Formulation*

We formulate the placement problem as an integer linear program. Let $N=V$ be the set of APs and $G \subseteq V$ be the set of gateways. G is a subset of V as is the case in Nortel solution [3]. We introduce a binary variable y_i to indicate whether a gateway $i \in G$ is set up. To represent gateways allocation for APs, we define another binary variable $x_{i,j}$ which takes the value of 1 whenever AP $j \in N$ is assigned to gateway $i \in G$. $h_{i,j}$ represents the minimum number of hops between AP $j \in N$ and gateway $i \in G$. $z_{i,j}^k$ 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* constraints, respectively. Our objective function is formulated as follows:

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

Subject to:

$$\text{(a)} \quad \forall j \in N : \sum_{i \in G} x_{i,j} = 1$$

$$\text{(b)} \quad \forall j \in N, i \in G : y_i \geq x_{i,j}$$

$$\text{(c)} \quad \forall j \in N : \sum_{i \in G} h_{i,j} \cdot x_{i,j} \leq R$$

$$\text{(d)} \quad \forall i \in G, k \in N : \sum_{j \in N} z_{i,j}^k \leq L$$

$$\text{(e)} \quad \forall i \in G : \sum_{j \in N} x_{i,j} \leq S$$

$$\text{(f)} \quad \forall i \in G : y_i \in \{0,1\}$$

$$\text{(g)} \quad \forall j \in N, i \in G : x_{i,j} \in \{0,1\}$$

$$\text{(h)} \quad \forall j \in N, k \in N, i \in G : z_{i,j}^k \in \{0,1\}$$

Condition (a) denotes that each AP is assigned to one and only one gateway. Inequality (b) implies that a gateway has to be set up before being assigned APs. Inequality (c) ensures that there exists a path with at most R hops between the AP and the assigned gateway. This constraint implies that a cluster of bounded radius can be formed. Inequalities (d) and (e) provide an upper bound on the *relay load* and *cluster size* constraints, 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 ensures the QoS requirements.

IV. Recursive Dominating Set Algorithm

A. *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 [19].

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 [19] 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.

