Minimum Interference Channel Assignment in Multiradio Wireless Mesh Networks

Anand Prabhu Subramanian, *Student Member*, *IEEE*, Himanshu Gupta, Samir R. Das, *Member*, *IEEE*, and Jing Cao, *Member*, *IEEE*

Abstract—In this paper, we consider multihop wireless mesh networks, where each router node is equipped with multiple radio interfaces, and multiple channels are available for communication. We address the problem of assigning channels to communication links in the network with the objective of minimizing the overall network interference. Since the number of radios on any node can be less than the number of available channels, the channel assignment must obey the constraint that the number of different channels assigned to the links incident on any node is at most the number of radio interfaces on that node. The above optimization problem is known to be NP-hard. We design centralized and distributed algorithms for the above channel assignment problem. To evaluate the quality of the solutions obtained by our algorithms, we develop a semidefinite program and a linear program formulation of our optimization problem to obtain lower bounds on overall network interference. Empirical evaluations on randomly generated network graphs show that our algorithms perform close to the above established lower bounds, with the difference diminishing rapidly with increase in number of radios. Also, *ns-2* simulations, as well as experimental studies on testbed, demonstrate the performance potential of our channel assignment algorithms in 802.11-based multiradio mesh networks.

Index Terms—Multi-radio wireless mesh networks, channel assignment, graph coloring, interference, mathematical programming.

1 Introduction

There is an increasing interest in using wireless mesh networks as broadband backbone networks to provide ubiquitous network connectivity in enterprises, campuses, and in metropolitan areas. An important design goal for wireless mesh networks is *capacity*. It is well known that wireless interference severely limits network capacity in multihop settings [1]. One common technique used to improve the overall network capacity is the use of multiple channels. Essentially, wireless interference can be minimized by using orthogonal (noninterfering) channels for neighboring wireless transmissions. The current IEEE 802.11 standard for WLANs (also used for mesh networks) indeed provides several orthogonal channels to facilitate the above. The presence of multiple channels requires us to address the problem of which channel to use for a particular transmission; the overall objective of such an assignment strategy is to minimize the overall network interference.

Dynamic channel assignment. One of the channel assignment approaches is to frequently change the channel on the interface; for instance, for each packet transmission based on the current state of the medium. Such *dynamic channel assignment* approaches [2], [3], [4], [5] require channel switching at a very fast time scale (per packet or a handful of packets). The fast-channel switching requirement makes these approaches unsuitable for use with commodity hardware, where channel switching delays itself can be in the order of milliseconds [6], which is an order of magnitude higher than typical packet transmission times (in microseconds). Some of the dynamic channel assignment approaches also require specialized MAC protocols or extensions of 802.11 MAC layer, making them further unsuitable for use with commodity 802.11 hardware.

Static or quasi-static channel assignment. Due to the difficulty of use of above dynamic approach with commodity hardware, there is a need to develop techniques that assign channels statically [7], [8], [9], [10], [11]. Such static assignments can be changed whenever there are significant changes to traffic load or network topology; however, such changes are infrequent enough that the channel-switching delay, and traffic measurement (see Section 2) overheads are inconsequential. We refer to the above as *quasi-static channel assignments*. If there is only one radio interface per router, then the above channel assignment schemes will have to assign the *same* channel to all radios/links in the network to preserve network connectivity. Thus, such assignment schemes require the use of multiple radio interfaces at each node. Due to board crosstalk or radio leakage [10], [12], commodity radios on a node may actually interfere even if they are tuned to different channels. However, this phenomena can be addressed by providing some amount of shielding or antenna separation [12] or increased channel separation (as is the case in 802.11a) [8].

Problem addressed. In our article, we address the problem of quasi-static assignment of channels to links in the context of networks with multiradio nodes. The objective of the channel assignment is to minimize the overall network interference. Channel assignment is done as some variation of a graph coloring problem, but it has an interesting twist in the context of mesh networks. The assignment of channels to links must obey the *interface*

- *A.P. Subramanian, H. Gupta, and S.R. Das are with the Computer Science Department, Stony Brook University, Stony Brook, NY 11794-4400. E-mail: {anandps, hgupta, samir}@cs.sunysb.edu.*
- *J. Cao is with Beihang University, 6863 Mailbox, Beijing 100083, P.R. China. E-mail: caojing@vrlab.buaa.edu.cn.*

Manuscript received 31 May 2007; revised 27 Feb. 2008; accepted 2 Apr. 2008; published online 25 Apr. 2008.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2007-05-0148. Digital Object Identifier no. 10.1109/TMC.2008.70.

constraint that the number of different channels assigned to the links incident on a node is at most the number of interfaces on that node. Different variations of this problem have been shown to be NP-hard [7], [9] before. Thus, efficient algorithms that run reasonably fast and provide good quality solutions are of interest. Since computing the optimal is intractable and approximation algorithms are still an open question, we take the approach of computing a *bound on the optimal* using mathematical programming approaches and develop heuristics that perform very close to the obtained bounds on the optimal.

Our contributions. For the above described channel assignment problem, we develop a centralized and a distributed algorithm. The centralized algorithm is based on a popular heuristic search technique called Tabu search [13] that has been used in the past in graph coloring problems. The distributed approach is motivated by the greedy approximation algorithm for the Max K-cut problem in graphs [14]. To evaluate their performances, we develop two mathematical programming formulations using semidefinite programming (SDP) and integer linear programming (ILP). We obtain *bounds* on the optimal solution by relaxing the ILP and SDP formulations to run in polynomial time. Finally, detailed ns-2 simulations and experimental study in an 11-node multiradio testbed demonstrate the full performance potential of the channel assignment algorithms in 802.11-based multiradio mesh networks.

The *salient features of our work* that set us apart from the existing channel assignment approaches on multiradio platforms are listed as follows:

- Our approach is "topology preserving," i.e., all links that can exist in a single channel network also exist in the multichannel network after channel assignment. Thus, our channel assignment does not have any impact on routing.
- Our approach is suitable for use with commodity 802.11-based networks without any specific systems support. We do not require fast channel switching or any form of MAC layer or scheduling support. While our algorithms indeed use interference and traffic models as input, such models can be gathered using experimental methods.
- Our work generalizes to nonorthogonal channels, including channels that are supposedly orthogonal but interfere because of crosstalk or leakage [12].
- Ours is the first work that establishes good lower bounds on the optimal network interference and demonstrates a good performance of the developed heuristics by comparing them with the lower bounds.

Paper organization. The rest of the paper is organized as follows: We start with describing the network model and the formulation of our problem in Section 2 and discuss related work in Section 3. We present our algorithms in Sections 4 and 5, respectively. In Section 6, we obtain lower bounds on the optimal network interference using semidefinite and linear programming. Section 7 presents generalizations of our techniques. We present simulation and experimental results in Sections 8 and 9, respectively.

2 Problem Formulation

In this section, we first present our network model and formulate of our channel assignment problem.

Network model. We consider a wireless mesh network with stationary wireless routers, where each router is equipped with a certain (not necessarily same) number of radio interfaces. We model the *communication graph* of the network as a general undirected graph over the set of network nodes (routers). An edge (i, j) in the communication graph is referred to as a *communication link* or *link* and signifies that the nodes i and j can communicate with each other as long as both the nodes have a radio interface each with a common channel. There are a certain number of channels available in the network. For clarity of presentation, we assume for now that the channels are orthogonal (noninterfering) and extend our techniques for nonorthogonal channels in Section 7.

Interference model. Due to the broadcast nature of the wireless links, transmission along a communication link (between a pair of wireless nodes) may interfere with transmissions along other communication links in the network. Two interfering links cannot engage in successful transmission at the same time if they transmit on the same channel. The *interference model* defines the set of links that can interfere with any given link in the network. There have been various interference models proposed in the literature, for example, the physical and protocol interference models [1], [15]. The discussion in this paper is independent of the specific interference model used as long as the interference model is defined on pairs of communication links.

For clarity of presentation, we assume a *binary interference model* for now (i.e., two links either interfere or do not interfere) and generalize our techniques to fractional interference in Section 7. Moreover, in our approach of quasi-static channel assignment, the level of interference between two links actually depends on the traffic on the links. However, for clarity of presentation, we assume uniform traffic on all links for now and generalize our techniques to nonuniform traffic in Section 7.

Conflict graph. Given an interference model, the set of pairs of communication links that interfere with each other (assuming them to be on the same channel) can be represented using a *conflict graph* [15]. To define a conflict graph, we first create a set of vertices V_c corresponding to the communication links in the network. In particular

$$V_c = \{l_{ij} | (i, j) \text{ is a communication link}\}.$$

Now, the conflict graph $G_c(V_c, E_c)$ is defined over the set V_c as vertices, and a *conflict edge* (l_{ij}, l_{ab}) in the conflict graph is used to signify that the communication links (i, j) and (a, b) interfere with each other if they are on the same channel. The above concept of a conflict graph can be used to represent any interference model. As defined above, the conflict graph does not change with the assignment of channels to vertices in the conflict graph.

We illustrate the concept of conflict graph in Fig. 1. The wireless network represented in Fig. 1 has five network nodes $A, B, \ldots, E$ and four communication links, as shown in the communication graph (see Fig. 1a). The conflict graph (see Fig. 1b) has four nodes each representing a communication

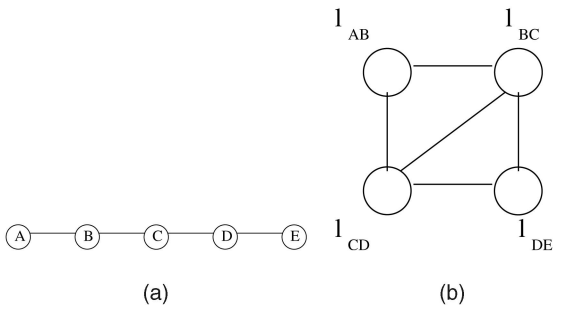

Fig. 1. Communication graph and corresponding conflict graph. (a) Communication graph. (b) Conflict graph.

link in the network. In this figure, we assume an 802.11 like interference model, where the transmission range and interference range are equal. When the RTS/CTS control messages are used links within two hops interfere. Thus, the communication link (A, B) interferes with the communication links (B, C) and (C, D) and not with (D, E).

