Adding Capacity Points to a Wireless Mesh Network Using Local Search

Joshua Robinson, Mustafa Uysal, Ram Swaminathan, and Edward Knightly

***Abstract*— Wireless mesh network deployments are popular as a cost-effective means to provide broadband connectivity to large user populations. As the network usage grows, network planners need to evolve an existing mesh network to provide additional capacity. In this paper, we study the problem of adding new capacity points (e.g., gateway nodes) to an existing mesh network. We first present a new technique for calculating gateway-limited fair capacity as a function of the contention at each gateway. Then, we present two online gateway placement algorithms that use local search operations to maximize the capacity gain on an existing network. A key challenge is that each gateway's capacity depends on the locations of other gateways and *cannot* be known in advance of determining a gateway placement. We address this challenge with two placement algorithms with different approaches to estimating the unknown gateway capacities. Our first placement algorithm, *MinHopCount,* is adapted from a solution to the facility location problem. MinHopCount minimizes path lengths and iteratively estimates the wireless capacity of each gateway location. Our second algorithm, *MinContention,* is adapted from a solution to the uncapacitated k-median problem and minimizes average contention on mesh nodes, i.e. the number of links in contention range of a mesh node and the number of routes using each link. We show that our gateway placement algorithms outperform a greedy heuristic by up to 64% on realistic topologies. For an example topology, we study the set of all possible gateway placements and find that there is large capacity gain between near-optimal and optimal placements, but the near-optimal placements found by local search are similar in configuration to the optimal.**

I. Introduction

Multi-tier wireless mesh networks are being deployed in many cities in order to provide ubiquitous Internet access [5]. A mesh network's directional links and gateway nodes that connect the wireless mesh with the wired Internet are critical *capacity points* as their location and quantity determines the maximum throughput supported by the network. Namely, the placement of these points determines the hop-length of the paths in the network, the amount of congestion, and the available bandwidth to and from the Internet. Prior work has shown how capacity scales asymptotically with the number of gateways and nodes [10], but does not consider how to choose gateway locations in specific topologies. Likewise, greedy heuristics [4] and local search operations [14] have been developed for gateway placement, but neither incorporates wireless contention nor studies incremental deployment.

In this paper, we study the gateway placement problem and then present and evaluate two local search algorithms. The gateway placement problem is related to the facility location and k-median problems and is NP-hard. Consequently, we develop local search approximation algorithms in order to 1) provide the ability to apply local changes when incrementally upgrading a network without recomputing the full placement, and 2) provide polynomial time approximation schemes.

First, we propose an efficient technique to incorporate the effects of wireless contention and calculate the gateway-limited fair capacity of a wireless mesh network. While previous work provides a computational framework for capacity [7], we focus here on *access* networks without direct client-to-client communication, i.e., networks in which all traffic traverses the gateway. Thus, capacity points necessarily carry more traffic than other mesh nodes, and consequently we define gateway-limited capacity in terms of the contention experienced at each gateway. Our calculations are suitable for local search where a large space of possible operations must be considered and, for optimization purposes, can be separated into two components: path lengths and contention.

We next present two local search-based gateway placement algorithms adapted from the facility location problem. A key challenge is that the contention at each gateway depends on the full routing matrix. Therefore, each gateway's capacity depends on the locations of other gateways and *cannot* be known in advance of determining other gateway placements. To address this challenge, our algorithms feature two different approaches to estimating the unknown gateway capacities. The first algorithm, termed *MinHopCount*, adapts a local search algorithm for the capacitated facility location problem [13] and iteratively estimates the unknown wireless gateway capacities. The idea of local search is to carefully choose a set of gateways to *close* and *open* a set of new gateways subject to capacity and budget constraints in order to minimize the objective of interest, i.e. the average hop count. Lowering hop count generally (but not always) increases capacity and has the additional critical property of obeying the triangle inequality. The second placement algorithm, *MinContention*, adapts from a solution to the uncapacitated k-median facility

J. Robinson, and E. Knightly are with Rice University, Houston, TX (http://networks.rice.edu). M. Uysal and R. Swaminathan are with HP Labs, Palo Alto, CA. The research of Robinson and Knightly was supported by HP Labs and NSF grant CNS-0325971.

location problem and minimizes the average contention for *all* mesh nodes with provable approximation ratio of $3 + \epsilon$ [2], where the ϵ parameter determines accuracy and runtime. To minimize contention, we assign link weights equal to the amount of contention caused by each link, considering contention on all nodes instead of only gateways. Further, the link weight is the union of the set of nodes in contention range of either end of the link, which preserves the triangle inequality for $swap()$-based local search. This local search is similar to open/close except that a *swap* must open an *equal* number of gateways as it closes. The MinHopCount algorithm is more general and can handle gateways with non-uniform costs, whereas the MinContention algorithm has a built-in budget constraint, and therefore retains a provable constant-factor approximation ratio.

Lastly, we evaluate the performance of our algorithms in realistic topologies. We first validate that our capacity calculation techniques correctly rank placements, as compared to ranking placements based on measurement data from an operational urban mesh network. To compare our placement algorithms, we perform numerical simulations on the topologies of three currently deployed mesh networks: Technology For All (TFA), Chaska, and GoogleWiFi.[1] We find that our local search algorithms perform up to 64% better than a greedy algorithm and produce placements within 2% of the optimal placement found via exhaustive search. We also study the degree of similarity between gateway placements using a hop-distance metric that measures the amount of change needed to transform one gateway placement to the other. We find that the relative distance between the optimal solution and near-optimal solutions found by local search is small, which indicates the suitability of local operations.

