Routing and Channel Allocation in Rural Wireless Mesh Networks

Partha Dutta, Sharad Jaiswal, Rajeev Rastogi
Bell Labs Research India
Bangalore, India

Abstract*— IEEE 802.11 Wi-Fi equipment based wireless mesh networks have recently been proposed as an inexpensive approach to connect far-flung rural areas. Such networks are built using high-gain directional antenna that can establish long-distance point-point links. In recent work [16], a new MAC protocol named *2P* has been proposed that is suited for the interference pattern within such a network. However, the 2P protocol requires the underlying graph (for each 802.11 channel) to be bi-partite. Under the assumption that 2P is the MAC protocol used in the mesh network, we make the following contributions in this paper. Given K non-interfering 802.11 channels, we propose a simple cut-based algorithm to compute K bi-partite sub-graphs (on each of which the 2P protocol can be run separately). We establish the class of graphs that can thus be completely covered by K bi-partite subgraphs. For the remaining set of graphs, we look into the "price" of routing all end-to-end demands over *only* the bi-partite subgraphs. We analytically establish what fraction of the max flow of the original mesh-graph can be routed over the bi-partite subgraphs. Finally we look into the problem of mismatch between the load on a link (as computed by max flow) and it's effective capacity under a given channel allocation. We propose heuristics to cluster links with similar loads into the same bi-partite graphs (channels) and through comprehensive numerical simulations show that our heuristics come very close to the best possible flow.

I. Introduction

In this paper, we describe channel allocation and routing algorithms for long-distance rural wireless mesh networks. Rural networks exist in population areas with very low paying capacity. Hence a very important requirement for these networks is to minimize the infrastructure costs. The cost of laying wire to rural areas is prohibitively expensive and this is also true of traditional wide area wireless technologies such as cellular networks and upcoming technologies like IEEE 802.16 Wi-Max. Recent work [16] in the literature has demonstrated an alternative approach by building rural mesh network prototypes using IEEE 802.11 Wi-Fi equipment. IEEE 802.11 equipment is highly commoditized and goes a long way in addressing the cost issues for this environment, and is thus the cheapest option to build such networks.

As depicted in Figure 1, a typical rural mesh network would consist of a cluster of villages connected with each other through point-to-point wireless links. Some special nodes in this mesh, called *gateway* nodes, will be connected to the wired internet. Other mesh nodes will connect to the gateway node (and thus, to the rest of the internet) through one or more hops in the mesh. Rural mesh networks are characterized by

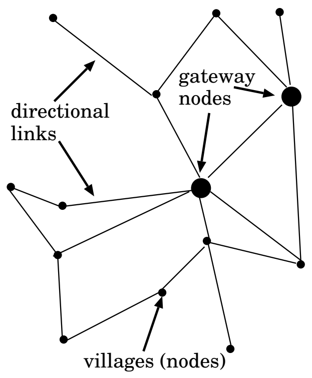

Fig. 1. A wireless rural mesh network

a fixed, outdoor topology (a node in this network will be a village) and very long-distance links between the nodes (about 10-15kms).

However, the 802.11 MAC was originally designed for (and widely deployed in) short-distance campus area networks with mobile nodes, and thus may not be well suited for deployment in the long distance, fixed topology, rural network environment. But, as [16] has demonstrated, by using high-gain directional antennae, a line-of-sight long-distance ($>$ 25 kms.) 802.11 link can be established. While this apparently physical limitation of 802.11 can be overcome by using such special-purpose antennae, there remain other issues.

Radiation leakage from the directional antennae causes adjacent links at a node to interfere with each other in certain communication modes. Specifically, a node cannot simultaneously transmit and receive on its adjacent links on the same channel. (This is called *Mix-Rx-Tx* interference [16].) An obvious approach towards addressing this problem is to operate the interfering links in different *non-interfering channels* of IEEE 802.11. The number of such channels, however, is limited - e.g. just 3 in 802.11b and 802.11g.