Notations. Here, we introduce some notations that we use throughout this paper:

- N is the set of nodes in the network.
- R_i is the number of radio interfaces on node $i \in N$.
- $\mathcal{K} = \{1, 2, \ldots, K\}$ is the set of K channels.
- $V_c = \{l_{ij} | (i, j) \text{ is a communication link}\}$.
- $G_c(V_c, E_c)$ is the conflict graph of the network.
- For $i \in N$, $E(i) = \{l_{ij} \in V_c\}$, i.e., $E(i)$ is set of vertices in V_c that represent the communication links incident on node i.

In addition, throughout this paper, we use variables u, v to refer to vertices in V_c, variables i, j, a, and b to refer to nodes in N, and the variable k to refer to a channel. Since assigning channel can be thought of as coloring vertices, we use the terms channel and colors interchangeably throughout our paper.

Channel assignment problem. The problem of channel assignment in a multiradio wireless mesh network can be informally described as follows: Given a mesh network of router nodes with multiple radio interfaces, we wish to assign a unique channel to each communication link[1] in the network such that the number of different channels assigned to the links incident on any node is at most the number of radios on that node. Since we assume uniform traffic on all links for now, we assign channels to all links and define the *total network interference* as the number of pairs of communication links that are interfering (i.e., are assigned the same channel and are connected by an edge in the conflict graph). The objective of our problem is to minimize the above-defined total network interference, as it results in improving the overall network capacity [1].

A note is in order regarding our choice of optimization objective. While a natural objective for the channel assignment problem would be to directly maximize the overall network throughput, it turns out that modeling network throughput analytically in a random access-based medium access model is hard. The previous works in the literature [15], [16], [17], [18] that maximize overall network throughput, assume a time-slotted synchronized medium access model with scheduling as one part of the problem. In a time-slotted synchronized medium access model, modeling throughput is much easier relative to a CSMA-based random access model. There indeed has been several attempts in recent literature to model link capacity [19], [20] in a CSMA-based network using measurements from real 802.11-based networks. However, these models are quite complex, and it is difficult to use them as objective functions for an optimization problem and, at the same time, develop efficient solution approaches. In our work, our interest is in developing solutions for use with commodity systems based on 802.11. Thus, in our channel assignment problem, we use an objective function than can be formally defined using the conflict graph model and optimized efficiently. With this argument, network interference is a more practical choice relative to network throughput.

Consider a wireless mesh network over a set N of network nodes. Formally, the *channel assignment problem* is to compute a function $f : V_c \to \mathcal{K}$ to minimize the *overall network interference* $I(f)$ defined below while satisfying the below *interface constraint*.

Interface constraint.

$$\forall i \in N, \quad |\{k \mid f(e) = k \text{ for some } e \in E(i)\}| \leq R_i.$$

Network interference $I(f)$.

$$I(f) = |\{(u, v) \in E_c \mid f(u) = f(v)\}|. \tag{1}$$

If we look at assignment of channels to vertices as coloring of vertices, then the network interference is just the number of monochromatic edges in the vertex-colored conflict graph. The channel assignment problem is NP-hard since it reduces to Max K-cut (as discussed below).

Input parameters—measuring interference and traffic. Note that under the simplified assumption of uniform traffic, the only input to our channel assignment problem is the network conflict graph. The conflict graph (along with the edge weights for fractional interference, see Section 7) can be computed using methods similar to recently reported measurement-based techniques in [21] and [20]. These techniques are localized due to the localized nature of interference and, hence, can be easily run in a distributed manner. Also, in most cases (for static network topologies), the above measurements need to be done only one time. For the case of nonuniform traffic, we need to measure the average (over the time scale of channel assignment) traffic (i.e., the function $t(.)$ in Section 7) on each link. Such traffic measurements can be easily done using existing software tools (e.g., COMO [22]).

Relationship with Max K-cut. Given a graph G, the Max K-cut problem [14] is to partition the vertices of G into K partitions in order to maximize the number of edges whose endpoints lie in *different* partitions. In our channel assignment problem, if we view vertices of the conflict graph assigned to a particular channel as belonging to one partition, then the network interference is actually the number of edges in the conflict graph that have endpoints in *same* partition. Thus, our channel assignment problem is

1. Note that merely assigning channels to radios is not sufficient to measure network interference/capacity, since a link still can use one of many channels for transmission.

basically the Max K-cut problem with the added interface constraint. Since Max K-cut is known to be NP-hard, our channel assignment problem is also NP-hard.

3 Related Work

The use of multiple channels to increase capacity in a multihop network has been addressed extensively. Generally, there have been two types of approaches: 1) fast switching of channels (possibly, on a per-packet basis) on a single radio or 2) assigning channels to radios for an extended period of time in a multiradio setting.

Fast switching of channels. In an MMAC protocol [3], the authors augment the 802.11 MAC protocol such that the nodes meet at a common channel periodically to negotiate the channels to use for transmission in the next phase. In SSCH [4], the authors propose dynamic switching of channels using pseudorandom sequences. The idea is to randomly switch channels such that the neighboring nodes meet periodically at a common channel to communicate. In DCA [2], the authors use two radios—one for the control packets (RTS/CTS packets) and another for data packets. The channel to send the data packet is negotiated using the control packets, and the data packets are sent in the negotiated channels. In AMCP [5], the authors uses a similar notion of a control channel, but a single radio and focus on starvation mitigation. In [23], the authors use a channel assignment approach using a routing protocol and then use these channels to transmit data. For coordination, control channels are used. In [24], two radio and single radio multichannel protocols are proposed, but separate control channels are not needed.

All the above protocols require a small channel switching delay (of the order of hundred microseconds or less), since channels are switched at a fast time scale (possibly, on a per-packet basis). However, the commodity 802.11 wireless cards incur a channel switching delay of the order of milliseconds (based on our observations), as channel switching requires a firmware reset and execution of an associated procedure. Similar experiences were reported in [6], and in particular, it has been shown in [4] and [25] that packet-based channel assignment may not be feasible in a practical setting [26]. In addition, the above approaches require changes to the MAC layer. Thus, the above approaches are not suitable with currently available commodity hardware.

Static/quasi-static channel assignment in multiradio networks. There have been many works that circumvent fast channel switching by assigning channels at a much larger time scale in a multiradio setting. This solution is deemed more practical as there is neither a need to modify the 802.11 protocol or need for interfaces with very low channel switching latency.

In particular, Raniwala and Chiueh [8] assume a tree-based communication pattern to ease coordination for optimizing channel assignment. Similar tree-based communication patterns have been used in [27]. The above schemes do not quantify the performance of their solutions with respect to the optimal. In addition, Tang et al. [11] consider minimum-interference channel assignments that preserve k-connectivity. None of the above schemes preserve the original network topology and, hence, may lead to inefficient assignments and routing in a more general peer-to-peer communication.

Topology preserving schemes. To facilitate independent routing protocols, our work focuses on developing quasi-static channel assignment strategies that preserve the original network topology. Prior works on topology preserving channel assignment strategies are given as follows: Adya et al. [10] propose a strategy wherein they assume a hard-coded assignment of channels to interfaces and then determine which channel/interface to use for communication via a measurement-based approach. They do not discuss how the channels are assigned to interfaces. In [7], Raniwala et al. propose a centralized load-aware channel assignment algorithm; however, they require that source-destination pairs with associated traffic demands and routing paths be known a priori. In [28], Das et al. present a couple of optimization models for the static channel assignment problem in a multiradio mesh network. However, they do not present any practical (polynomial time) algorithm. In [29], the authors propose a linear optimization model that assign channels to interfaces, and then, assign interfaces to neighbors so that neighbors having interfaces with common channels can communicate. In contrast, in our model, we assign channels to links directly. In [30], a purely measurement-based approach is taken for channel assignment to radios (instead of links). Here, one radio at each node is tuned to a common channel to preserve the original topology; however, this can be wasteful when only a few interfaces are available. Moreover, assignment of channels to radios still leaves the problem of which channel to use for a transmission/link. In [31], the authors propose a simple greedy algorithm for channel assignment in multiradio networks. They assume a binary interference model and do not show any performance bounds. In the closest related work to ours, Marina and Das in [9] address the channel assignment to communication links in a network with multiple radios per node. They propose a centralized heuristic for minimizing the network interference. We compare the performance of our proposed algorithm with this heuristic and show a significant improvement.

Other related work. In other related work, Kyasanur and Vaidya [32] propose a hybrid channel assignment strategy: some interfaces on a node have a fixed assignment, and the rest can switch channels as needed. To put things in perspective, our work presents algorithms for making these fixed assignments. The authors in [15], [16], [17], and [18] address joint channel assignment, routing, and scheduling problems. These papers make an assumption of synchronized time-slotted channel model as scheduling is integrated in their methods. This makes modeling network throughput straightforward and consideration of a joint channel assignment and routing problem practical. However, the synchronized time-slotted model is hard to implement in commodity radios that use 802.11, as in 802.11 scheduling is done following a CSMA-based random access paradigm. In addition, these works often make impractical assumptions. For example, Jain et al.'s [15] approach requires an enumeration of all maximal sets of noninterfering links (independent sets), and Kumar et al. [16] considers networks with bounded "interference degrees."

In remaining related works, Kyasanur and Vaidya [33] derives upper bounds on the capacity of wireless multihop networks with multiple channels, and [26] investigate granularity of channel assignment decisions by assigning

channels at the level of components (links, paths, or general graph component) in single radio networks.

On the theoretical front, the related Max K-cut problem has been studied extensively. In particular, Frieze and Jerrum [14] gives a constant approximation algorithm using semidefinite algorithm for general graphs, while [34] consider uniformly random $G_{n,p}$ graphs and give an approximation scheme. As a hardness result, Kann et al. [35] proves that unless $\mathrm{P} = \mathrm{NP}$, the Max K-cut problem cannot be approximated within a factor of $1 - \frac{1}{34K}$.

4 Centralized Tabu-Based Algorithm

In this section, we describe one of our algorithms for the channel assignment problem based on the Tabu search [13] technique for coloring vertices in graphs. Our Tabu-based algorithm is centralized. Centralized algorithms are quite practical in "managed" mesh networks, where there is already a central entity. Moreover, they are amenable to a higher degree of optimization, easier to upgrade, and use of "thin" clients. Centralized approaches have indeed been proposed in various recent works [9], [7], [11] and have also become prevalent in the industry (e.g., WLAN and mesh products from Meru Networks [36] and Tropos [37]).