The remainder of this paper is structured as follows. Section II introduces our mesh capacity calculation and formally defines the gateway placement problem. Section III presents the two local search algorithms and Section IV describes our evaluation of the placement algorithms. Section V discusses related work, and then Section VI concludes.

II. Adding Capacity Points

In this section, we first introduce a new technique for calculating the gateway-limited fair capacity of a mesh network. We then formulate the problem of upgrading the capacity of an existing mesh network. For ease of discussion, we refer to all capacity points as "gateways" whether they are a true wireline gateway or a wireless link that does not interfere with the remaining mesh network's resources, e.g., a directional WiFi or WiMax link to a wireline gateway (see TFA for an example of directional WiFi gateways).

[1]See tfa.rice.edu, www.chaska.net, and wifi.google.com.

In this work, we consider multi-tier wireless mesh networks, consisting of a backhaul tier for interconnection between mesh nodes, an access tier for connection between mesh nodes and clients, and a capacity injection tier to wirelessly connect the mesh nodes to the wired Internet. Further, we focus on a single-radio, single-channel backhaul and access tier architecture, although it is a simple extension to separate access tier contention and consider dual radio platforms that have a separate access and backhaul radio. We let the user-specified cost of installing a physical wire or dedicated wireless connection be different for each location and allow non-uniform capacities at each location.

A. Gateway-Limited Fair Capacity

The capacity calculation captures the impact of wireless contention on the utilization of the wireless medium in a computationally efficient manner. Our capacity calculation considers access networks where all traffic to and from clients must traverse a gateway, making the gateways bottlenecks in the network. Therefore, we focus on the performance of the gateway nodes in our definition of the gateway-limited fair capacity. The advantages of this model over previous, more general computational models [7] are 1) exact computation in polynomial time (important for evaluating many possible local search operations) and 2) extension to local search algorithms by enabling tractable approximations which optimize over one of two components of capacity definition: route lengths or contention.

A key aspect of our technique for calculating capacity is to model the wireless interface of a gateway as alternating its time between transmitting to one-hop neighbors, receiving from one-hop neighbors, and deferring to other neighbors within contention range. The time a gateway spends deferring to ongoing transmissions in contention range reduces the gateway's available capacity. Therefore, we define the *gateway-limited fair capacity* as a function of the airtime utilization of the gateways, which depends on the routes used and amount of time the routes lead to a gateway deferring. In this definition, gateway capacity is significantly affected by fairness. For example, allocating all resources to one-hop flows and none to multi-hop flows will yield the greatest capacity but would be undesirable as large portions of the network would be non-functional. Consequently, we impose a per mesh node fairness constraint, requiring that each mesh node receive its fair share of the wireless airtime at the gateway nodes.

More formally, let n be the total number of mesh nodes in the network, and m the total number of links. Define $\mathcal{G}$ as the set of all potential gateway locations, which is a subset of $\mathcal{M}$, the set of all mesh nodes. Mesh node i has a traffic demand $\mathbf{d}[i]$ that represents the aggregate demand of all the end-clients associated with it. We represent the routes used by each mesh node

to reach one or more gateways as a two-dimensional matrix $\mathbf{R}$, where $\mathbf{R}[i,j]$ indicates the amount of node i's demand that traverses link j. We designate $src(i)$ as the access tier link for mesh node i and assign $\mathbf{R}[i, src(i)] = \mathbf{d}[i]$. Our calculations ensure fairness by requiring that $\lambda\mathbf{d}[i]$ units of mesh node i's demand are served by gateways. The positive-valued λ parameter is uniform for all mesh nodes and therefore leads to weighted fair shares being enforced. We scale the demands with the λ parameter such that they are feasible, and then find the $\mathbf{R}$ matrix as solution to a transhipment problem optimizing capacity, potentially allowing multipath routing. We represent the contention caused by each link in a two-dimensional matrix $\mathbf{I}$, where $\mathbf{I}[i,j]$ indicates if link j is in contention range of node i. The $\mathbf{I}$ matrix notation extends to links that, due to physical layer shadowing, only cause contention during a fraction of time.

Our technique for calculating the amount of time a gateway is idle due to contention proceeds as follows. A link induces contention equal to the number of mesh nodes that cannot be actively transmitting or receiving when the link in question is active. Consequently, the total contention on a gateway depends on how many routes use the link and how much demand is routed over the link. We use contention as a simplification of interference, as we are concerned specifically with situations in which a node is forced to defer due to either concurrent transmission or interference. We assume a perfect MAC protocol without unfairness or hidden terminal effects.

The total contention on a gateway node $g \in \mathcal{G}$ caused by link j is $\sum_{i=1}^{n} \mathbf{R}[i,j] \times \mathbf{I}[g,j]$. The total contention on gateway g, v_g is then given by:

$$v_g = \sum_{j=1}^{m} \sum_{i=1}^{n} \mathbf{R}[i,j] \times \mathbf{I}[g,j] \tag{1}$$

The fair wireless capacity of a gateway is computed as follows. Gateway g services total demand s_g, which is the sum of demands on all links incident to gateway g, denoted by $link(g)$:

$$s_g = \sum_{i=1}^{n} \sum_{j \in link(g)} \mathbf{R}[i,j] \tag{2}$$

Expressing the capacity of gateway g as the amount of wireless time v_g required to serve s_g units of time at the wired interface, $u_g = s_g/v_g$. Thus, the total gateway-limited fair capacity is the sum of u_j terms for all $j \in \mathcal{G}$.