The presence of Mix-Rx-Tx interference affects the throughput of the mesh graph by restricting the number of links that can operate simultaneously. Specifically, as we will explain in more detail later, Mix-Rx-Tx interference imposes the restriction that only a *bi-partite* subgraph of the network graph can be active on one channel. Given this fundamental restriction, the crucial questions are - how to select bi-partite subgraphs to route the traffic of the entire mesh graph? And, what is the drop in performance suffered by routing only over the links in the bi-partite subgraphs as compared to routing over the entire mesh graph? Towards this end, we make several contributions.

We consider a mesh graph $G = (V, E)$, a set of source-destination demands D and K non-interfering 802.11 channels. Given that only a bi-partite subgraph can be activated for a given channel, it follows that with K channels we can simultaneously activate only K bi-partite sub-graphs. We propose an approach to find K bi-partite subgraphs by recursively applying a cut algorithm on G. We then show that if this cut algorithm always gives us the *Max-Cut* of the underlying graph, then if at least λ fraction of each source-destination demands can be routed over G, then in the worst-case $\lambda' = \Omega(\frac{2^K \lambda}{2^K + log(|E|)})$ fraction of each source-destination demands can be met by routing *only* over the K bi-partite subgraphs. We also identify the class of graphs for which all of the traffic that can be carried over the mesh graph can also be carried by the bi-partite subgraphs (i.e., $\lambda' = \lambda$).

Given our algorithm to route traffic over K bi-partite subgraphs of G, the next question is how to schedule the links on the bi-partite graphs. In recent work, Raman et al. [16] have proposed a new MAC protocol (called *2P*) to schedule links within a bi-partite graph in the presence of Mix-Rx-Tx interference. We adopt this protocol for scheduling links since it has the nice property of ensuring 100% utilization of links. However, a shortcoming of this protocol is that it requires all links in a bi-partite graph to be active for the same fraction of time in a given direction. To cope with this shortcoming, as part of our work, we also propose heuristics to cluster links with similar fractions, within the same bi-partite graph. Finally, we carry out extensive simulations to validate our observations and evaluate the quality of the proposed routing and channel allocation schemes.

A. Related work

The paper closest to our work is [15]. In this work Raman et al. consider the channel allocation problem in rural mesh networks. They also assume that the 2P protocol will be used for scheduling over bi-partite subgraphs in the given mesh graph. The authors consider the *specific* case of how to cover the input graph with bi-partite subgraphs when 3 non-interfering channels (as is the case in IEEE 802.11b) are available. They observe that any 6-edge colorable graph (and thus, by Vizing's theorem, any graph with a maximum degree 5) can be covered by 3 bi-partite subgraphs, and propose an algorithm to achieve this based on edge coloring.

We consider the *general* case of how to cover an input graph with bi-partite subgraphs when K channels are available. This can be important since the number of 802.11 channels that are available to the mesh network can vary. For example, 802.11a equipment provides 11 non-interfering channels, as compared to the 3 made available by 802.11b and 802.11g. In some cases, channels may have to be set aside for a local WLAN within a village or because of RF pollution (interference from other neighboring networks). On an opposite note, a recent work [8] has demonstrated the possibility of squeezing out 4 non-interfering channels from IEEE 802.11b and g equipment. Therefore, in our work, using a cut algorithm to identify bi-partite graphs, we describe the class of graphs that can be completely covered by any given K channels (i.e., by K bi-partite subgraphs).

In the case when K bi-partite graphs are not sufficient to cover the entire mesh graph, we also provide guarantees on the fraction of traffic that can be routed over the bi-partite subgraphs, as compared to routing over the entire mesh graph. This is a an aspect of our work not covered in [15]. Finally, as in [15] we also propose and evaluate heuristics to cluster links with similar fractions in the same bi-partite graph. However, we propose a novel metric, based on the desired *interval* of fractions that can be assigned to a link, to match links with bi-partite subgraphs. This approach offers more flexibility in finding the best assignment between links and bi-partite graphs and, as our initial experiments suggest, performs better than heuristics proposed in [15].