Algorithm overview. Recall that our channel assignment problem is to color the vertices V_c of the conflict graph G_c using K colors while maintaining the interface constraint and minimizing the number of monochromatic edges in the conflict graph. In other words, the channel assignment problem is to find a solution/function $f : V_c \rightarrow \mathcal{K}$ with minimum network interference $I(f)$ such that f satisfies the interference constraint. Our Tabu-based algorithm consists of two phases. In the first phase, we use Tabu search based technique [13] to find a good solution f without worrying about the interface constraint. In the second phase, we remove interface constraint violations to get a feasible channel assignment function f.

First phase. In the first phase, we start with a random initial solution f_0 wherein each vertex in V_c is assigned to a random color in $\mathcal{K}$. Starting from such a random solution f_0, we create a sequence of solutions $f_0, f_1, f_2, \ldots, f_j, \ldots,$ in an attempt to reach a solution with minimum network interference. In the jth iteration $(j \geq 0)$ of this phase, we create the next solution f_{j+1} in the sequence (from f_j) as follows:

The jth iteration. Given a solution f_j, we create f_{j+1} as follows: First, we generate a certain number (say, r) of random neighboring solutions of f_j. A random neighboring solution of f_j is generated by picking a random vertex u and reassigning it to a random color in $(\mathcal{K} - \{f_j(u)\})$. Thus, a neighboring solution of f_j differs from f_j in the color assignment of only one vertex. Among the set of such randomly generated neighboring solutions of f_j, we pick the neighboring solution with the lowest network interference as the next solution f_{j+1}. Note that we do not require $I(f_{j+1})$ to be less than $I(f_j)$, so as to allow escaping from local minima.

Tabu list. To achieve fast convergence, we avoid reassigning the same color to a vertex more than once by maintaining a *Tabu list* τ of limited size. In particular, if f_{j+1} was created from f_j by assigning a new color to a vertex u, then we add $(u, f_j(u))$ to the tabu list τ. Now, when generating random neighboring solutions, we ignore neighboring solutions that assign the color k to u if (u, k) is in τ.

Termination. We keep track of the best (i.e., with the lowest interference) solution f_{best} seen so far by the algorithm. The first phase terminates when the maximum number (say, i_{max}) of allowed iterations have passed without any improvement in $I(f_{best})$. In our simulations, we set i_{max} to $|V_c|$. Since network interference $I(f)$ takes integral values and is at most $(|V_c|)^2$, the value $I(f_{best})$ is guaranteed to decrease by at least 1 in $i_{max} = |V_c|$ iterations (or else, the first phase terminates). Thus, the time complexity of the first phase is bounded by $O(rd|V_c|^3)$, since each iteration can be completed in $O(rd)$ time, where r is the number of random neighboring functions generated, and d is the maximum degree of a vertex in the conflict graph. Note that network interference of a neighboring solution can be computed in $O(d)$ time. A formal description of the first phase is shown in Algorithm 1.

Algorithm 1: First phase of Tabu-based algorithm.
Input: Conflict Graph $G_c(V_c, E_c)$; Set of channels $\mathcal{K}$.
Output: Channel Assignment Function $f_{best} : V_c \rightarrow \mathcal{K}$.
Start with a random assignment function f_0;
$f_{best} = f_0$; $I_{best} = I(f_0)$; $\tau = null$; $j = 0$; $i = 0$;
while $I(f_i) > 0$ and $i \leq i_{max}$ **do**
Generate r random neighbors of f_j;
Each neighbor is generated by randomly picking a u in V_c and $k \in \mathcal{K}$ such that $k \neq f_j(u)$ and $(u, k) \notin \tau$ and changing $f_j(u)$ to k
Let f_{j+1} be the neighbor with lowest interference.
Add $(u, f_j(u))$ to τ.
If τ is full, delete its oldest entry;
If $(I(f_{j+1}) < I_{best})$
then $I_{best} = I(f_{j+1})$; $f_{best} = f_{j+1}$; $i = 0$;
else $i = i + 1$;
end if;
$j = j + 1$;
end while
RETURN f_{best};

Second phase. Note that the solution f returned by the first phase may violate interface constraints. Thus, in the second phase, we eliminate the interface constraints by repeated application of the following "merge" procedure. Given a channel/color assignment solution f, we pick a network node for the merge operation as follows: Among all the network nodes wherein the interface constraint is violated, i.e., whose number of radios is less than the number of distinct colors assigned to the incident communication links, we pick the node wherein the difference between the above two terms is the maximum. Let i be the node picked as above for the merge operation. We reduce the number of colors incident on i by picking (as described later) two colors k_1 and k_2 incident on i and changing the color of all k_1-colored links to k_2. In order to ensure that such a change does not create interface constraint violations at other nodes, we *iteratively* "propagate" such a change to all k_1-colored links that are "connected" to the links whose color has been just changed from k_1 to k_2. Here, two links are said to be connected if they are incident on a common node. Essentially, the above propagation of color-change ensures that for any node j, either *all or none* of the k_1-colored links incident on j are changed to color k_2, see Fig. 2. The completion of the above described color-change propagation marks the completion of *one* merge procedure.

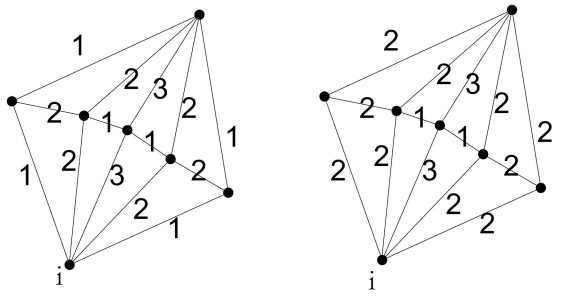

Fig. 2. Merge operation of second phase. The two figures are the communication graphs of the network before and after the merge operation. Labels on the links denote the color/channel. Here, the merge operation is started at node i by changing all its 1-colored links to color 2.

The above described merge procedure reduce the number of distinct colors incident on i by one and does not increase the number of distinct colors incident on any other node (due to the all or none property). Thus, repeated application of such a merge operation is guaranteed to resolve all interface constraints. Note that a merge operation probably will result in increase in network interference. Thus, for a given node i, we pick those two color k_1 and k_2 for the merge operation that cause the least increase in the network interference due to the complete merge operation.

5 Distributed Greedy Algorithm (DGA)

In this section, we describe our DGA for the channel assignment problem. Our choice of greedy approach is motivated by the following two observations.

Max K-cut problem in random graphs. As described before, the Max K-cut problem on a given graph G is to partition the vertices of G into K disjoint subsets such that the sum of number of edges with endpoints in different partitions is maximized. In [34], the authors consider $G_{n,p}$ graphs that are defined as random graph over n vertices, where each edge exists with a uniform probability of p. The authors design an algorithm with an approximation ratio $1 - \frac{1}{Kx}$ (where $x \geq 1$) for the Max K-cut problem in such $G_{n,p}$ graphs. In particular, they obtain a lower bound on the size of the Max K-cut in $G_{n,p}$ graphs problem using a simple greedy heuristic and obtain an upper bound using a relaxed SDP given in [14]. They show that the lower and upper bounds are close with very high probability. In effect, the authors show that the greedy heuristic delivers a $1 - \frac{1}{Kx}$ factor approximation solution with very high probability. The greedy heuristic proposed in [34] for Max K-cut works by deciding the partition of one vertex at a time in a greedy manner (i.e., place the vertex in the partition that results in maximizing the number of edges with endpoints in different partitions).

Conflict Graph is $G_{n,p}$. It can be shown that a network formed by randomly placed nodes in a fixed region generates a random conflict graph G_c, which is also $G_{n,p}$. Here, we assume an interference model wherein two communication links (u, v) and (r, s) interfere with each other depending on the locations of the nodes u, v, r, and s (as is the case with protocol interference model [1]). Now, the vertices $l_{u,v}$, $l_{r,s} \in V_c$ representing the communication links (u, v) and (r, s) are connected in G_c if and only if the communication links (u, v) and (r, s) interfere with each other. Thus, the probability of an edge between two vertices of V_c depending only on the locations of the involved network nodes, and since the network nodes are randomly placed, the probability of an edge between two vertices in V_c is uniform.

The above observations motivate use of a greedy approach for our channel assignment problem.

Centralized greedy algorithm. We start with presenting the centralized version, which yields a natural distributed implementation. In the initialization phase of our greedy approach, each vertex of V_c is colored with the color 1. Then, in each iteration of the algorithm, we try to change the color of some vertex in a greedy manner without violating the interface constraint. This strategy is different from the Tabu-based algorithm, where we resolve interface constraint violations in the second phase while not worrying about introducing them in the first phase. In each iteration of the greedy approach, we try to change the color of some vertex $u \in V_c$ to a color k. We look at all possible pairs of u and k, considering only those that do not result in the violation of any interface constraint and pick the pair (u, k) that results in the largest decrease in network interference. The algorithm iterates over the above process until there is no pair of u and k that decreases the network interference any further. Note that a vertex in V_c may be picked multiple times in different iterations. However, we are guaranteed to terminate because each iteration monotonically decreases the network interference. In particular, as noted in the previous section, since the network interference takes integral values and is at most $(|V_c|)^2$, the number of iterations of the greedy algorithm is bounded by $(|V_c|)^2$. Since each iteration can be completed in $O(dK|V_c|)$, where K is the total number of colors, and d is the maximum degree of a vertex in the conflict graph, the total time complexity of the greedy algorithm is $O(dK|V_c|^3)$. The pseudocode for the centralized version of the greedy algorithm is shown in Algorithm 2.

Algorithm 2. Centralized greedy algorithm.
Input: Conflict Graph $G_c(V_c, E_c)$; Set of channels $\mathcal{K}$.
Output: Channel Assignment Function $f : V_c \rightarrow \mathcal{K}$.
Initialization:
$f(u) = 1$, $\forall u \in V_c$
Repeat
(1) Choose the pair $(u, k) \in (V_c \times \mathcal{K})$, such that when $f(u)$ is assigned to k, the interference constraint is not violated and the total network interference $(I(f))$ decreases the most
(2) Set $f(u) = k$
Until $I(f)$ cannot be decreased any further.

DGA. The above-described greedy approach can also be easily distributed by using a localized greedy strategy. The distributed implementation differs from the centralized implementation in the following aspects. First, in the distributed setting, multiple link-color pairs may be picked simultaneously across the network by different nodes. Second, the decision of which pair is picked is based on the local information. Last, to guarantee termination in a distributed setting, we impose additional restriction that each pair (u, k) is picked at most once (i.e., each vertex $u \in V_c$ is assigned a particular color k at most once) in the entire duration of the algorithm.