This sum is a lower bound of the actual gateway-limited capacity due to potential double-counting of links in contention with the gateway. This may occur if two links that contend with a specific gateway are not in contention with each other and can therefore be active simultaneously. In this case, the gateway is deferring to two links at once, whereas our calculation would count separately the defer time for both links.

B. Gateway Placement Problem

We refer to the problem of deciding how best to place a fixed number of additional capacity points in an existing mesh network so as to maximize the overall capacity improvement as the *gateway placement problem*. It is defined as follows. Let $\mathbf{G}$ be a $(0,1)$-vector of size n that indicates whether a given mesh node i is a capacity point or not. On an operational mesh network, $\mathbf{G}[i] = 1$, for all $i \in \mathcal{G}$. Let the monetary cost of installing a capacity point i be $\mathbf{f}[i]$ and the set $\mathcal{G}_0$ represent the currently deployed capacity points We define the total cost, $C(\mathbf{G})$, of installing new capacity points in the mesh network as:

$$C(\mathbf{G}) = \sum_{\forall i \notin \mathcal{G}_0} \mathbf{f}[i] \times \mathbf{G}[i].$$

We express the gateway placement problem as maximizing the network capacity given a specified budget for adding capacity points. Our formulation contrasts with previous work ([4] and [14]) which does not directly account for wireless contention effects or consider the need for upgrading existing mesh deployments.

The placement problem is difficult because it requires simultaneously solving three subproblems: 1) gateway selection, 2) client assignment to gateways, and 3) route selection. The approximation schemes we present in the next section use local search techniques to decouple these subproblems. The algorithms solve (2) and (3) together (i.e., a transhipment problem) in order to evaluate the effectiveness of all possible local operations and thereby choose operations that best solve (1).

III. Solving the Placement Problem

The gateway placement problem involves maximizing capacity directly, which can be expressed as an integer program (IP). We instead propose two local search based algorithms due to the following disadvantages of an IP: (i) an IP cannot solve the problem exactly in polynomial time, (ii) prior work has shown that a simplified version of the problem, capacitated facility location, has an unbounded integrality gap [12], and (iii) an IP is not suitable for online computation, e.g., it precludes the case of incrementally adding gateways without recomputing the locations of *every* gateway.

We therefore take an alternate approach of maximizing capacity with local search algorithms. We present two algorithms that optimize one of the two major components of our capacity calculation: the size of the routes in $\mathbf{R}$ or the impact of contention in $\mathbf{I}$ on mesh nodes. We first do this by minimizing hop count as a capacitated facility location problem with budget constraint and solving with local operations $open()$ and $close()$ and iterative capacity estimation. Our second approach is to minimize average contention as an uncapacitated k-median problem, solved by local $swap()$ operations.

A. Solving by Minimizing Hopcount

We first review the facility location problem and then describe how we map the gateway placement problem.

1) Facility Location Problem: The gateway placement problem is a generalization of the capacitated facility location problem [13], which is defined as follows. Let $\mathcal{M}$ be a set of customers, and $\mathcal{W}$ be a set of facilities. Each customer $i \in \mathcal{M}$ has a demand $\mathbf{d}[i]$. Each facility $j \in \mathcal{W}$, has a maximum capacity $\mathbf{u}[j]$ and a facility cost $\mathbf{f}[j]$. The cost matrix $\mathbf{C}[i,j]$ represents the cost of serving one unit of demand from customer i by the facility j. A facility can satisfy a customer demand only if it is open. The facility location problem then is finding a set of facilities to *open*, $\mathcal{G}$, with minimum total cost:

$$\sum_{j \in \mathcal{G}} \mathbf{f}[j] + \sum_{j \in \mathcal{G},\ i \in \mathcal{M}} \mathbf{C}[i,j] \times \mathbf{X}[i,j]$$

where $\mathbf{X}[i,j]$ denote the fraction of demand from customer i served by facility j.

In our formulation of the gateway placement problem, the facilities map to capacity points and the customers correspond to mesh nodes. The key differences are:

- The wireless capacity of each gateway depends on nearby contention, which in turn depends on the placement of other gateways. Therefore, the capacities are not known *a priori* because it would require knowledge of the final placement.
- Defining a cost function for serving mesh node i by gateway j does not preserve the triangle inequality. This cost is equal to the fair share of mesh node i at gateway j. If the customer and facility cost metrics do not preserve the triangle inequality, no constant factor approximation algorithms are known.

Despite these differences, the local search algorithms developed for the facility location problem apply to the gateway placement problem (both differences are addressed in our adaption of the algorithm, as described later).

2) Local Search Operation: We next describe the local search algorithm [13] for the facility location problem in the context of the gateway placement problem, highlighting modifications to account for gateway placement specifics. We denote s as a node and $\mathcal{T}$ as a set of nodes, which we describe how to find later in this section. The algorithm can do one of three operations to improve the solution: $add(s)$ installs a gateway at node s, $open(s,\mathcal{T})$ installs a gateway at node s and removes gateways at all nodes in set $\mathcal{T}$, and $close(s,\mathcal{T})$ removes the gateway at node s and installs gateways at all nodes in set $\mathcal{T}$.