The Digital Gangetic Plains project [6] in and around Kanpur, India is an operational example of the type of rural mesh network that motivates this work. There have recently been other examples of community wireless mesh networks, such as [19] and MIT's Roofnet [7]. However these projects are, for most part, built using omnidirectional antenna, and occupy smaller areas.

Link scheduling in wireless mesh networks is a much studied problem. However, for our particular setting (long-distance, fixed-topology meshes with directional antennae), the 2P protocol [16] is the only distributed scheduling protocol that we know of. (However, refer to [10], [9], [5], [18], [2] for other instances of link scheduling in wireless mesh networks.)

There has also been a considerable amount of work on channel allocation in wireless mesh networks and analyzing how to meet end-to-end demands, for example, in [3], [17], [13], [14]. These works principally differ from ours in that they consider omnidirectional antennae and thus analyze networks with different interference properties as compared to ours.

B. Organization

The paper is organized as follows. In Section II we describe the rural mesh network architecture and communication model. We describe how the links interfere with each other, and explain the 2P protocol used to schedule links given this interference in a bi-partite graph. We formally describe the goals of this paper in Section III. In Section IV, we present our approach to select bi-partite subgraphs from a graph G given K non-interfering channels. We also describe a large class of graphs that can be completely covered by K bi-partite subgraphs. In Section V, we consider node demands, and analyze what fraction of node demands, that can be routed over the entire mesh graph, can also be routed over only the bi-partite subgraphs. In Section VI, through extensive numerical simulations, we evaluate our proposed schemes and heuristics.

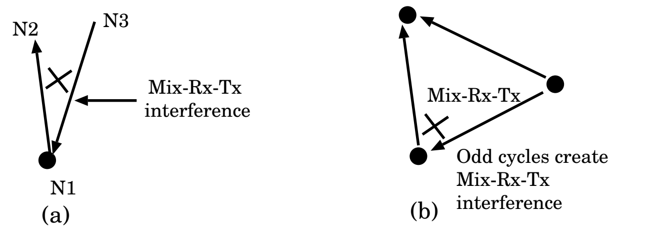

Fig. 2. (a) Mix-Rx-Tx interference at node $N1$ between transmission on link $(N1, N2)$ and receptions on link $(N1, N3)$ (b) Odd cycles in a graph cause Mix-Rx-Tx interference.

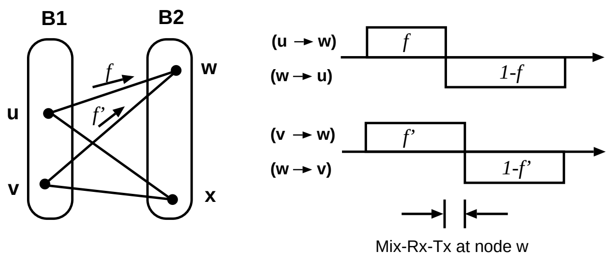

Fig. 3. Different fractions on adjacent links cause Mix-Rx-Tx interference.

Finally, we conclude with a summary of our contributions and directions for future work in Section VII.

II. System Model

In this section, we describe the architecture of a wireless rural mesh network. We also describe how the nodes will communicate with each other and discuss the interference model, i.e., the cases in which the network links will interfere with each other. We describe the constraints (based on the interference model) on which links can be active simultaneously and then briefly present the 2P link scheduling protocol proposed by Raman et al. [16] that aims to schedule network links within these constraints.

a) Rural mesh network architecture: As depicted in Figure 1, the nodes in the mesh will be villages. Some nodes will be designated as *gateway* nodes that will connect to the wired Internet. The gateway nodes will connect to other mesh nodes through line-of-sight long-distance links. Nodes connected to each other through directional antennas can directly communicate with each other, and through one or more such hops be connected to the gateway nodes.

b) Interference.: We have described how nodes in our mesh networks will communicate with each other using directional antennae. While directional antennae are designed to transmit and receive in a specific direction, the directionality of this radiation becomes effective only at longer distances from the sender. This is also called the *near field effect*. Because of this effect (as shown in Figure 2(a)), at any node, simultaneous transmissions and receptions on the *same channel* are not possible since the transmissions will interfere with the receptions. This is called Mix-Rx-Tx interference.[1] Avoiding Mix-Rx-Tx interference, while keeping all links active, requires that there be no odd cycles among the edges in the graph (as shown in Figure 2(b)). This further implies that only a bi-partite subgraph of the input graph can be activated for a given channel at any point of time.