In the distributed implementation, each vertex $u = l_{ij} \in V_c$ corresponding to the link (i, j) is *owned* by i or j, whichever has the higher node ID. This is done to ensure consistency of color information across the network. Initially, each vertex in V_c is assumed to colored 1. Let $m \geq 1$ be the parameter defining the local neighborhood of a node. Based on the information available about the colors of links in the m-hop neighborhood of i, each network node i selects (after waiting for a certain random delay) a (u, k) combination such that we have the following:

1. $u = l_{ij}$ is owned by i.
2. Changing the color of u to k does not violate the interface constraint at node i or j.
3. The pair (u, k) has not been selected before by i.
4. The pair (u, k) results in the largest decrease in the "local" network interference.

Then, the node i sends a `ColorRequest` message to node j. The node j responds with the `ColorReply` message, if and only if changing the color of u to k still does not violate the interface constraint at node j. On responding with the `ColorReply` message, the node j *assumes*[2] that the color of u has been changed to k. On receiving the `ColorReply` message for j, the node i sends a `ColorUpdate`(u, k) message to all its m-hop neighbors. If a `ColorReply` message is not received within a certain time period, the node i abandons the choice of (u, k) for now and starts a fresh iteration. Since each pair (u, k) is picked at most once, then the total number of iterations (over all nodes) in the above algorithm is at most $O(|V_c|K)$. The pseudocode for the DGA that runs in every node $i \in V$ is shown in Algorithm 3.

The above DGA is localized and can be made to work in dynamic topologies. Our simulation results showed that the above distributed algorithm performs almost same as the centralized version due to the localized nature of the network interference objective function. The input network parameters of traffic and interference are measured as discussed in Section 2.

Algorithm 3. DGA for each node $i \in V$.

Input: "Local" network and conflict graph; set of channels $\mathcal{K}$.

Output: Channel Assignment (i.e., $f(u)$) for all links $u \in V_c$ incident on node i.

Repeat

Among all pairs (u, k), where $u \in V_c$ is owned by i and $k \in \mathcal{K}$

that is not already chosen and does not violate interface constraint at i

choose the one that produces largest decrease in local interference.

Send `ColorRequest`(u, k) to node j, where $u = (i, j)$.

Wait for `ColorReply`(u, k) message from node j.

If `ColorReply`(u, k) message is not received within a certain time

Abandon the choice (u, k).

Until Local interference cannot be decreased any further, or all (u, k) combinations have already been chosen.

When `ColorRequest`(u, k) message is received from node j, where $u = (i, j)$:

If assigning channel k to link u does not cause interface constraint violation

Send `ColorReply`(u, k) message to node j.

When `ColorReply`(u, k) message is received from node j:

Set $f(u) = k$ and send `ColorUpdate`(u, k) message to "local" neighborhood

When `ColorUpdate`(u, k) message is received:

Update locally maintained channel assignment of links in the local network graph.

2. Such an assumption may need to be later corrected through communication with i if the `ColorUpdate`(u, k) message is not received from i within a certain amount of time.

6 Bounds on Optimal Network Interference

In this section, we derive lower bounds on the minimum network interference using semidefinite and linear programming approaches. These lower bounds will aid in understanding the quality of the solutions obtained from the algorithms presented in previous sections.

6.1 Semidefinite Programming Formulation

In this section, we model our channel assignment problem in terms of a SDP.

SDPs. SDP [38] is a technique to optimize a linear function of a symmetric positive-semidefinite matrix[3] subject to linear equality constraints. SDP is a special case of convex programming [39], since a set of positive semidefinite matrices constitutes a convex cone. SDPs can be solved in polynomial time using various techniques [40]. The reader is referred to [38] and [41] for further details on SDP and its application to combinatorial optimization. The standard form of SDP is given as follows:

$$\begin{aligned} \text{Minimize} \quad & \mathrm{C.X} \\ \text{such that} \quad & \mathrm{A_i.X} = b_i, \qquad 1 \leq i \leq m, \text{and} \\ & X \succeq 0, \end{aligned}$$

where C, A_i $(\forall i)$, and X are all symmetric $n \times n$ matrices, and b_i is a scalar vector. The constraint $X \succeq 0$ implies that the variable (to be computed) matrix X must lie in the closed convex cone of a positive semidefinite matrix. Also, the . (dot) operation refers to the standard inner product of two symmetric matrices.

As mentioned in Section 2, our channel assignment problem is essentially the Max K-cut problem in the conflict graph with the additional interface constraint. Below, we start with presenting the SDP for the Max K-cut problem in [14]. We then extend it to our channel assignment problem by adding the interface constraint.

SDP for Max K-cut. Let y_u be a variable that represent the color of a vertex $u \in V_c$. Instead of allowing y_u to take 1 to K integer values, we define y_u to be a vector in

3. A matrix is said to be *positive semidefinite* if all its eigenvalues are nonnegative.

$\{a_1, a_2, \ldots, a_K\}$, where the a_i vectors are defined as follows [14]: We take an equilateral simplex Σ_K in $\mathbf{R}^{K-1}$ with vertices $b_1, b_2, \ldots, b_K$. Let $c_K = \frac{(b_1+b_2+\ldots+b_K)}{K}$ be the centroid of Σ_K, and let $a_i = b_i - c_K$ for $1 \leq i \leq K$. Also, assume $|a_i| = 1$ for $1 \leq i \leq K$. Now, the Max K-cut problem can be formulated as an integer quadratic program as follows [14]:

$\mathbf{IP_{Max-K}}$:

$$\text{Maximize} \quad \frac{K-1}{K} \sum_{(u,v)\in E_c} (1 - y_u.y_v)$$

$$\text{such that} \quad y_u \in \{a_1, a_2, \ldots, a_K\}.$$

Note that since $a_i.a_j = \frac{-1}{K-1}$ for $i \neq j$, we have

$$1 - y_u.y_v = \begin{cases} 0, & \text{if } y_u = y_v, \\ \frac{K}{K-1}, & \text{if } y_u \neq y_v. \end{cases}$$

Interface constraint. We now add the interface constraint to the above formulation for Max K-cut. For each $i \in N$, let

$$\Phi_i = \sigma(E(i), R_i) - \left(\binom{|E(i)|}{2} - \sigma(E(i), R_i) \right) / (K-1),$$

where $\sigma(E(i), R_i)$ is as defined as follows:

$$\sigma(S, K) = \frac{\beta\alpha(\alpha+1) + (K-\beta)\alpha(\alpha-1)}{2}, \tag{2}$$

where $\alpha = \lfloor \frac{|S|}{K} \rfloor$ and $\beta = |S| \ mod \ K$. It can be shown [42] that the number of monochromatic edges in the clique of size $|S|$ when colored by K colors is at least $\sigma(S, K)$. Now, we add the following constraint to represent the interface constraint:

$$\sum_{u,v\in E(i)} y_u.y_v \geq \Phi_i \quad \forall i \in N. \tag{3}$$

Recall that vertices in $E(i)$ form a clique in the conflict graph and cannot be partitioned into more than R_i partitions to satisfy our interface constraint. Now, $\sigma(E(i), R_i))$ gives a lower bound on the number of monochromatic edges in this clique $(E(i))$ [42], and thus, $\binom{|E(i)|}{2} - \sigma(E(i), R_i))$ is an upper bound on the number of nonmonochromatic edges. Since we know that $y_u.y_v = 1$ for any monochromatic edge (u, v) and $y_u.y_v = \frac{-1}{K-1}$ for any nonmonochromatic edge, we have constraint in (3).

Note that even though (3) is a valid constraint, it does not necessarily restrict the number of colors assigned to vertices of $E(i)$ to R_i. Thus, the IP_{Max-K} augmented by (3) only gives an upper bound on the number of nonmonochromatic edges.

Relaxed SDP for channel assignment. Since we cannot solve the integer quadratic program IP_{Max-K} for problems of reasonable size, we relax it by allowing the variables y_u to take any unit vector in $R^{|V_c|}$. Since $y_u.y_v$ can now take any value between 1 and -1, we add an additional constraint to restrict $y_u.y_v$ to be greater than $\frac{-1}{K-1}$. The relaxed SDP for the channel assignment is given as follows:

$$\text{Maximize} \ \frac{K-1}{K} \sum_{(u,v)\in E_c} (1 - y_u.y_v)$$

$$\text{such that}$$

$$y_u \in R^{|V_c|} \text{ and } |y_u| = 1$$

$$y_u.y_v \geq \frac{-1}{K-1}, \quad \forall u \neq v, \text{ and}$$

$$\sum_{u,v\in E(i)} y_u.y_v \geq \Phi_i, \quad \forall i \in N.$$

Standard SDP formulation. Now, we convert the above relaxed version into the standard SDP formulation. Let W be the $|V_c| \times |V_c|$ symmetric matrix representing the adjacency matrix of the graph G_c, and let e be the $|V_c| \times 1$ vector containing all 1s. Now, let $L = d(W.e) - W$ denote the Laplacian of the W matrix, where $d(W.e)$ is the $|V_c| \times |V_c|$ matrix with $W.e$ as the main diagonal. Finally, let

$$C = -\frac{L(K-1)}{2K},$$

X be the semidefinite $|V_c| \times |V_c|$ matrix representing $y_u.y_v$ for all $u, v \in V_c$. Now, the SDP for the channel assignment problem in the standard SDP form (Matrix Notation) [34] can be represented as follows:

$$\text{Minimize} \quad C.X$$

$$\text{such that}$$

$$diagonal(X) = e$$

$$X_{u,v} \geq \frac{-1}{K-1}, \quad \forall u \neq v \in V_c,$$

$$A_i.X \geq 2\Phi_i, \quad \forall i \in N, \text{ and}$$

$$X \succeq 0,$$

where each $A_i (i \in V)$ is a $|V_c| \times |V_c|$ matrix representing $E(i)$. In particular, the $A_i[u, v] = 1$ if $(u, v) \in E_i$, and 0 otherwise. Also, the inequalities in the above constraints can be converted into equalities by subtracting linear positive variables from the left-hand side.

The solution to the above SDP gives an upper bounds on the number of nonmonochromatic edges, and the lower bound on the optimal network interference can be obtained by subtracting it from $|E_c|$. This SDP can solved using standard SDP solver such as DSDP 5.0 [43].