Let us refer to the set of available gateway locations as $\mathcal{W}$, which is a site-specific subset of all mesh node locations. Let $\mathcal{G}$ represent the set of *installed* gateway locations throughout the execution of the algorithm, i.e., $\mathbf{G}[i] = 1$, if $i \in \mathcal{G}$. The local search algorithm operates as follows. We start with an arbitrary valid gateway placement and perform one of the three operations, $add()$, $open()$, and $close()$, to improve the quality of the solution. To ensure the algorithm terminates in polynomial time, we require that each step lowers the cost by at least $c(S)/p(n,\epsilon)$, where $p(n,\epsilon)$ is a chosen polynomial in n and $1/\epsilon$. Here, $\epsilon > 0$ indicates the error tolerance, and the algorithm's run time is polynomial in $1/\epsilon$.

We now review in more detail each local search operation. Because all possible combinations for set $\mathcal{T}$ cannot be evaluated in polynomial time, the algorithm instead finds a good choice for the set $\mathcal{T}$ as the solution to a knapsack problem, where $\mathcal{T}$ is found as the set of items to put in the knapsack. The operations proceed as follows:

- $add(s)$ – For all non-gateway nodes s, evaluate the cost to open a gateway at $s \in \mathcal{W}$. This cost evaluation requires solving a transhipment problem to find optimal routing matrix $\mathbf{R}$ for the set of all installed gateways in $\mathcal{G} \cup \{s\}$.
- $open(s,\mathcal{T})$ – Install gateway at node $s \in \mathcal{W}$ and remove gateways in set $\mathcal{T} \subseteq \mathcal{G} - \{s\}$, reassigning mesh nodes served by $\mathcal{T}$ to the gateway at s. Note that gateway s could already have been installed with some unused capacity.
- $close(s,\mathcal{T})$ – Remove gateway $s \in \mathcal{G}$ and install a set of gateways $\mathcal{T} \subseteq \mathcal{W} - \{s\}$. Then reassign routes destined to s to gateways in $\mathcal{T}$ without any effect on mesh nodes served by other gateways.

```
Let M be the set of all mesh nodes
Initialize u[i] values to gateway wired capacities
// Note that valid here means satisfies budget

Do {
  // Run local search algorithm with u[i] capacities
  Start with arbitrary, valid solution G
  Do {
    Foreach s ∈ M
      Find valid add(s)
      Find valid open(s, T)
        where T is solution to knapsack problem
        with knapsack size of u[s]
      Find valid close(s, T)
        where T is minimal covering knapsack
        with knapsack size of u[s]
    Calculate Δ cost for all valid operations
    Apply operation to G with best Δ cost
  } while (Δ cost ≥ C(G)/p(n, ε))
  Output G as locally optimal solution

  Calculate capacities û[i] of placement G
  Update u_cur[i] to new lower bound if û[i] < u_prev[i])
} while (Σ_{i=1}^{N} u_prev[i] − u_cur[i] ≥ φ)

Output G as solution
```

TABLE I

PSEUDOCODE FOR MINHOPCOUNT ALGORITHM.

3) Adapting MinHopCount: We next describe our modifications to allow the facility location algorithm to minimize hop count subject to a budget constraint and gateway capacities. We then describe our technique for iteratively estimating gateway capacity.

We use hop count as the cost function in our problem, which is a first-order approximation of the capacity, i.e. it reduces the contention in Eq. 1 by reducing the value of $\mathbf{R}$ entries. Another important advantage of this metric is that it preserves the triangle inequality, which is necessary for provable bounds on the local search algorithm's performance.

We also add a budget constraint to the MinHopCount algorithm, making the problem we solve a generalization of the capacitated k-median problem (more general because we allow all gateway costs, $\mathbf{f}[i]$, to be different). While there are no known constant factor approximation algorithms for the capacitated k-median problem, we show through evaluation that the algorithm performs close to optimal in realistic topologies (see Section IV).

The local search operations find a placement subject to gateway capacity constraints, but these gateway capacities are not known *a priori* because they depend on the full gateway placement. As shown in Table 1, we use lower bound estimates for the gateway capacities, $\mathbf{u}[i]$, and iteratively update the gateway capacity lower bounds after successive runs of the local search algorithm. The algorithm terminates when the current sum of the lower bound capacity estimates $\mathbf{u}_{cur}[i]$ does not decrease by more than user-chosen parameter ϕ from the previous iteration's estimate, $\mathbf{u}_{prev}[i]$. Intuitively, we capture the lower bound capacity of a near-optimal placement, which is a tighter bound than the worst-case lower bounds of all placements. The run time of the MinHopCount algorithm is polynomial in $\frac{1}{\epsilon}$ and $\frac{1}{\phi}$.

B. Solving by Minimizing Contention

Our second local search algorithm, MinContention, finds the gateway placement that minimizes the average contention in the network. We first describe the k-median problem and review a local search algorithm using local $swap()$ operations. Second, we discuss how to map the gateway placement problem to this algorithm such that we find the placement with the lowest average contention region size.

1) The k-Median Problem: The k-median problem is a variant of the facility location problem where there are only a fixed number k of facilities that can be opened. The objective is to minimize the cost of connecting all clients to a facility. We consider the uncapacitated version of the problem as there is currently no known constant factor algorithm for the capacitated k-median problem. In contrast, there is a local search algorithm for the uncapacitated k-median problem with a locality gap of $3+2/p$ [2], where the locality gap is the maximum difference between the worst local optimum and the global optimum and the parameter p controls the number of gateways the algorithm considers for simultaneous swapping. This locality gap results in an approximation ratio of $3 + \epsilon$. This local search algorithm is based on repeatedly swapping p open gateways for p unopen gateways until no swaps can improve the solution. A larger p value leads to more accurate results but with exponential increase in running time.