B. *Algorithm Description*

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

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.      $S' = v \bigcup$ Neighbors$(v, Adj)$
7.      $SP\_tree$ = Build_tree$(v, S', Adj)$
8.      if  Satisfy_QoS$(SP\_tree, L, S)$
9.          $C \leftarrow C \bigcup v$
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$(C, i+1, R, L, S)$
```

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 neighbours. After v is selected, the nodes in S' are removed from U, and v is placed in C (line 9 and 10). 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 QoS requirements. 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(u); such that $\bigcup_{v^i \in V^i} \text{cover}(v^i) = V^0$.

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 neighbour 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 increases 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 terms of re-clustering with

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

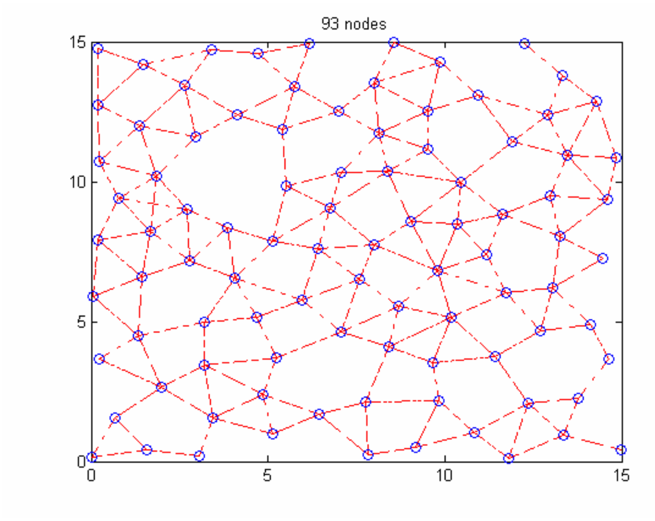

Figure 1: Original network consisting of 93 nodes. We aim to place a minimum number of gateways satisfying the cluster radius R=6 constraint.

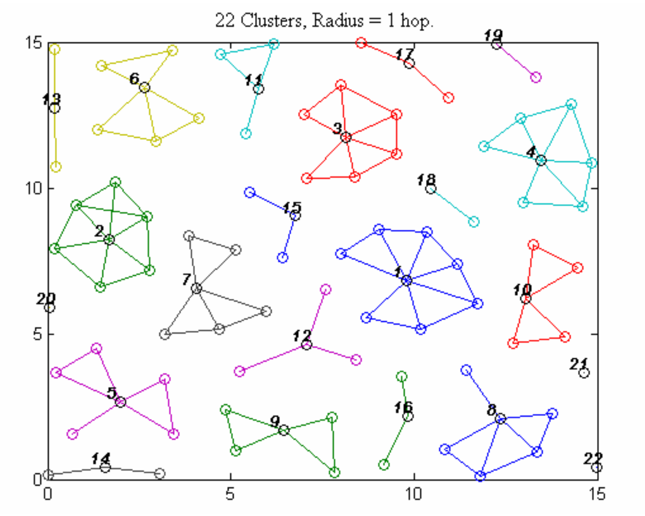

Figure 2: The resulting cluster-heads constitute V^1 , as a result of the first iteration. This consists of minimal dominating set over $G(V,E)$ of Fig. 1.

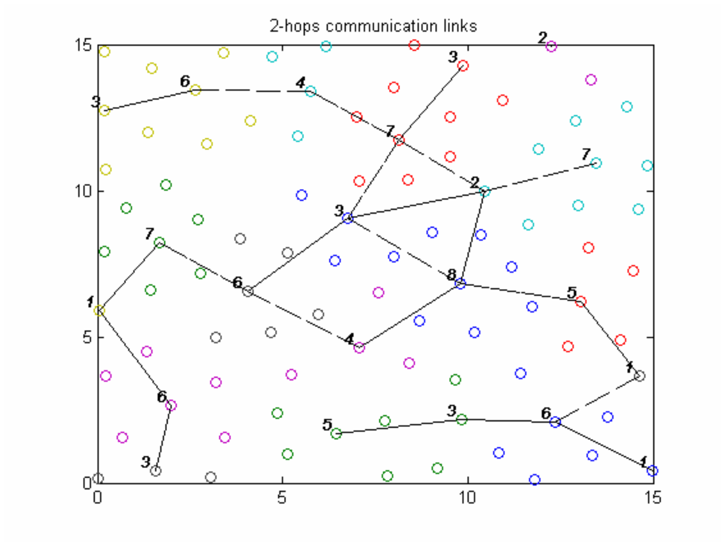

Figure 3: The graph $G^1(V^1,E^1)$

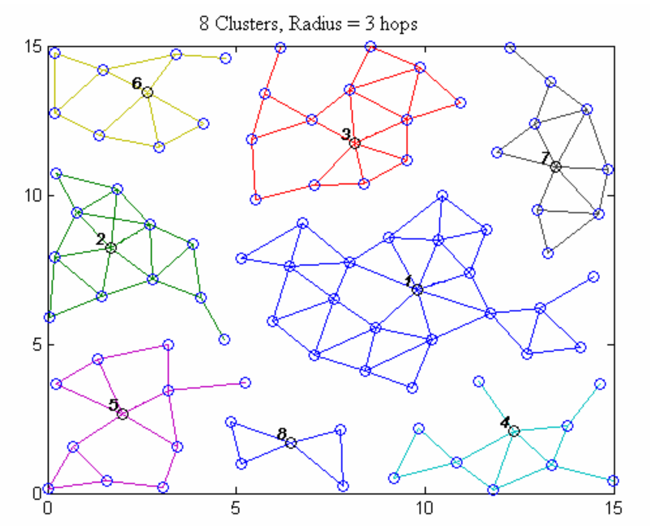

Figure 4: The cluster-heads constitute V^2 as a result of the second iteration. It consists of a minimal dominating set over $G^1(V^1,E^1)$ of Fig. 3.

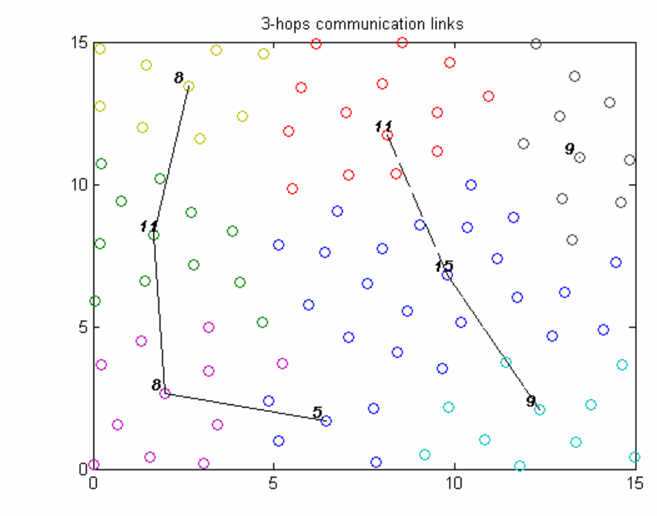

Figure 5: The graph $G^2(V^2,E^2)$

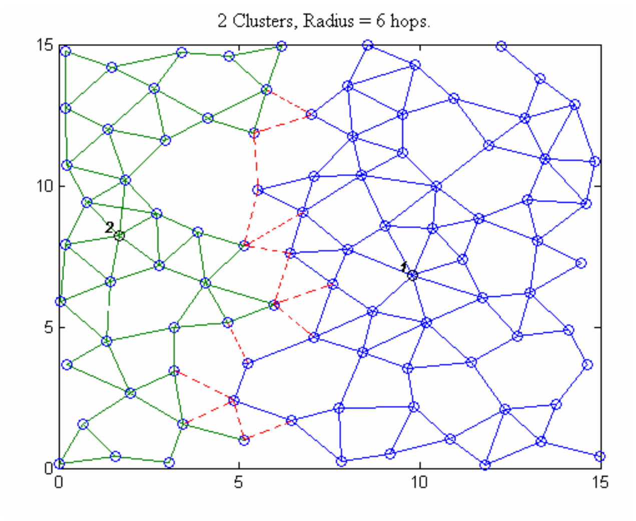

Figure 6: This is the last iteration, as the clusters radius equals the upper bound R=6. $|V^2|$=2, hence the recursive algorithm places 2 gateways at the labeled positions.

neighboring nodes at intermediate iterations. We will show in Section V 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.

C. 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 15x15, as shown in Fig. 1. The algorithm is implemented in Matlab. The goal is to divide 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 $G(V,E)$. Fig. 2 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.

Fig. 3 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. 3 represent the weight computed by the greedy algorithm, at line 5, which is the degree of the node observed in Fig. 2, and consequently shows the order in which they were selected.

Fig. 4 consists of finding a minimal dominating set V^2 over $G^1(V^1,E^1)$ shown in Fig. 3. The index at each cluster-head shows the order in which $v^2\in V^2$ were selected. Fig. 5 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. 6 shows the resulting V^3 at the 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.