6.2 Linear Programming Formulation

In our simulations, we observed that solving the SDP formulation presented in the previous section can take a long time (12 hours on a 2.4-GHz Intel Xeon machine with 2 Gbytes of RAM for a 50-node network) and memory and, hence, may not be feasible for very large network sizes. Thus, in this section, we formulate our channel assignment problem as an ILP and use the relaxed linear program with additional constraints to estimate the lower bound on the optimal network interference. The LP formulation can be solved in a much less time (less than an hour versus 12 hours) than the SDP formulation but yields a slightly looser lower bound than SDP on the optimal network interference. Note that the SDP and LP formulations are used only to demonstrate the performance of our Tabu-based and Greedy algorithms.

ILP. Recall that N is the set of network nodes, R_i is the number of radio interfaces for a node i, $\mathcal{K}$ is the set of

available channels, and $G_c(V_c, E_c)$ is the conflict graph. Also, $E(i)$ represents the set of vertices in V_c that represent the communication links incident on node $i \in N$.

We use the following set of binary integer (taking values 0 or 1) variables and constraints in our ILP formulation:

- Variables Y_{uk}, for each $u \in V_c$ and $k \in \mathcal{K}$. The variable Y_{uk} is 1 if and only if the vertex $u \in V_c$ is assigned the channel k. Essentially, the variables Y_{uk} define the channel assignment function. Since, each vertex in V_c is given exactly one channel, we have the following constraints:

$$Y_{uk} = \{0, 1\}, \quad \forall u \in V_c, \quad \forall k \in \mathcal{K}, \tag{4}$$

$$\sum_{k \in \mathcal{K}} Y_{uk} = 1, \quad \forall u \in V_c. \tag{5}$$

- Variables X_{uv}, for each edge $(u, v) \in E_c$. The variable X_{uv} is 0 only if the vertices u, $v \in V_c$ are assigned different channels.[4] The following equation defines the value of X_{uv} in terms of the Y variables:

$$X_{uv} = \{0, 1\}, \quad \forall (u, v) \in E_c, \tag{6}$$

$$X_{uv} \geq Y_{uk} + Y_{vk} - 1, \quad \forall (u, v) \in E_c, \quad \forall k \in \mathcal{K}. \tag{7}$$

  The variables X_{uv} are used to define the network interference (the objective function defined later).
- Variables Z_{ik}, for each network node $i \in N$ and channel $k \in \mathcal{K}$. The variable Z_{ik} is 1 if and only if some $u \in E(i)$ has been assigned a channel k; note that, u represents a communication link incident on $i \in N$:

$$Z_{ik} = \{0, 1\}, \quad \forall i \in N, \quad \forall k \in \mathcal{K}, \tag{8}$$

$$Z_{ik} \geq Y_{uk}, \quad \forall u \in E(i), \quad \forall i \in N, \quad \forall k \in \mathcal{K}, \tag{9}$$

$$Z_{ik} \leq \sum_{u \in E(i)} Y_{uk}, \quad \forall i \in N, \quad \forall k \in \mathcal{K}. \tag{10}$$

  The last equation above is used to enforce that Z_{ik} is 0 if there is indeed no vertex $u \in E(i)$ that has been assigned a channel k. The equation below enforces the interface constraint using Z variables:

$$\sum_{f=1}^{k} Z_{if} \leq R_i \quad \forall i \in N. \tag{11}$$

Objective function. Our objective function for the above ILP is to

$$\text{Minimize} \sum_{(u,v) \in E_c} X_{uv}.$$

4. If vertices u and v in V_c are assigned same channel, then X_{uv} can be 0 or 1. However, X_{uv} will be chosen to be 0 to minimize the objective function (see below), as there are no additional constraints involving X_{uv}. The additional constraints in (12) and (13) can be looked upon as derivations of (7).

Linear programming. Due to NP-hardness of ILP, solving the above ILP is intractable for reasonably sized problem instances. Thus, we relax the above ILP to a linear program (LP) by relaxing the integrality constraints. In particular, we replace (6), (4), and (8) by the following equation:

$$0 \leq X_{uv}, Y_{uk}, Z_{ik} \leq 1.$$

The solution to the relaxed LP gives only a lower bound on the optimal solution to the ILP. Through simulations, we have observed that the lower bound obtained by the above LP formulation is very loose. Thus, in order to obtain a tighter lower bound, we add additional constraints as follows:

Clique constraint. For each vertex $u \in V_c$, let S_u be the set of vertices in a maximal clique containing u. As discussed in Section 6.1, we can lower bound the number of monochromatic edges in a complete graph of size $|S_u|$ when colored by K colors as $\sigma(S_u, K)$ using (2). The above observation yields the following additional constraint:

$$\sum_{v,w \in S_u} X_{vw} \geq \sigma(S_u, K) \quad \forall u \in V_c. \tag{12}$$

Since the set of vertices $E(i)$ in V_c forms a clique in G_c and uses at most R_i colors (due to the interface constraint on node i), we also have the following constraint:

$$\sum_{(u,v) \in E(i)} X_{uv} \geq \sigma(E(i), R_i) \quad \forall i \in N. \tag{13}$$

The two additional constraints pose a lower bound on the interference on clique like subgraphs. This helps to reduce the gap between the actual integer optimum and the relaxed linear solution.

Number of variables and constraints. The number of variables in the above LP formulation is $|E_c| + K(|V_c| + N)$, and the total number of equations/constraints are $2(|V_c| + |N|) + K(2|V_c| + 2|N| + |E_c|)$, including the integrality constraints. We solve the LP using GLPK [44], a public-domain MIP/LP solver.

7 Generalizations

In the previous sections, for sake of clarity, we made various assumptions, namely, uniform traffic on all communication links, a binary interference model, and orthogonal channels. In this section, we generalize our techniques to relax these assumptions. These generalizations are quite useful in practical deployments. For example, the links in the network communication graph may carry different amounts of traffic. Thus, the average interference must be weighted by traffic as interfering traffic is not the same for all interfering link pairs. Also, channels—even when they are orthogonal in theory—do interfere due to device imperfections (e.g., radio leakage, improper shielding, etc.) [12]. Thus, modeling of nonorthogonal (i.e., interfering) channels is a good idea. In addition, this also allows us to explicitly utilize nonorthogonal channels. Finally, regardless of traffic and use of different channels, path loss effects can influence the degree of interference between two links—and thus, result in fractional interference between two links.

Nonuniform traffic and fractional interference. Let u and v be two vertices in the conflict graph, $r(u,v)$ (a real number between 0 and 1) be the level of interference between two links corresponding to the vertices u and v when both links carry saturated traffic. The level of interference $r(u,v)$ between pairs of links u and v can be computed using techniques similar to that in [21]; Section 9 gives a detailed description of how it is computed in our experimental study. Let $t(u)$ and $t(v)$ denote the normalized traffic (with respect to saturated traffic) on the links corresponding to the vertex u and v, respectively. Note that in our network model, we assume that the traffic is known a priori. Measurements of these parameters was discussed in Section 2. Based on the above notations, the overall network interference $I(f)$ for a given channel assignment function $f : V_c \rightarrow \mathcal{K}$ can be defined as follows: Let $M = \{(u,v) | u,\ v \in V_c \text{ and } f(u) = f(v)\}$. Then,

$$I(f) = \sum_{(u,v)\in M} t(u)t(v)r(u,v).$$

Note that $t(u)t(v)r(u,v)$ is a reasonable way to model the level of interference between the nodes u and v with given traffic loads, since $r(u,v)$ is the level of interference with saturated traffic and $t(u)$ and $t(v)$, the respective traffic loads, are normalized with respect to the saturated traffic.

For the generalized interference and traffic model, the Tabu-based and Greedy algorithms use the above definition of network interference; no additional changes are required. Similarly, the LP and SDP formulations of the channel assignment problem can be generalized by appropriately extending the objective function; no other changes are required in the list of variables and constraint equations.

Nonorthogonal channels. Let $c(k_1,k_2)$, a value between 0 and 1, denote the level of interference between two channels k_1 and k_2. For nonorthogonal channels, the overall network interference can be further generalized as follows for a given channel assignment function $f : V_c \rightarrow \mathcal{K}$:

$$I(f) = \sum_{(u,v)\in E_c} t(u)t(v)r(u,v)c(f(u),f(v)).$$

As before, Tabu-based and Greedy algorithms can use the above definition of network interference without any additional changes. However, in the LP formulation, we need to replace (7) by the following:

$$X_{uv} \geq Y_{uk_1} + Y_{vk_2} - 2 + c(k_1,k_2), \quad \forall (u,v) \in E_c,\ \forall k_1,k_2 \in \mathcal{K}.$$

Unfortunately, the SDP formulation cannot be generalized easily for nonorthogonal channels. The problem arises from the difficulty in choosing appropriate vectors a_i such that $a_i.a_j$ is proportional to $c(i,j)$ for all channels i, $j \in \mathcal{K}$. The values $c(i,j)$ are characteristics of the channel spectrum and can be measured independently.

8 Simulation Results

In this section, we study the performance of our designed algorithms for the channel assignment problem through extensive simulations. The main goal of the simulation-based evaluation is to understand the performance of our algorithms in large-scale network scenarios. In Section 9, we will present a detailed experimental evaluation of our algorithms on an 11-node multiradio mesh testbed. We present our performance results for two different settings. First, we evaluate a graph-theoretic performance metric and, then, evaluate throughput improvement using ns2 simulations. We start with discussing various algorithms used for comparison.

Algorithms. In addition to our designed algorithms (Tabu-based and Distributed Greedy) and the lower bounds obtained from the linear and SDP techniques, we also present results for two other algorithms for comparison. In particular, we simulate a modified version of the centralized CLICA heuristic presented in [9] for a slightly different version of the channel assignment problem.[5] We refer to the modified algorithm in [9] as CLICA-SCE. We also simulate a *random* algorithm that uses only a limited number of channels (equal to the number of radio interfaces), assigns a different channel to each radio interface and, then, selects a random interface (and hence, channel) for transmitting a packet. See Section 3 for a discussion on other related works.

We note here that the network interference metric is actually a localized metric since a communication link interferes with only "neighboring" communication links. Thus, we observed that the centralized version of the greedy algorithm performed almost exactly the same as the DGA.

8.1 Graph-Theoretic Performance Metric

In this set of experiments, we generate random networks by randomly placing a number of nodes in a fixed region and evaluate various algorithms based on a certain graph-theoretic performance metric. To solve LPs, we used GLPK [44], which is a public-domain MIP/LP solver, while to solve SDPs, we used DSDP 5.0 [43], [45], which uses an efficient interior-point technique.