The main idea of the MinContention algorithm is to install k gateways to minimize the average contention on the mesh nodes, which is a function of which links contend with each node and how often those links are used in routes. As per our definition in Section II, our actual objective is to minimize the total contention on gateways, but we cannot do that because it requires knowing the full gateway placement to correctly assign link weights. We therefore solve the problem of minimizing the contention on *all* nodes as a means of approximating the gateway contentions. Note that a disadvantage of this algorithm over the previously discussed MinHopCount algorithm is that it requires identical gateway costs.

2) Swap-based Local Search: The MinContention algorithm is summarized in Table 2. The cost of a placement is the sum of the active link weights, which are each assigned to be the total number of mesh nodes in contention range of the link. Additionally, we scale the shortest paths' weight in proportion to the node's traffic demand. This allows us to take client demands into account and favor installing gateways nearer to mesh nodes with greater demand.

Find a feasible starting placement $\mathbf{G}$

Do {
  Find all valid $swap(\mathcal{S}, \mathcal{T})$
    where $\mathcal{S}$ is set of p gateways to open
    and $\mathcal{T}$ is set of p gateways to close
  Calculate Δ cost for each operation
  Apply *swap* with largest positive Δ cost
} while (Δ cost $\geq \frac{C(\mathbf{G})}{p(n,\epsilon)}$)
Output $\mathbf{G}$ as locally optimal solution

TABLE II
PSEUDOCODE FOR MINCONTENTION ALGORITHM.

3) Triangle Inequality for Contention: In order for the above algorithm to have a provable locality gap of $3 + 2/p$, the link weights must obey the triangle inequality. As previously stated, the assigned link weight is the size of the union of the sets of nodes in contention range with each endpoint of the link, which we now show preserves the triangle inequality.

Consider a triangle of three mesh node locations, a, b, and c, and the resulting three links between them, labeled AB, AC, and BC. The contention caused by link AB is less than the sum of the contention of links

AC and BC for the following reason. Let function $\Gamma()$ represent the number of nodes in contention with a node or link. By definition, we have that link AB's contention consists of the nodes in contention range of nodes a and b, resulting in $\Gamma(AB) = \Gamma(a) \cup \Gamma(b)$. The contention caused by links AC and BC is $(\Gamma(a) \cup \Gamma(c)) + (\Gamma(b) \cup \Gamma(c))$ and is smallest when node c contends with no mesh nodes. Therefore the contention is lower-bounded by $\Gamma(a)+\Gamma(b)$, which is greater than or equal to $\Gamma(AB)$, ensuring that the triangle inequality is preserved.

IV. Evaluation

In this section we examine the performance of our placement algorithms. We first examine the proposed technique for capacity calculation with measurement data. We next study the algorithms on regular grid topologies and then real topologies that underlie three deployed mesh networks, and finally study the characteristics of an optimal placement.

A. Validating Capacity Calculation

We first show the ability of our proposed capacity calculation technique to accurately rank gateway placements from highest to lowest total capacity. We compare our calculation with measured throughput data from the operational TFA mesh network, in order to show that a better placement as per our capacity calculation is also a better placement as per measurements. TFA is a multi-tier mesh network providing Internet access in a densely populated, single-family residential, urban neighborhood with 18 deployed mesh nodes [3]. In the topology, two mesh nodes are connected if their link is on average usable at greater than 1 Mbps.

At the time of these experiments, the TFA network, featured two capacity points: gateway *GW-A* is a true wireline gateway and gateway *GW-B* is connected to *GW-A* via a directional link. We measure three different capacity point configurations: *GW-A* only, *GW-B* only, and both *GW-A* and *GW-B*. We observe the network during weekday peak hours for each of the three configurations, each measured on a subsequent weekday. For example, on one day, the directional link was disabled, making *GW-B* a non-gateway mesh node. The measured throughput is the peak rate (in Mbps) of data flow between the TFA network and the Internet over the observed time period. The traffic measured is the naturally occurring usage of the network. The throughput of the network with both gateways peaks at 2.2 Mbps. The *GW-A* configuration has a peak throughput of 1.46 Mbps and the *GW-B* configuration peaks at 610 Kbps. Using our technique in Section II, we calculate gateway-limited fair capacity of the *GW-A*-only configuration as 1.5 Mbps, the *GW-B*-only configuration as 1.35 Mbps, and both gateways together as 1.75 Mbps. Our technique predicts the correct ranking, with the *GW-A*-only configuration achieving 11% greater capacity than the *GW-B*-only configuration. The actual throughput values are lower than our capacity calculations due to several factors not included such as control overhead, MAC unfairness, and non-backlogged traffic sources.

B. Performance of Placement Algorithms

We now study the performance of MinHopCount and MinContention algorithms presented in Sections III-A and III-B on grid-based and realistic topologies. For all experiments, we consider an 802.11b system with the single-link wireless throughput assumed to be 6 Mbps. All mesh node locations are fixed and gateways can be installed on any mesh node. We compare the algorithms against a greedy placement strategy which repeatedly places the gateway that leads to the largest reduction in average path length.