D. 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 obtain 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 words, 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 $|V^1|$x$|V^1|$ 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 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 on the distance from 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 $d(v,v^2) \le d(v,v^1) + d(v^1,v^2)$. Given that the distance between a node and itself is zero, we get $d(v,v^1) \le 1$ and $d(v^1,v^2) \le 2$ for all $v\in V$, $v^1\in V^1$, and $v^2\in V^2$. Hence, $d(v,v^2) \le 3$.

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. Recal that $v=v^0$, we derive the following theorem by recursion:

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

Consequently, our algorithm will be able to guarantee 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:

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

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

$$i = 1,2,3,4,...$$
$$r_i = 1,3,6,10,...$$

Similarly, for r_1= 2, we obtain

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

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

The maximum number of iterations that can be reached corresponds to the scenario where there is only one remaining gateway serving the whole network. In the best case, assuming the gateways are optimally placed, the iterations will proceed until r_i exceeds the *radius* of the network. On the other hand, assuming the worst placement of gateways, the iteration will proceed until r_i exceeds the *diameter* of the network. For a definition of a network *radius* and *diameter*, please refer to Section III-A.

E. Algorithm Performance

In this section, we study the run time and the approximation factor of our algorithm. The body of the **While** loop can be implemented to run in time $O(|V|)$. We need $O(|V|)$ to select the node with the highest degree, at line 5. Similarly, $O(|V|)$ is required to set up a spanning tree [20], and to check whether the QoS constraints are satisfied.

On the other hand, the **While** loop will iterate a maximum of $|V|$ times until all $|V|$ nodes are covered. To show that, we assume first that F is the family of possible covers S'. The number of iterations of the **While** loop is bounded by $\min(|V|,|F|)$. However, in our context, each AP and its neighbours form a possible cover S', since each AP is a candidate dominating node. The bound is therefore reduced to $\min(|V|,|F|)=\min(|V|,|V|)=|V|$. The method **Recursive_DS**(V,i,R,L,S) can be therefore implemented to run in $O(|V|^2)$.

Theorem 2: 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.

Proof: The gateway placement algorithm have a run time of $i_{\max}\times O\left(|V|^2\right)$, where $i_{\max}$ is the number of performed recursive iterations. In the previous section, we have shown that

$$\begin{aligned} r_i &= \left(i\times(i+1)\times\frac{1}{2}\right)+(r_1-1)\times i \\ &= \frac{i^2}{2}+\frac{i}{2}+(r_1-1)\times i \end{aligned}$$

Since i and r_1 are positive numbers, $r_i > i^2/2$. Hence, we get $i<\sqrt{2r_i}$ and $i_{\max}<\sqrt{2R}$.

In addition to running in polynomial time, our algorithm can be shown to provide a near-optimal solution, in terms of the number of gateways placed.

Theorem 3: The recursive greedy dominating set is a polynomial-time $\rho(n)$-approximation algorithm, where $\rho(n)=\prod H(\Delta_i)$, Δ_i is the graph degree at iteration i of G^i, and $H(\ )$ is the harmonic function.