Graph parameters. We consider two sets of random network, namely, dense and sparse networks, generated by randomly placing 50 nodes in 500×500 and 800×800 square meters of area, respectively.[6] In dense networks, the average node degree is around 10, while in sparse networks, the average node degree is around 5. Each node has the same number of radio interfaces and has a uniform transmission and interference range of 150 meters. Two nodes are connected by a communication link if they lie within each other's *transmission range*. Also, two communication links (i,j) and (g,h) interfere with each other if and only if either g or h lies within the *interference range* of i or j; this is based on the protocol interference model [1]. We assume orthogonal channels and uniform traffic on all links.

Performance metric. We evaluate the performance of our algorithms in random networks using the metric "fractional network interference." Given a channel assignment function f computed by an algorithm, the *fractional network interference* is defined as the ratio of network interference $(I(f))$ and the total number of edges in the conflict graph. This represents the number of conflicts that remain even after channel assignment relative to the number of conflicts in the

5. In CLICA [9], a communication link may multiplex between multiple channels, but in our network model, each communication link uses exactly one channel for transmission. We modify CLICA to use our network model.

6. We evaluated networks of size up to 750 nodes and varying densities, with similar performance results for all algorithms. However, the LP and SDP formulations for networks of size larger than 50 nodes took unreasonably long computation time.

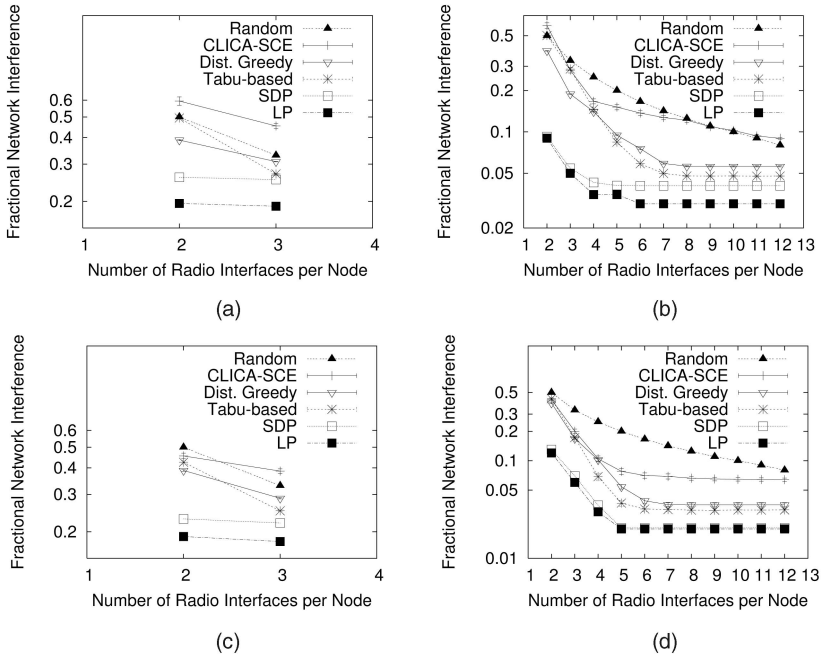

Fig. 3. Fractional network interference of solutions delivered by various algorithms compared with the lower bounds in dense or sparse networks for 3 or 12 channels. (a) Dense, 3 channels. (b) Dense, 12 channels. (c) Sparse, 3 channels. (d) Sparse, 12 channels.

single-channel network. The fractional network interference for the random algorithm is given by $\frac{1}{R}$, where R is the number of radios on each node. Note that the above performance metric is purely graph-theoretic, and hence, we do not use any network simulator for these experiments.

Results. In Fig. 3, we plot the fractional network interference for a varying number of radio interfaces/node in dense and sparse networks using 3 and 12 channels. In general, both our algorithms perform extremely well compared to the CLICA-SCE and random algorithms. The Tabu-based algorithm almost always performs better that than the DGA, except when the number of radios is very small. When the number of radios is very small, the second phase of Tabu-based algorithm is forced to perform many inefficient merge operations that lead to performance degradation.

The performance of our algorithms compared to the lower bounds obtained from the LP and SDP formulations shows that our algorithms deliver very good solutions, particularly for larger number of radios. Note that the vertical axis of the plots is presented in log scale for ease of viewing. The performance difference between the Tabu-based algorithm and the SDP lower bound is about 1 percent to 4 percent when the number of radios is large. We can also see that the SDP formulation delivers a much better lower bound than the LP formulation, for all parameter values. However, as we noted before, running SDP is significantly more computationally expensive (in terms of time and memory) than LP.

The comparison of plots for dense and sparse networks bring out interesting features. The fractional interference reduces with increase in number of radios per node; however, this trend saturates beyond a certain number of radios. This saturation point is reached with smaller number of radios for sparse networks than for dense networks, for the same number of channels. This is because the denser networks can potentially support more concurrent transmissions than the sparse networks. Similar trends were observed in [9].

8.2 ns2 Simulations

In this set of experiments, we study the impact of channel assignment in improving throughput in an 802.11-based mesh network. We compare the performance of various algorithms by measuring the *saturation throughput* using ns2 simulations over randomly generated networks. We consider networks of 50 nodes randomly placed in a 1,000 $\times$ 1,000 square meters area. The transmit power, receive, and carrier sense thresholds in the default setting of ns2 are such that the transmission range is 250 meters and the interference range is 550 meters. We used the same default radio parameters as in ns2 [46], except that we set the channel data rate to 24 megabits per second. All transmissions are unicast transmissions following the 802.11 MAC protocol with RTS/CTS, and the packet size is fixed to 1,000 bytes.

Performance for various traffic models. We use three different traffic models:

- *Single-hop traffic model*. This model consists of identical Poisson traffic for each communication link. The single-hop traffic model is useful to evaluate the performance in the case when all links in the network carry the same load.
- *Multihop peer-to-peer traffic model*. In this model, 25 randomly selected source-destination pairs communicate using multihop routes. The routes are computed statically using the shortest number of hops as the metric and do not change for the lifetime of the simulation.
- *Multihop gateway traffic model*. In this model, four random nodes are selected as gateways, and 25 source nodes send traffic to their nearest (in terms of hops) gateway. Routes are determined as in the previous

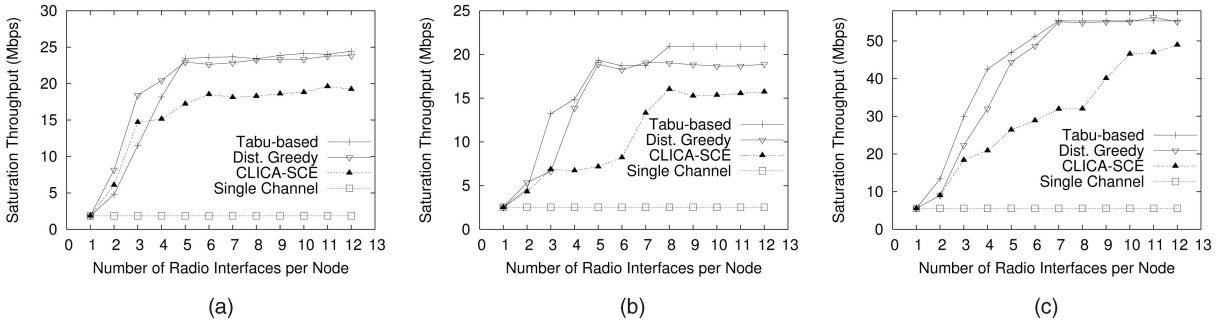

Fig. 4. Saturation throughput in ns2 simulations for 12 channels and various traffic models, namely, (a) single hop, (b) multihop peer-to-peer, and (c) multihop gateway.

traffic model. Such a traffic model will be common when the mesh network is used for Internet gateway connectivity.

Note that in the last two traffic models the traffic on the links is nonuniform. The traffic information is used in the channel assignment algorithms, as suggested in Section 7.

Fig. 4 plots *saturation throughput* against number of radio interfaces per node for the three traffic models and 12 channels (as we are experimenting with an 802.11a like system). We obtain the saturation throughputs as follows: For a particular number of radios and channels, we run a series of simulations, increasing the offered load each time, starting from a low value. We stop when the throughput does not increase any further with increase in the offered load.

We note that in all the three traffic models, our algorithms perform very well. We also see that the observations we made from the earlier graph-theoretic evaluations translate well into the ns2 results. The saturation throughput remain same after a certain number of radios, as inferred in the graph-theoretic simulations. Also, the relative performance of the algorithms in the ns2 simulations is the same as observed in the graph-theoretic simulations. This indirectly establishes the merit of the chosen interference model, optimization objective, and use of graph-theoretic measures as a method of performance evaluation.

Modeling nonorthogonal channels. So far, we have used only perfectly orthogonal channels. This however is a limitation in systems such as 802.11b, where few orthogonal channels are available. Since our techniques are general enough to handle nonorthogonal channels (Section 7), we now model a nonorthogonal channel situation.

We assume an 802.11b like system, where there are 11 channels, with only three of them being mutually orthogonal. For modeling the interference between nonorthogonal channels, we follow the technique outlined in Section 7. We use the data in [47] to model the "weighted" nature of conflicts. This data is obtained based on a simple analysis of the amount of overlapped spectrum between every pair of channels in 802.11b. We also did direct measurements on an 802.11b testbed to estimate interference between nonorthogonal channels and the values we obtained are similar to those quoted in [47]. Since such measurements can be very much hardware and environment specific, we stick to the data in [47].

In the ns2 simulator, we model interchannel interference as follows: Physical layer frames transmitted on channel k_1 arriving at a radio interface tuned to channel k_2 are reduced in power depending on the degree of noninterference. For example, if a k_1-frame arrives at a k_1-interface, the frame does not undergo any power reduction. On the other hand, if a k_1-frame arrives at a k_2-interface, where k_1 and k_2 are perfectly orthogonal, then the k_1-frame is completely silenced. Power reduction between 0 percent and 100 percent occur for other intermediate cases. In the simulator, the interference (e.g., carrier sense or collisions) is calculated only after such power reduction.

We use the peer-to-peer multihop traffic model (as defined before) to show the performance of our algorithms with nonorthogonal channels, see Fig. 5. We observe that both our algorithms perform better when using all available 11 channels than when using only the three mutually orthogonal channels. The factor of improvement is less in the Tabu-based algorithm compared to the DGA due to the inefficiency of the merge operations. Overall, the use of nonorthogonal channels is a better choice than restricting channel assignments to only orthogonal channels.

9 Experimental Evaluation

In this section, we present an experimental evaluation of our channel assignment algorithms compared to the random channel assignment algorithm and CLICA-SCE algorithm. We start by describing our testbed and then discuss about generating the conflict graph for a given topology of mesh nodes. We then describe the channel assignment procedure and present the performance results.