[1]Apart from Mix-Rx-Tx interference, the point-to-point links can overlap spatially. Because of side-lobe radiation, adjacent directional links need a minimum angular separation to operate. Otherwise, the receiving ends of such links can be affected by the transmissions of all overlapping links. We consider dealing with interference arising because of overlapping links to be a topology design problem. Therefore, in this paper, we assume that following some topology design process, we are given a network graph that satisfies the angular separation criterion.

Another observation (as pointed out in [16]) is that while Mix-Rx-Tx interference prevents simultaneous Tx and Rx at a node, a node is allowed to synchronously transmit (or receive) on all its adjacent links. This is called SynTx (or SynRx). SynRx/SynTx together are known as SynOp: synchronous operation of links at a node. In recent work, Raman et al. [16] have proposed the *2P* MAC protocol based on SynOp. The protocol operates on bi-partite graphs by switching each node between two phases: SynRx and SynTx. When a node switches from SynRx to SynTx, its neighbors switch from SynTx to SynRx, and vice versa. The 2P protocol has a desirable property that it guarantees 100% utilization of all links (i.e., the link is active in one direction or the other). However, the 2P algorithm has a constraint on the fraction of time links are active in a given direction [15]. Consider the 2P algorithm on a bi-partite subgraph (say, with two independent sets B1 and B2) and operating on a single channel. Assume that a link is always active in one direction or the other. Then the fraction of time a link is active in a given direction (say from B1 to B2) must be identical for all links. Otherwise, if any two links are active for different fractions of time from B1 to B2, then, as every link is always active in one direction or the other, this difference in fractions propagates through the graph, and gives rise to different fractions at some pair of adjacent links. But, as shown in Figure 3, different fractions at two adjacent links cause Mix-Rx-Tx interference at the common node.

Ensuring that all links are active in a given direction for the same fraction of time, may result in reduced throughput as the optimal flow allocation in the graph may require different fractions for different links. Thus links in the same bi-partite subgraph should have identical fractions, although different bi-partite subgraphs can have different fractions. Later in the paper we also present and evaluate heuristics to cluster links with similar fractions in the same bi-partite graphs.

III. Problem statement

We now formally define the objectives of this paper. We are given an undirected graph $G = (V, E)$, where each link is of capacity L,[2] K channels, and a set $D = \{(s_i, t_i, d_i)\}$ of demands, where (s_i, t_i, d_i) denotes a demand of $d_i > 0$ from s_i to t_i. Due to the underlying scheduling algorithm,

[2]We assume equal link capacities for ease of presentation. Our results can be easily extended for unequal link capacities.

any subgraph on which routing can be done should satisfy the following two requirements.

R1. The subgraph is a union of K edge-disjoint bi-partite subgraphs, say $B_1, \ldots, B_K$ (where each bi-partite subgraph corresponds to a set of links that use the same channel).

R2. Let B_k^0 and B_k^1 denote the two independent sets of each bi-partite subgraph B_k. Then for each B_k, the capacities of all links in B_k in the direction from B_k^0 to B_k^1 are identical (and the capacities in the reverse direction are also identical, because the total capacity of a link is L). We call the fraction of time a link is operational from B_k^0 to B_k^1 as the *fraction* assigned to the link, i.e., the ratio of the capacity allocated to the direction B_k^0 to B_k^1 to L. Thus R2 requires that the fraction of all links in a bi-partite subgraph are identical [15].