1) Regular Grid Topology: We first examine the placement algorithms on a 7×7 regular grid topology. A mesh node communicates directly with at most 4 neighbors and contends with all two-hop neighbors. The only significant distinction between nodes are those that are on the borders. Figure 1 shows the performance of the greedy, MinHopCount, and MinContention algorithms as a function of how many new capacity points are added. For a network of this size, we also include the optimal placements found using brute force search.

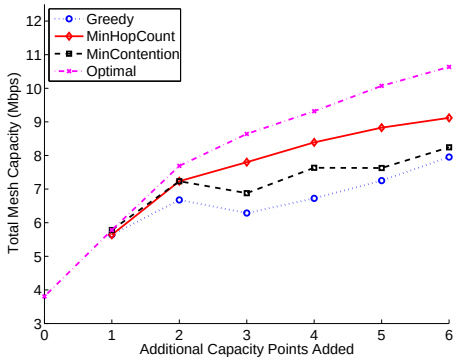

Fig. 1. Capacity of placements arrived at by algorithms on a square grid topology with 49 mesh nodes.

For adding between three and six gateways, the MinHopCount and MinContention algorithms find placements with capacities at least 86% and 77% of the optimal respectively, while the greedy placement is at least 72% of the optimal. The MinHopCount algorithm performs better in this regular topology because contention is uniform, leading to fewer or no situations where a greater hop count leads to better capacity. We also find that the each algorithm sometimes performs slightly worse with *more* gateways due to the fact that the hop count and contention metrics they use are only a first order approximation of the mesh capacity. Note

that, in this topology, the marginal benefit of each new gateway decreases due to the increasing level of contention between gateways. This effect is significant as gateways serve the most traffic and therefore cause more contention than other mesh nodes.

2) Real-World Topologies: We next consider our placement algorithms on the topologies of three currently deployed mesh networks: TFA, Chaska, and Google. These topologies present a new challenge in that the connectivity and contention matrices are no longer uniform for each mesh node. In these topologies, the local search algorithms have greater gain over greedy heuristics than in grid topologies because the irregular contention leads to situations where longer routes result in higher capacity. For each topology, we fix a number of already installed gateways and focus on upgrading with new gateways.

For the TFA topology, we are able to directly measure the signal strength between each pair of nodes. The topology is then a combination of this information with empirically measured communication and contention thresholds. For the Chaska and Google topologies, we must estimate the connectivity information with AP coordinates and manufacturer's information, introducing possible errors. While the true connectivity matrix is not observable externally, we assume a link exists if the physical distance is less than 200 meters.

The first deployed topology corresponds to the 195 node Chaska topology [1]. We begin with four known gateways and place additional capacity points in the network using the greedy, MinHopCount, and MinContention algorithms, plotting the results in Figure 2. Optimal does not appear here as the network size prohibits exhaustive search. The MinContention algorithm typically performs the best, up to 64% better than the greedy placement, because the local search improves upon previous choices it made and the algorithm considers the amount of contention caused by a path and not just the path length. For this topology, the MinHopCount performs up to 30% better than greedy, but its performance becomes similar to greedy beyond 15 gateways. We suspect that the MinHopCount's capacity estimates, $\mathbf{u}[i]$, degrade for more than 15 gateways for this topology, and it is the main reason that its performance becomes similar to greedy.

The second deployed topology considered in Figure 2 is the 447-node Google Mountain View network. In this network, we consider the current configuration of 59 gateways and use our algorithms to determine upgrade locations. MinContention outperforms greedy by up to 8%, though not with small budgets. Conversely, MinHopCount performs best with small budgets, but is approximately 10% lower capacity than greedy when considering larger budgets. This is a result of our simple capacity estimation strategy, which does not take into account contention between gateways. Note that our topology estimation results in a conservative and regular contention pattern in this topology.

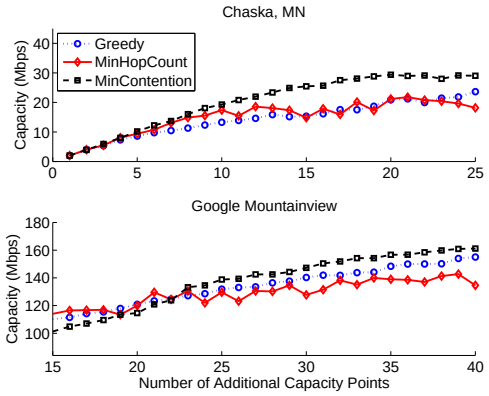

Fig. 2. Performance of placement algorithms on Chaska (top) and Google (bottom) topologies.

The third deployed topology we consider is the TFA network expansion, consisting of the currently deployed 18 nodes and 35 planned nodes. The current topology features two capacity points: one wired gateway and one directional antenna connection. Figure 3 presents the results of adding a small number of gateways to the projected TFA topology while holding fixed the current two gateways. Also included are the optimal values found via exhaustive search.

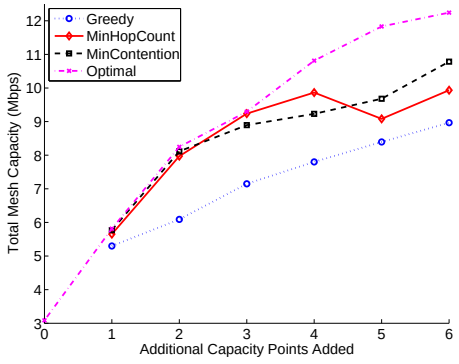

Fig. 3. Comparison of MinHopCount and MinContention algorithms with optimal placement for TFA-projected topology.