We refer to Chvatal [19] for the proof.

F. Algorithm Enhancement

In this section, we refine our algorithm and propose a *weighted* recursive algorithm. We compute a weight w^i to each node v^i in order to effectively select dominating nodes at each iteration. The weight would reflect the coverage of a node v^i in the original network $G(V,E)$ and the relative distance to the nodes it covers. The initial algorithm uses a binary adjacency matrix, making the node's degree in $G^i(V^i,E^i)$, instead of $G(V,E)$, the major contributor to its weight in the greedy selection step, line 5.

The *weighted recursive algorithm* uses a *weighted* adjacency matrix to represent the connection state between vertices in $G^i(V^i,E^i)$. v_m^i and v_n^i have a non-zero weight $w_{m,n}^i$ if v_m^i and v_n^i are adjacent in G^i. The *weighted* adjacency matrix is not symmetric because the vertices have dominance relationship over each other.

The weights are calculated as follows:

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

and

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

where the initial conditions are

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

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

The following is an insight to the calculation of the weights. Given the nature of the spanning tree, if a node v_m^i dominates v_n^i at iteration i, it would cover all the nodes covered by v_n^i from iteration i-1 down to iteration 1. It is therefore beneficial for every node to carry a weight corresponding to the number of nodes it covered at previous iterations.

However, the weight of a node v_n^i is not simply the sum of the nodes it covers. Instead, it is a weighted sum, inversely proportional to the distance of the nodes to v_n^i. That is, a node farther away will have a lower contribution to the total weight of node v_n^i since it negatively impacts the *relay load* constraint. The notion of distance is incorporated through multiplying W_m^{i-1} by $1/i$. The weight is therefore inversely proportional to the iteration number, suggesting that nodes covered by v_n^i at iteration i are farther than nodes covered by v_n^i at earlier iterations.

W_m^i corresponds to the weight used and calculated by the greedy step to select a dominated node v_m, at Line 5. The weight W_m^i is stored with v_m in C, at Line 9. It is then used in the next iteration to populate the *weighted* adjacency matrix $w_{m,n}^{i+1}$, which in turn is used by the greedy selection 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 steps at iteration i-1 and carried forward by w_m^i to iteration i.

V. Experimental Analysis

A. Alternative Algorithms

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

1) Iterative Greedy Dominating Set

We compare our placement algorithms to the scheme proposed by Bejerano in [18]. The idea in [18] is to break the problem into two sub-problems and to solve each one separately. The first sub-problem seeks to find a minimum 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.