The *concurrent flow value* of a routing is the largest λ such that the routing satisfies λ fraction of all demands. For a graph G, the *maximum concurrent flow* is defined as the maximum λ such that there is a routing over G with concurrent flow value λ [1].

We first consider the symmetric demands problem. Here we assume that demands are symmetric, i.e., if $(s_i, t_i, d_i) \in D$ then $(t_i, s_i, d_i) \in D$. Note that, for symmetric demands, there is a routing that achieves max concurrent flow, and which it assigns equal loads on both directions of a link. Hence, we can satisfy requirement R2 by assigning a fraction of 0.5 to all links. Now, assuming that R2 is already satisfied, among all subgraphs of G that satisfy R1, we want to find one that has the highest maximum concurrent flow.

We next consider the same problem but with asymmetric demands, i.e., we make no assumptions on demands (and hence, we cannot assume that R2 is already satisfied). Consider all subgraphs of G and assignment of fractions to each link (i.e., assignment of capacities to both directions of each link) such that requirements R1 and R2 are satisfied. We want to find a subgraph and assignment of fractions that has the highest maximum concurrent flow.

IV. Partitioning into Bi-partite graphs

To obtain K bi-partite graphs, we observe that edges in a cut naturally specify a bi-partite graph. In the following, we will use the term cut and bi-partite graph interchangeably. (We will denote the number of edges in a cut C by $|C|$.) We obtain K bi-partite subgraphs by iteratively applying a cut algorithm $\mathcal{C}$. On the original graph $G = (V, E)$, we apply $\mathcal{C}$, and the first bi-partite graph is given by the edges in the first cut. We then remove the edges of the first bi-partite graph from G, and apply the cut algorithm again to obtain the next bi-partite graph. We repeat the process K times to obtain the K bi-partite graphs.

Ideally, we would like our cut algorithm to cover the graph using the smallest number of cuts. As finding a max-cut for a general graph is an NP-hard problem,[3] we use the following simple local search based $1/2$ approximation of max-cut [20]. (We denote this algorithm by *LS*.) Initially, the set of nodes are arbitrarily divided into two sets. Each step of the algorithm, called a *flip*, does the following. It selects a node v such that v has more neighbors in its own set than in the other set, and then moves v to the other set. The algorithm terminates when there is no node that can be flipped.[4] It is easy to see that the algorithm takes at most $|E|$ flips to terminate – each flip increases the number of edges in the cut by at least 1. Moreover, upon termination, for each node v, the degree of v in the cut is greater than or equal to its degree in the remaining graph (otherwise, v can be flipped). In other words, an LS-cut satisfies the following property:

P1. After removing the cut from any graph G, the degree of each node in the remaining graph is at most half of its degree in G. ■

We now characterize a class of graphs that can be covered by K LS-cuts, denoted by $B_1, \ldots, B_K$. Let $LS_K = \cup_{1 \le k \le K} B_k$.

Lemma 1: For every node v, the degree of v in $G - LS_K$ is at most $d/2^K$, where d is the degree of v in G.

Proof: The lemma follows from an obvious induction on K using property P1. Here, we simply show the induction step. Let d be the degree of v in G. Suppose that after K cuts, the degree of v in $G - LS_K$ is $y \le d/2^K$. Now consider the $K + 1^{st}$ cut on $G - LS_K$. From property P1, the degree of v in $G - LS_{K+1}$ is at most $y/2 \le d/2^{K+1}$. ■

Theorem 2: If the degree of every node in G is at most $2^K - 1$ then LS_K covers G.

Proof: Suppose that the maximum degree of G is at most $2^K - 1$. Consider any node v with degree d. From Lemma 1, the number of incident edges of v in $G - LS_K$ is at most $d/2^K \le (2^K - 1)/2^K < 1$. Thus G is covered by LS_K. ■

In 802.11b and g, where there are three non-interfering channels, Theorem 2 implies that *LS*-cuts can cover any graph whose maximum degree is at most 7. In contrast, with three channels, the approach of [15] can only cover graphs whose maximum degree is at most 5.