The local search algorithms closely approximate the optimal solution for the addition of up to 3 gateways; MinHopCount and MinContention solutions are within 97% and 96% of the optimal. As the budget increases, the solutions decline to as low as 80% of the optimal, with MinHopCount declining more. Greedy performs worst with small budgets, but improves as the marginal gain of additional gateways declines and allows the greedy to make up for early suboptimal choices. The MinHopCount algorithm performs worse with five gateways than with four due to the fact that iterative capacity

estimation does not directly take into account inter-gateway contention and hence MinHopCount does not perform as well when gateways contend with each other. In other words, it conservatively chooses longer paths so as to ensure that gateway capacity constraints are not violated.

In summary, we found that our local search algorithms significantly outperform a greedy algorithm, by up to 64%, and this gain is more pronounced on irregular topologies. For small budgets, the algorithms achieve very close ($\geq 97\%$) to optimal capacities and for larger budgets, MinContention performs best.

C. TFA Placements Case Study

We next study in greater detail the TFA network using an exhaustive study of all possible placements. We find that while the best placements have *similar* configurations, i.e. roughly the same gateway locations, there is a large capacity gap between near-optimal placements and the optimal. In other words, the optimal placement has significantly higher capacity than a near-optimal placement, demonstrating the need for good approximation algorithms. Further, the configuration of gateways in the optimal placement is similar to a near-optimal placement, indicating the applicability of local search operations for finding optimal placements. We consider the case of adding four additional capacity points in the projected TFA topology.

1) Distribution of Placement Quality: We present in Figure 4 a histogram of all possible ways to install four additional capacity points. Four candidate locations have been chosen based on availability of structures to mount antennas and we compare this manual placement with our algorithms. To understand the space of possible placements, Figure 4 is a histogram of the capacities resulting from all possible gateway placements, found via exhaustive search. The average placement results in a capacity of 7.7 Mbps with standard deviation of 1.2 Mbps and the optimal placement is 11.7 Mbps.

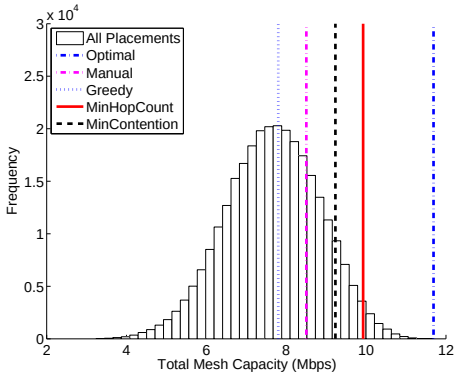

Fig. 4. Histogram of all placements of 4 new gateways in the projected TFA topology, along with the capacities found by our algorithms.

The difference in capacity between the mean placement and the optimal placement is a factor $1.7\times$, indicating the need for a good approximation algorithm. The MinHopCount and MinContention algorithms achieve 85% and 79% of the optimal configuration respectively, whereas the greedy placement achieves less than 60% of the optimal. Also, MinContention with $p = 1$ gives the same capacity as with $p = 4$ (maximum p possible in this case because $p \leq k$.

We also consider the percentage of *all possible* placements with higher capacity than the placement arrived at by our algorithms for this TFA scenario. MinHopCount achieves 85% of the optimal capacity, but only 0.1% of all possible placements result in higher capacity. For MinContention and greedy, approximately 1% and 15% of all placements result in higher capacity respectively. These results demonstrate the importance of a good approximation algorithm as there are large capacity gains due to finding better placements from among the top 0.1% of all placements.

2) Characterizing Similarity of Placements: In this section, we examine the characteristics of the solutions found by the local search algorithms in comparison to the optimal placement. We define a simple metric to capture the amount of similarity between any two gateway placements: the hop distance between the gateways in the two placements. The distance is calculated as the minimum hop cost to move the gateways in one placement to match the gateways in the other. This is equivalent to a transhipment problem where the demands are the capacities of the gateways in one placement, and the capacities correspond to the gateways in the second placement.

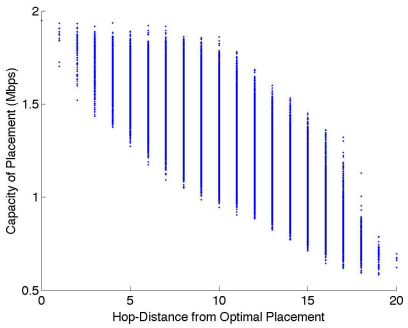

Fig. 5. Scatter plot for all possible gateway placements in the TFA-proj topology with 4 new gateways.

Figure 5 plots the ranges of capacities obtained when adding four additional gateways to the TFA network as a function of their distance from the optimal placement using this metric. We find that there is a strong correlation between distance from the optimal placement and the capacity with a correlation coefficient of -0.807. In other words, the higher capacity placements are most likely to be similar in configuration to the optimal

placement. A carefully designed local search algorithm can take advantage of this similarity to find the optimal placement. In the example of adding four gateways, we find that the distance between the second-best placement and the optimal placement is four hops. This means that the second-best placement is a minor perturbation of the best placement, and therefore the optimal placement can be found from the second-best placement with a local operation that moves gateways a combined total of four hops (not four hops per gateway). Both of our algorithms find placements within six hops from the optimal.