The iterative greedy dominating set heuristic [18] [19] is used for clustering, in the first sub-problem. This approach looks for the minimum dominating set of the power graph $G^R(V^R,E^R)$. It consists of selecting iteratively the node whose *R*-neighborhood contains the greatest number of remaining nodes that are uncovered.

2) *Augmenting Placement*

Similar to [17] and [16], the *augmenting* placement represents another alternative for clustering. The algorithm is similar to the iterative greedy placement with respect to its internal procedure; however, it does not make *greedy* decisions regarding the next placement of additional gateways. Any placement providing subsequent coverage to uncovered nodes is considered.

B. *Performance Evaluation*

We evaluate the performance of the four different placement algorithms using various QoS parameters in terms of cluster size *S*, relay load *L* and radius size *R*. 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 nodes placed in an area of 10x10. The communication radius is set to 1, and the minimum distance separating any pair of nodes is set to 0.6 because placing APs very close to each others is not common in practice. 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.

1) *Effects of Relay Load*

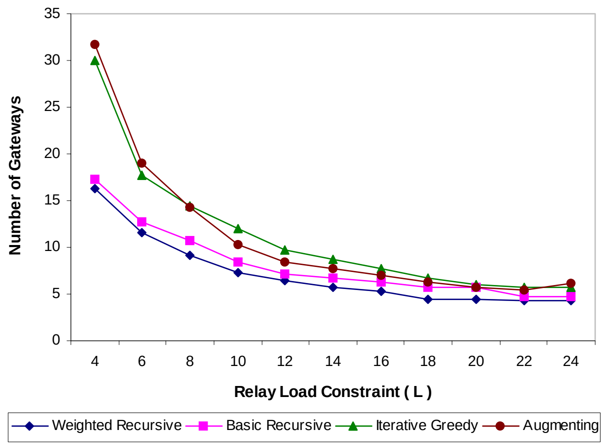

Figure 7: Effects of relay load constraints. Cluster Radius=6, Cluster Size=NaN

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. 7, 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 gateways required by the *recursive* algorithms.

On the other hand, as the constraint *L* on the *relay load* is relaxed, 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 *weighted recursive* algorithm performs best for all values of *L*, followed by the *basic recursive* algorithm.

2) *Effects of Cluster Size*

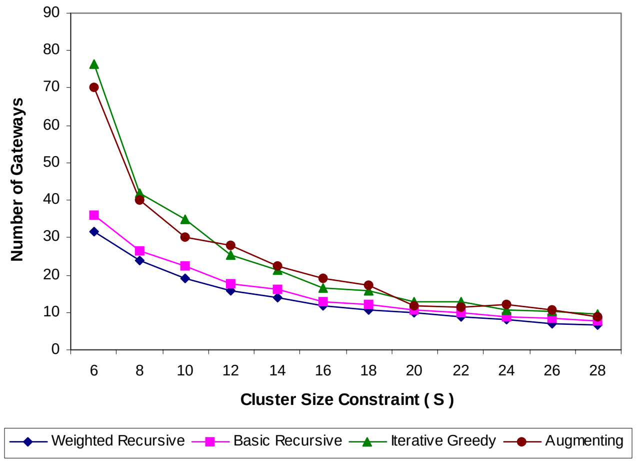

Figure 8: Effects of the cluster size constraints. Cluster radius = 6, Relay Load = NaN

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. 8 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. A large number of small sub-clusters results without the possibility to merge with neighbouring clusters. The sub-clustering effect constitutes the major *pitfall* for the iterative algorithms.

On the other hand, the number of gateways required by the *recursive* algorithms increases almost linearly as the constraint on the cluster size becomes stricter. The reason is that clusters have the chance to merge with other clusters at earlier iterations in order to form *feasible* clusters. In contrast, the iterative algorithms form large clusters first and then subdivide them individually into *feasible* sub-clusters.

As shown in Fig. 8, our *recursive* algorithms consistently yield a lower number of gateways than the *iterative* algorithms. For example, for *S*=6, they require only 50% of the number of gateways placed by the *iterative* algorithms.

3) *Effects of Cluster Radius*

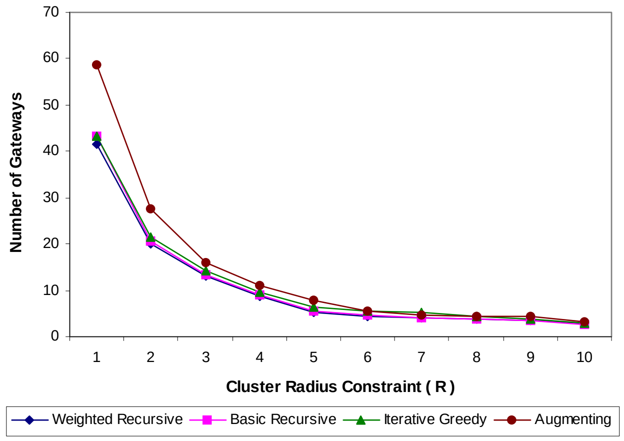

Figure 9: No relay load or cluster size constraints: L = NaN, S = NaN. Comparing clustering algorithms.

Next we compare the performance of the four algorithms as a function of *cluster radius R*. First, we illustrate their performance with no additional *relay load* or *cluster size* constraints. In such scenario, the performance reflects only the embedded clustering algorithms. As shown in Fig. 9, the four algorithms show relatively similar performance in terms of clustering alone.