V. Meeting node demands

We now consider node demands and provide guarantees on the max (concurrent) flow for the subgraph LS_K obtained in the previous section. Obviously, for the same set of demands, the max flow of LS_K can be at most the max flow of G, and the equality trivially holds when LS_K covers G (see Theorem 2). Thus we compare the max flow of LS_K with

[3]Note, however, that max-cut can be solved in polynomial time for planar graphs [12].

[4]The bipartite subgraph B output by LS algorithm might be disconnected even if the input graph G is connected. It is, however, straightforward to move some links from $G - B$ to B such that B becomes connected while remaining bipartite.

that of G. We consider two cases: symmetric demands and asymmetric demands.

Symmetric Demands. Recall that, for symmetric demands, there is a routing that achieves max flow, and which assigns equal loads for both directions of a link. Thus, we can always satisfy requirement R2 by assigning a fraction of 0.5 to all links. So, in this section, we assume that requirement R2 is already satisfied.

We now show a worst-case guarantee on the max flow of a subgraph that is obtained by applying max-cut (instead of LS-cut). We denote these K max-cuts by $C_1, \ldots, C_K$. Let $M_K = \cup_{1 \le k \le K} C_k$. We use the following property of a max-cut to show our result.

P2. A cut MC of a graph G' is a max-cut if and only if, for every cut C of G', the number of edges in $MC \cap C$ is at least $|C|/2$. ■

Observe that, every max-cut also satisfies property P1. To see why, consider the cut defined by all incident edges of a vertex v in graph G. From property P2 of max-cut, the degree of v, in the graph obtained by removing a max-cut from G, is at most half of the degree of v in G. Thus, Lemma 1 and Theorem 2 also hold for max-cut.

Lemma 3: For any cut C of G, the number of edges in C that are also in M_K is at least $(1 - 1/2^K)|C|$.

Proof: The proof follows from a simple induction on K using property P2. Here, we describe the induction step. Suppose that after K max-cuts, the number of edges of C in M_K is $y \ge (1 - 1/2^K)|C|$. Let C' be the remaining edges of C in $G' = G - M_K$. Note that C' is a cut of G' and contains $|C| - y$ edges of C. Consider the $K + 1^{st}$ max-cut C_{K+1}. From property P2, the number of edges of C' in C_{K+1} is at least $|C'|/2$. Thus the number of edges of C in M_{K+1} is at least $y + (|C| - y)/2 \ge (1 - 1/2^{K+1})|C|$. ■

Corollary 4: For any cut C of G, the ratio of the number of edges in C that are also in M_K to the number of edges in C that are also in $G - M_K$ is at least $2^K - 1$. ■

Lemma 5: On removing a max-cut from a graph G', the remaining graph G'' contains at most half of all the edges in G'.

Proof: Recall that max-cut satisfies property P1. Thus, the sum of the degree of all nodes in G'' is at most half of the sum of the degree of all nodes in G'. Therefore, the number edges in G'' is at most half of the number of edges in G'. ■

Lemma 6: The number of edges in $G - M_K$ is at most $|E|/2^K$.

Proof: The proof is by induction on K. Suppose that after K max-cuts, the number of edges in $G - M_K$ is $z \le |E|/2^K$. Consider the $K + 1^{st}$ max-cut C_{K+1} in the remaining graph $G' = G - M_K$. From Lemma 5, the number of edges of G' in $G' - C_{K+1}$ is at most $z/2 \le |E|/2^{K+1}$. ■

Theorem 7: Suppose that demands are symmetric, and at least λ fraction of every demand can be concurrently routed over G. Then $\Omega(\frac{2^K \lambda}{2^K + log(|E|)})$ fraction of every demand can be concurrently routed over M_K.