V. Related Work

A gateway placement algorithm using a greedy heuristic has been presented [4] to serve neighborhood networks, as well as a local search algorithm [14] for minimizing a combined cost and hop count metric. These approaches differ from ours in that we 1) incorporate wireless contention, 2) consider deployed city-wide mesh topologies, and 3) present two local search approximation algorithms, one of which has provable constant-factor approximation ratio. Others developed general techniques to calculate network capacities with interference [7] and incorporating multiple radios and channels [8]. These techniques require solving linear and mixed integer programs to find upper and lower capacity bounds. In contrast, we present a simple technique for exactly calculating gateway-limited fair capacity in polynomial time.

The capacity of hybrid wired and wireless networks has been studied in [10], though this study only provides asymptotic bounds and does not address the gateway placement problem. For regular topologies, [15] studies the impact of gateway density on network capacity and presents techniques to calculate connectivity to gateways.

We adapt our algorithms from solutions to the related capacitated facility location problem and uncapacitated k-median problem. Constant-factor approximations are known to exist for these problems using both local search [2], [13] and LP relaxation methods [9]. The algorithms we present can be improved with more sophisticated local search techniques [11], [16] that achieve better approximation ratios. For the closely related capacitated k-median problem, there is a constant-factor algorithm with up to $50\times$ violation of capacity constraints [6], making the algorithm too inaccurate for our purposes.

VI. Conclusions

We study the gateway placement problem, first introducing a technique to efficiently compute gateway-limited fair mesh capacity as a function of the contention at each gateway. We then present two gateway placement algorithms adapted from local search heuristics for related facility location problems with provable performance guarantees. The MinHopCount algorithm adapts a local search algorithm for the capacitated facility location problem and minimizes the average wireless hop count for all paths in the network, iteratively estimating the gateways' wireless capacities. The MinContention algorithm is adapted from a solution to the uncapacitated k-median problem and minimizes the average contention region size within a provable approximation ratio of $3 + \epsilon$. MinHopCount is more general and can handle non-uniform gateway costs, while MinContention is able to provide better performance guarantees. Our numerical results on three real topologies show that our algorithms outperform a greedy heuristic and achieve close to the optimal capacity. Further, we show that near-optimal solutions have similar gateway configuration as the optimal, but the difference in capacity is large, which supports the use of local search operations on near-optimal placements.

References

[1] Chaska.net and Tropos unwire Chaska, Minnesota. Tropos Networks White Paper, January 2007.

[2] Vijay Arya, Naveen Garg, Rohit Khandekar, Kamesh Munagala, and Vinayaka Pandit. Local search heuristic for k-median and facility location problems. In *Proceedings of ACM STOC*, 2001.

[3] Joseph Camp, Joshua Robinson, Chris Steger, and Edward Knightly. Measurement driven deployment of a two-tier urban mesh access network. In *Proceedings of ACM MobiSys*, Uppsala, Sweden, June 2006.

[4] Ranveer Chandra, Lili Qiu, Kamal Jain, and Mohammad Mahdian. Optimizing the placement of integration points in multi-hop wireless networks. In *Proceedings of ICNP*, Berlin, Germany, October 2004.

[5] Steven Cherry. Wi-Fi nodes to talk amongst themselves. *IEEE Spectrum*, July 2006.

[6] Julia Chuzhoy and Yuval Rabani. Approximating k-median with non-uniform capacities. In *Proceedings of SODA '05*, 2005.

[7] Kamal Jain, Jitendra Padhye, Venkat Padmanabhan, and Lili Qiu. Impact of interference on multi-hop wireless network performance. In *Proceedings of ACM MobiCom*, Sept. 2003.

[8] Murali Kodialam and Thyaga Nandagopal. Characterizing the capacity region in multi-radio multi-channel wireless mesh networks. In *Proceedings of ACM MobiCom*, Cologne, Germany, August 2005.

[9] Retsef Levi, David B. Shmoys, and Chaitanya Swamy. LP-based approximation algorithms for capacitated facility location. In *Proceedings of IPCO*, New York, NY, June 2004.

[10] Benyuan Liu, Zhen Liu, and Don Towsley. On the capacity of hybrid wireless networks. In *Proceedings of IEEE INFOCOM'03*, San Fransisco, CA, April 2003.

[11] Mohammad Mahdian and Martin Pal. Universal facility location. In *Proceedings of the 11th Annual European Symposium on Algorithms (ESA)*, 2003.

[12] Mohammad Mahdian, Yinyu Ye, and Jiawei Zhang. Improved approximation algorithms for metric facility location problems. In *5th International Workshop on Approximation Algorithms for Combinatorial Optimization*, 2002.

[13] Martin Pal, Eva Tardos, and Tom Wexler. Facility location with nonuniform hard capacities. In *IEEE Symposium on Foundations of Computer Science*, pages 329–338, 2001.

[14] Rajesh Prasad and Hongyi Wu. Minimum-cost gateway deployment in cellular wi-fi networks. In *Consumer Communications and Networking Conference*, Las Vegas, NV, January 2006.

[15] Joshua Robinson and Edward Knightly. A performance study of deployment factors in wireless mesh networks. In *Proceedings of IEEE INFOCOM 2007*, Anchorage, AK, May 2007.

[16] Jiawei Zhang, Bo Chen, and Yinyu Ye. Multi-exchange local search algorithm for the capacitated facility location problem. In *Proceedings of IPCO*, Berlin, 2004.