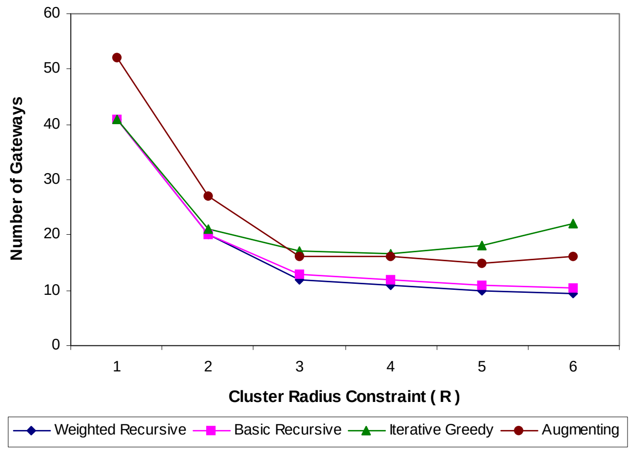

Figure 10: With relay load constraint: L=6

However, when adding a *relay load* constraint such as *L*=6, the performance of the four algorithms differs significantly as shown in Fig. 10. A similar performance to Fig. 10 is obtained when considering the addition of *cluster size* constraint instead of *relay load* constraint.

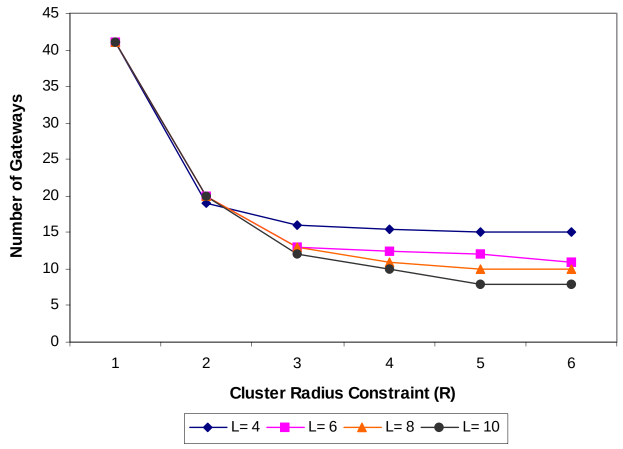

Figure 11: Weighted Recursive Algorithm: effect of the relay load L constraint on the performance, as a function of cluster radius R.

For further illustration, we plot separately the performance of the *weighted recursive* algorithm and the *iterative greedy* algorithm while varying the *relay load* constraint. We pick those two algorithms to contrast the recursive to the iterative approach. We observe in Fig. 11 that the *recursive* algorithm reacts smoothly and consistently as the *relay load* constraint *L* becomes more restrictive. Intuitively, for a given *relay load L*, as we increase the upper bound on *cluster radius R*, we expect the number of resulting clusters to decrease and then stabilize at a certain value (representing the relaxation of constraint *R*).

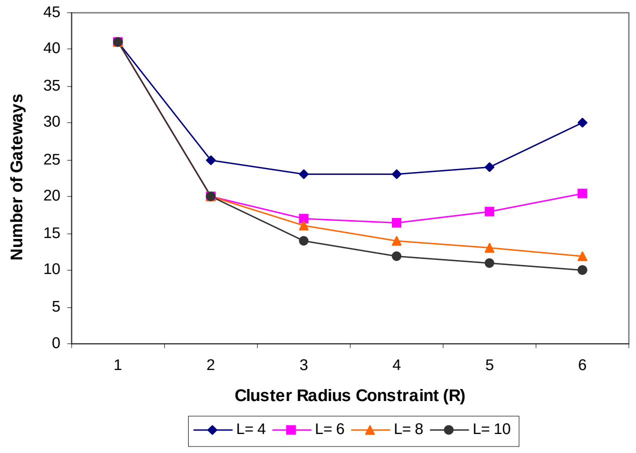

Figure 12: Iterative Greedy Algorithm: effect of the relay load L constraint on the performance, as a function of cluster radius R.

However, the *iterative* algorithm, as shown in Fig. 12, performs inconsistently as the upper bound *R* on the *cluster radius* increases. Surprisingly, if we consider *L*=4 in Fig. 12, the algorithm places more gateways for a *cluster radius* constraint of 6 than a radius constraints of 3 or 4. Such unexpected performance of the *iterative greedy* algorithm had been pointed out in [18], however interpreted differently.

This problem occurs whenever the *cluster radius* is big enough to accommodate a large number of 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. Consequently, a large number of small sub-clusters is obtained.

This problem is absent in the proposed *recursive* algorithms because clusters are not formed initially unless they satisfy all QoS constraints.

We note that the two proposed *recursive* algorithms perform better than the two other alternatives, as shown in Fig. 10. Specifically, the *weighted recursive* algorithm performs best in all scenarios.

VI. Conclusion and Future Work

In this paper, we elaborated on the importance of clustering for the efficient operation of WMNs. 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 WMN 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 compared the performance of the recursive algorithm to other alternatives. We showed that it places up to 50% less gateways, and exhibits smooth and consistent performance when subject to various QoS constraints.