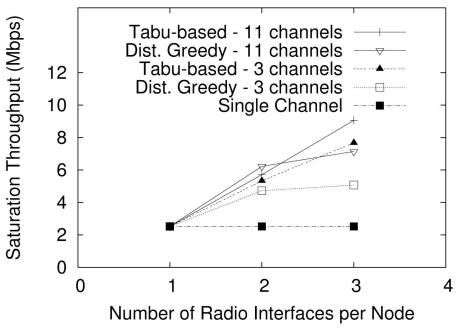

Fig. 5. Saturation throughput in ns2 simulations when using nonorthogonal channels with 802.11b-like multichannel model (11 channels with varying degrees of interference; three channels are mutually orthogonal).

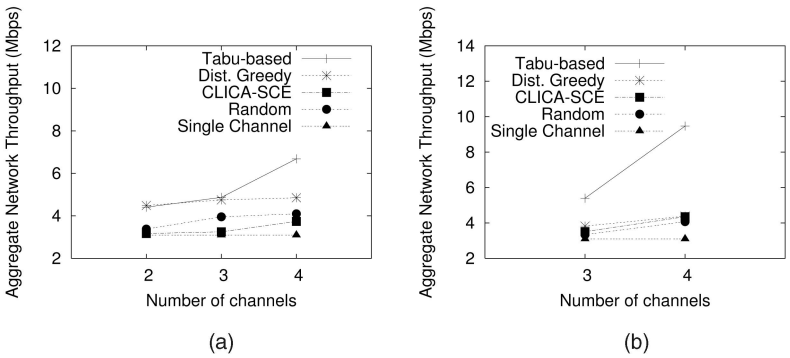

Fig. 6. Aggregate network throughput in the network when transmit power of each node is set to 11 dBm. (a) Two radio case. (b) Three radio case.

9.1 Testbed

The mesh testbed used in our experiments is an indoor wireless testbed that consists of 11 nodes, each of which is a Soekris [48] net4801 embedded computer running Pebble Linux [49] with the Linux 2.4.31 kernel. The PCI-slot in the embedded computer is expanded into four mini-PCI slots using RouterBoard 14 [50], which allows us for using four mini-PCI wireless cards. We use three 802.11 a/b/g mini-PCI wireless cards based on an Atheros [51] chipset with external antennas in each mesh node. We use the latest `madwifi` [52] driver for the 802.11 interfaces. In our experiments, we configured the 802.11 interfaces in 802.11a mode as there are 12 orthogonal channels, and the communication range of each wireless node is less compared to 802.11b/g modes so that we could get different multihop topologies within indoors. Due to board crosstalk or radio leakage [10], [12], there is interference between the radios in the same node even when they are configured to separate orthogonal channels. In order to over come this, we separated the three external antennas at a distance of about 1.5-feet based on measurements similar to [12]. With this setup, we could use 7 (channels 36, 44, 52, 60, 149, 157, and 165) out of the 12 orthogonal channels in each mesh node without interference between different radios on the same node.

9.2 Generating Communication Graph and Conflict Graph

The nodes in our mesh testbed are static and in order to generate different topologies, we used different transmit powers. In the experiments reported in this paper, we have used two different transmit powers (11 dBm and 15 dBm) to generate one sparse and one dense topology. The first step in our experimental procedure is to generate the communication graph for a given configuration of nodes. The communication graph is generated by allowing each node to transmit broadcast packets one after another and all other nodes measuring the delivery ratio and throughput. This takes about $O(n)$ time, where n is the number of nodes in the network. In our study, we repeat the above procedure multiple times and choose links with delivery ratio more than 80 percent as the stable links in the network and use only these links. The reason behind this is that the links that have poor delivery ratio are very unstable, making it harder to get good statistical confidence in the results with runs of reasonable lengths. Also, the number of stable links is much less compared to the total number of possible links in the network. This also reduces the time to compute the conflicts between link pairs. Among the seven orthogonal channels, channels 149, 157, and 165 are in the upper band of the 5-GHz spectrum, and the link characteristics using these channels are significantly different from the rest of the four channels we considered in the lower and middle band. Since all the channel assignment algorithms considered for the study here assume that all channels to be alike, we had to restrict our study to the four channels only (i.e., channels—36, 44, 52, and 60).

Once the links in the communication graph are decided, we schedule pairs of links simultaneously and measure the throughput on each link in the presence of transmission on the other link. If there are m stable links in the network, this measurement takes $O(m^2)$ time to complete. The conflict graph is then computed using the method used in [21]. The whole process of generating the conflict graph is automated and the conflict graph is fed as input to the channel assignment algorithms.

9.3 Performance Results

In our performance study, we use saturated load on all stable links as determined while generating the communication graph for each transmit powers simultaneously. Each experiment is repeated 10 times, and the average value is reported. Figs. 6 and 7 show the aggregate throughput in the network when using channel assignments from our Tabu-based and Greedy algorithm[7] compared to the random channel assignment and CLICA-SCE algorithms for two different topologies (sparse topology using 11 dBm transmit power in all node and dense topology using 15 dBm transmit power in all nodes). We also show the aggregate network throughput when using the same channel in all links. This serves as a base case.

For the sparse topology, Fig. 6a shows the case when each node uses two radios, and Fig. 6b shows the case when each node uses three radios. We see the Tabu-based algorithm perform extremely well compared the other algorithms in both the cases. There is a notable difference in aggregate throughput when using four channels in this topology using our Tabu-based algorithm. This is because the level of interference resulting from the channel assignment when using four channels is much less compared to when using 2 or 3 channels. The performance of the Greedy algorithm is not significantly better when compared to the random and CLICA-SCE algorithms. In the dense topology, both our

7. In order to make the experimental procedure simple, we considered a centralized version of our greedy algorithm. As noted in Section 8, the performance of both versions are similar.

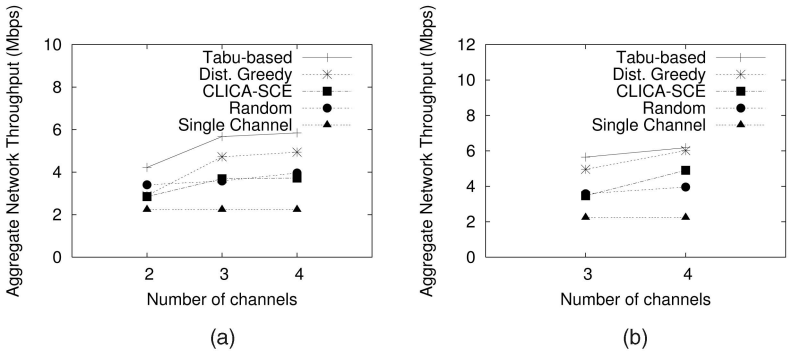

Fig. 7. Aggregate network throughput in the network when the transmit power of each node is set to 15 dBm. (a) Two radio case. (b) Three radio case.

algorithms perform well, and there is a notable increase in the aggregate throughput when using three channels compared to using two channels due to higher reduction in level of interference. Beyond three channels, the improvement is negligible. The improvement in aggregate throughput when using more number of channels and interfaces is largely dependent on the actual topology and the way links interfere in the network. Compared to the single channel case, there is a significant increase in aggregate throughput in the network. This shows the effectiveness of using good channel assignment algorithms.

10 Conclusion

In this paper, we have formulated and addressed the channel assignment problem in multichannel wireless mesh networks, where each node may be equipped with multiple radios. We have presented centralized and distributed algorithms that assign channels to communication links in the network with the objective of minimizing network interference. Using linear programming and SDP formulations of our optimization problem, we obtain tight lower bounds on the optimal network interference and empirically demonstrate the goodness of the quality of solutions delivered by our algorithms. Using simulations on *ns*2 and detailed experimental study on an 11-node multiradio mesh testbed, we observe the effectiveness of our approaches in improving the network throughput. One of the future directions is to consider assignment of multiple channels to each link.

Acknowledgments

Anand Prabhu Subramanian, Himanshu Gupta and Samir Das's research has been supported in part by the US NSF Grants OISE-0423460, CNS-0519734, CNS-0721455, IIS-0713186, CNS-0721701, and CNS-721665. Jing Cao's research has been supported in part by the China 863 Program 2006AA01Z259.

References

[1] P. Gupta and P.R. Kumar, "The Capacity of Wireless Networks," *IEEE Trans. Information Theory*, vol. 46, no. 2, 2000.

[2] S.-L. Wu, C.-Y. Lin, Y.-C. Tseng, and J.-P. Sheu, "A New Multi-Channel MAC Protocol with On-Demand Channel Assignment for Multi-Hop Mobile Ad Hoc Networks," *Proc. Int'l Symp. Parallel Architectures, Algorithms, and Networks (ISPAN)*, 2000.

[3] J. So and N. Vaidya, "Multi-Channel MAC for Ad Hoc Networks: Handling Multi-Channel Hidden Terminals Using a Single Transceiver," *Proc. ACM MobiHoc*, 2004.

[4] P. Bahl, R. Chandra, and J. Dunagan, "SSCH: Slotted Seeded Channel Hopping for Capacity Improvement in IEEE 802.11 Ad-Hoc Wireless Networks," *Proc. ACM MobiCom*, 2004.

[5] J. Shi, T. Salonidis, and E. Knightly, "Starvation Mitigation through Multi-Channel Coordination in CSMA Multi-Hop Wireless Networks," *Proc. ACM MobiHoc*, 2006.

[6] R. Chandra, P. Bahl, and P. Bahl, "MultiNet: Connecting to Multiple IEEE 802.11 Networks Using a Single Wireless Card," *Proc. IEEE INFOCOM*, 2004.

[7] A. Raniwala, K. Gopalan, and T. Chiueh, "Centralized Channel Assignment and Routing Algorithms for Multi-Channel Wireless Mesh Networks," *ACM SIGMOBILE Mobile Computing and Comm. Rev.*, vol. 8, no. 2, 2004.

[8] A. Raniwala and T. Chiueh, "Architechture and Algorithms for an IEEE 802.11-Based Multi-Channel Wireless Mesh Network," *Proc. IEEE INFOCOM*, 2005.

[9] M.K. Marina and S. Das, "A Topology Control Approach to Channel Assignment in Multi-Radio Wireless Mesh Networks," *Proc. Second Ann. Int'l Conf. Broadband Networks (Broadnets)*, 2005.