The main advantage of the proposed *recursive* algorithms is that clusters have the chance to merge with other clusters at earlier iterations in order to form *feasible* clusters satisfying all QoS constraints. No cluster is formed unless it satisfies all QoS constraints. In contrast, the existing *iterative* algorithms consider first clustering the WMN with regard to the upper bound on *cluster radius R*. At a later stage, they subdivide the clusters individually until the remaining QoS constraints are satisfied. This results into a large number of small clusters without the possibility to merge with neighboring clusters.

The followings are possible directions for future work. First, taking into account wireless interference would provide a better assessment of the capacity available for APs to generate traffic, although would add complexity to the algorithm. Second, it would be interesting to consider a decentralized version of the algorithm's design, given the locality of computations. Third, it is also interesting to study the impact of topology changes, and whether it introduces any significant ripple effect.

References

[1] Seattle wireless, http://www.seattlewireless.net
[2] Bay area wireless users group, http://www.bawug.org
[3] Nortel, http://www.nortel.com
[4] Tropos Networks, http://www.tropos.com
[5] BelAir Networks, http://www.belairnetworks.com
[6] J. Li, C. Blake, D. De Couto, H, Imm, L. Morris. Capacity of Ad Hoc Wireless Networks. International Conference on Mobile Computing and Networking, 2001
[7] P. Hsiao, A Hwang, H Kung, D Vlah. Load Balancing Routing for Wireless Access Networks. IEEE INFOCOM, 1999
[8] D. Shmoys. Approximation Algorithms for Facility Location Problems. APPROX 2000, pp. 27--32, September 2000.v
[9] A. Kuehn and M. J. Hamburger. A heuristic Program for Locating Warehouses. Management Sci., 9:643--666, 1963.
[10] 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.
[11] V. Bharghavan, and B. Das. Routing in Ad Hoc Networks Using Minimum Connected Dominating Sets. International Conference on Communications, June 1997
[12] J. Wu, and H. Li. On Calculating Connected Dominating Set for Efficient Routing in Ad Hoc Wireless Networks. Workshop on Discrete Algothrithms and Methods for mobile Computing and Communications, 1999
[13] 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, June 2002
[14] S Banerjee, S Khuller. A Clustering Scheme for Hierarchical Control in Multi-hop Wireless Networks. INFOCOM 2001
[15] A Antis, R Prakash, T Vuong, D Huynh. Max-min d-Cluster Formation in Wireless Ad Hoc Networks. IEEE INFOCOM, 2000
[16] J Wong, R Jafari, M Potkonjak. Gateway Placement for Latency and Energy Efficient Data Aggregation. IEEE International Conference on Local Computer Networks, 2004
[17] R. Chandra, L. Qiu, K. Jain, and M. Mahdian, Optimizing the Placement of Integration Points in Multi-hop Wireless Networks, IEEE ICNP 2004
[18] Y. Bejerano. Efficient Integration of Multihop Wireless and Wired Networks with QoS Constraints. IEEE/ACM Transactions on Networking, 2004
[19] V. Chvatal. A Greedy Heuristic for the Set-Covering Poblem. Mathematics of Operation Research, 4(3):233--235, 1979.
[20] D. Karger, P. Klein, and R. Tarjan. A Randomized Linear-Time Algorithm to Find Minimum Spanning Trees. J. ACM, vol. 42, 1995, pp. 321-328.
[21] Resource Management in Wireless Mesh Networks, http://bcr2.uwaterloo.ca/~mesh/index.htm