[10] A. Adya, P. Bahl, J. Padhye, A. Wolman, and L. Zhou, "A Multi-Radio Unification Protocol for IEEE 802.11 Wireless Networks," *Proc. First Ann. Int'l Conf. Broadband Networks (Broadnets '04)*, Oct. 2004.

[11] J. Tang, G. Xue, and W. Zhang, "Interference-Aware Topology Control and QoS Routing in Multi-Channel Wireless Mesh Networks," *Proc. ACM MobiHoc*, 2005.

[12] J. Robinson, K. Papagiannaki, C. Diot, X. Guo, and L. Krishnamurthy, "Experimenting with a Multi-Radio Mesh Networking Testbed," *Proc. Int'l Workshop Wireless. Network Measurements (WiNMee)*, 2005.

[13] A. Hertz and D. de Werra, "Using Tabu Search Techniques for Graph Coloring," *Computing*, vol. 39, no. 4, 1987.

[14] A. Frieze and M. Jerrum, "Improved Approximation Algorithms for MAX k-CUT and MAX BISECTION," *Algoritmica*, vol. 18, 1997.

[15] K. Jain, J. Padhye, V.N. Padmanabhan, and L. Qiu, "Impact of Interference on Multi-Hop Wireless Network Performance," *Proc. ACM MobiCom*, 2003.

[16] V.S.A. Kumar, M.V. Marathe, S. Parthasarathy, and A. Srinivasan, "Algorithmic Aspects of Capacity in Wireless Networks," *SIGMETRICS Performance Evaluation Rev.*, vol. 33, no. 1, 2005.

[17] M. Alichery, R. Bhatia, and L. Li, "Joint Channel Assignment and Routing for Throughput Optimization in Multi-Radio Wireless Mesh Networks," *Proc. ACM MobiCom*, 2005.

[18] X. Lin and S. Rasool, "A Distributed Joint Channel-Assignment, Scheduling and Routing Algorithm for Multi-Channel Ad Hoc Wireless Networks," *Proc. IEEE INFOCOM*, 2007.

[19] A. Kashyap, S. Ganguly, and S.R. Das, "A Measurement-Based Approach to Modeling Link Capacity in 802.11-Based Wireless Networks," *Proc. ACM MobiCom*, 2007.

[20] C. Reis, R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan, "Measurement-Based Models of Delivery and Interference in Static Wireless Networks," *Proc. ACM SIGCOMM*, 2006.

[21] J. Padhye, S. Agarwal, V. Padmanaban, L. Qiu, A. Rao, and B. Zill, "Estimation of Link Interference in Static Multi-Hop Wireless Networks," *Proc. Internet Measurement Conf. (IMC)*, 2005.

[22] *The CoMo Project*, http://como.intel-research.net/, 2008.

[23] M. Gong, S. Midkiff, and S. Mao, "A Combined Proactive Routing and Multi-Channel MAC Protocol for Wireless Ad Hoc Networks," *Proc. Second Ann. Int'l Conf. Broadband Networks (Broadnets),* 2005.
[24] R. Maheshwari, H. Gupta, and S.R. Das, "Mutichannel MAC Protocols for Wireless Networks," *Proc. Int'l Conf. Sensor and Ad Hoc Comm. and Networks (SECON),* 2006.
[25] J. So and N.H. Vaidya, "Multi-Channel Mac for Ad Hoc Networks: Handling Multi-Channel Hidden Terminals Using a Single Transceiver," *Proc. ACM MobiHoc,* 2004.
[26] R. Vedantham, S. Kakumanu, S. Lakshmanan, and R. Sivakumar, "Component Based Channel Assignment in Single Radio, Multi-channel Ad hoc Networks," *Proc. ACM MobiCom,* 2006.
[27] Q. Xue and A. Ganz, "Temporal Topology Control in Multi-Channel Multihop Wireless Access Networks," *Proc. Second Ann. Int'l Conf. Broadband Networks (Broadnets),* 2005.
[28] A. Das, H. Alazemi, R. Vijayakumar, and S. Roy, "Optimization Models for Fixed Channel Assignment in Wireless Mesh Networks with Multiple Radios," *Proc. Int'l Conf. Sensor and Ad Hoc Comm. and Networks (SECON),* 2005.
[29] A.H.M. Rad and V. Wong, "Joint Channel Allocation, Interface Assignment and MAC Design for Multi-Channel Wireless Mesh Networks," *Proc. IEEE INFOCOM,* 2007.
[30] K. Ramachandran, E. Belding, K. Almeroth, and M. Buddhikot, "Interference-Aware Channel Assignment in Multi-Radio Wireless Mesh Networks," *Proc. IEEE INFOCOM,* 2006.
[31] B. Ko, V. Misra, J. Padhye, and D. Rubenstein, "Distributed Channel Assignment in Multi-Radio 802.11 Mesh Networks," *Proc. IEEE Wireless Comm. and Networking Conf. (WCNC),* 2007.
[32] P. Kyasanur and N.H. Vaidya, "Routing and Link-Layer Protocols for Multi-Channel Multi-Interface Ad Hoc Wireless Networks," *ACM SIGMOBILE Mobile Computing and Comm. Rev.,* vol. 10, no. 1, pp. 31-43, 2006.
[33] P. Kyasanur and N.H. Vaidya, "Capacity of Multi-Channel Wireless Networks: Impact of Number of Channels and Interfaces," *Proc. ACM MobiCom,* 2005.
[34] A. Coja-Oghlan, C. Moore, and V. Sanwalani, "MAX k-CUT and Approximating the Chromatic Number of Random Graphs," *Proc. Int'l Colloquium on Automata, Languages and Programming (ICALP),* 2003.
[35] V. Kann, S. Khanna, J. Lagergren, and A. Panconesi, "On the Hardness of Approximating Max k-Cut and Its Dual," *Chicago J. Theoretical Computer Science,* no. 2, June 1997.
[36] *Meru Networks,* http://www.merunetworks.com/index.shtml, 2007.
[37] *Tropos Networks,* http://www.tropos.com, 2008.
[38] M.X. Goemans and D.P. Williamson, "Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming," *J. ACM,* vol. 42, no. 6, 1995.
[39] P.M. Vaidya, "A New Algorithm for Minimizing Convex Functions over Convex Sets," *Math. Programming,* vol. 73, no. 3, 1996.
[40] M. Grotschel, L. Lovasz, and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization.* Springer, 1987.
[41] F. Alizadeh, "Interior Point Methods in Semidefinite Programming with Applications to Combinatorial Optimization," *SIAM J. Optimization,* vol. 5, pp. 13-51, 1995.
[42] R. Montemanni, D. Smith, and S. Allen, "Lower Bounds for Fixed Spectrum Frequency Assignment," *Annals of Operations Research,* vol. 107, Oct. 2001.
[43] S.J. Benson and Y. Ye, "DSDP5: Software for Semidefinite Programming," submitted to *ACM Trans. Math. Software,* Technical Report ANL/MCS-P1289-0905, Math. and Computer Science Division, Argonne Nat'l Laboratory, http://www.mcs.anl.gov/benson/dsdp, Sept. 2005.
[44] *GLPK: GNU Linear Programming Kit,* http://www.gnu.org/software/glpk/glpk.html, 2003.
[45] S.J. Benson, Y. Ye, and X. Zhang, "Solving Large-Scale Sparse Semidefinite Programs for Combinatorial Optimization," *SIAM J. Optimization,* vol. 10, no. 2, pp. 443-461, 2000.
[46] *The Network Simulator ns-2,* http://www.isi.edu/nsnam/ns/, 2008.
[47] *Cirond Technologies Inc. (2002) Channel Overlap Calculations for 802.11b Networks,* white paper, http://www.cirond.com/White_Papers/FourPoint.pdf, 2002.
[48] *Soekris Engineering,* http://www.soekris.com/, 2008.
[49] *NYCwireless Pebble Linux,* http://www.nycwireless.net/pebble, 2008.
[50] *Router Board—RB14,* http://www.routerboard.com/rb11.html, 2008.
[51] *Atheros Communications,* http://www.atheros.com, 2008.
[52] *MADWIFI Project,* http://sourceforge.net/projects/madwifi/, 2007.

Anand Prabhu Subramanian received the BE degree in computer science and engineering from Anna University, Chennai, India, in 2004 and the MS degree in computer science from Stony Brook University in 2007. He is a PhD student in the Computer Science Department, Stony Brook University, New York. His research interests include wireless networks, systems, and algorithms. His current research focuses on improving capacity and connectivity in Wi-Fi-based ubiquitous wireless access networks. More information about his research can be found at http://www.cs.sunysb.edu/~anandps. He received the best paper award in ACM MobiSys Conference in 2007. He is a student member of the IEEE.

Himanshu Gupta received the BTech degree in computer science and engineering from the Indian Institute of Technology in 1992, Bombay, and the MS and PhD degrees in computer science from Stanford University in 1999. He joined the faculty of the Department of Computer Science, Stony Brook University, in Fall 2002. His recent research activities focus on theoretical issues in ad hoc wireless networks and sensor networks and sensor network databases. His other research interests are in database systems and theory, wherein he is interested in materialized views, (multiple) query optimization, and data analysis.

Samir R. Das received the PhD degree in computer science from the Georgia Institute of Technology, Atlanta, in 1994. He is currently an associate professor in the Computer Science Department, State University of New York, Stony Brook. His research interests are in wireless networking and mobile computing, focusing on protocols, systems, and performance evaluation. He received the US National Science Foundation's CAREER Award in 1998 and the best paper award in ACM MobiSys Conference in 2007. He has been a speaker in the distinguished visitor program of the IEEE Computer Society from 2001 to 2003. He is the cochair of the technical program committee for the ACM MobiHoc Symposium in 2001 and ACM MobiCom Conference in 2004. He currently serves or has served on the editorial board of the *IEEE/ACM Transactions on Networking*, *IEEE Transactions on Mobile Computing*, *ACM/Kluwer Wireless Networks Journal*, and the *Ad Hoc Networks* journal. More information about him and his research can be found at http://www.cs.sunysb.edu/~samir. He is a member of the IEEE.

Jing Cao received the BS degree in computer science and technology from Beihang University in 2004. He is currently a PhD student in the computer school of Beihang University, research member at State Key Laboratory of Virtual Reality Technology and Systems, and a visiting scholar in the Computer Science Department, Stony Brook University. His research interests include wireless networks, systems, and algorithms. His current work focuses on ad hoc networking, TDMA over Wi-Fi, and mesh network. He is a member of the IEEE.