Proof: When λ fraction of every demand is concurrently routed over G, every link has at most L load. Then, for any $x \le 1$, $x\lambda$ fraction of every demand can be concurrently routed over G, with at most xL load for every link. Consider the same flow over M_K. Each edge in M_K has at least $(1 - x)L$ free capacity, and every edge in $G - M_K$ has at most xL load. Thus, we can route $x\lambda$ fraction of each demand over M_K if we can reroute the load of the edges that are not in M_K, over the free capacity of the edges that are in M_K. In other words, we can satisfy $x\lambda$ fraction of each demand over M_K, if we have a max (concurrent) flow of at least 1 for the following multicommodity flow problem: a demand of xL corresponding to every link in $G - M_K$, to be routed over M_K, where each link in M_K has capacity $(1 - x)L$.

From Aumann and Rabani [4], we know that the max flow is $R/O(log(d))$ where R is the sparsity ratio and d is the number of demands. In our multicommodity flow problem, the number of demands is the number of edges in $G - M_K$, which we know from Lemma 6, is at most $\frac{|E|}{2^K}$. Sparsity ratio R is the minimum ratio over all cuts, of the capacity of the edges across the cut to the demands across the cut. Thus, in our multicommodity flow problem, R is the minimum over all cuts C of $\frac{e_C(1-x)L}{e'_C xL}$, where e_C is the number of edges of C that are also in M_K, e'_C is the number of edges of C that are also in $G - M_K$. (Recall that, the edges in e_C have free capacities of $(1 - x)L$ and the edges in e'_C have demands of xL.)

But we know from Corollary 4 that for any cut C, $\frac{e_C}{e'_C} \ge 2^K - 1$. Thus max flow is at least $\frac{(2^K-1)(1-x)L}{xL\ O(log(\frac{|E|}{2^K}))}$. Doing simple manipulations it follows that if $x = \Omega(\frac{2^K}{2^K + log(|E|)})$ then max flows is at least 1 (see Appendix VIII). Thus, $\Omega(\frac{2^K \lambda}{2^K + log(|E|)})$ fraction of every demand can be concurrently routed over M_K. ■

Corollary 8: Suppose that at least λ fraction of every demand can be concurrently routed over G. Then, if $K = log(log|E|)$ then $\Omega(\lambda)$ fraction of each demand can be concurrently routed over M_K. ■

Theorem 7 gives a worst-case guarantee on the max flow when the bi-partite graphs are obtained using max-cut. Although computing a max-cut is NP-hard in general graphs, it can be computed in polynomial time for planar graphs [12]. Thus, the above theorem gives a guarantee on max flow for planar graphs. Even for general graphs, in Section VI, we show through numerical simulations on randomly generated graphs that LS-cuts come close to meeting the above guarantee.

Asymmetric Demands. A routing for max flow with asymmetric demands may result in different loads for the two directions of a link, and therefore, the routing may require different fractions for links in the same bi-partite graph. We extend our approach for symmetric demands for this case.

Initial Assignment of Fractions. As in the symmetric case, using LS-cut, we obtain a subgraph LS_K that is the union of K bi-partite subgraphs $B_1, \ldots, B_K$, and then compute the routing for max flow. Since demands are asymmetric, the resulting routing may require that the links in the same subgraph have different fractions. To satisfy R2, we now need to assign a single fraction f_k to each bi-partite subgraph B_k, where every link $e \in B_k$ is assigned fraction f_k. This in turn, may decrease routed flows, and hence, reduce max flow. To limit this decrease in max flow, we try to minimize the *total mismatch* of LS_K, where the mismatch of a link is the absolute difference between the fraction required by a link (which is given by the routing) and the fraction that we assign to the bi-partite subgraph that contains the link.

Reducing Total Mismatch. An obvious way to reduce the total mismatch is to move links from one bi-partite subgraph to another such that links in the same bi-partite subgraph have fractions close to each other. However, while moving links we need to ensure that each of the subgraphs remains bi-partite. This problem of selecting bi-partite subgraphs so as to minimize the total mismatch has been shown to be NP-hard [15]. We thus give a heuristic to minimize the total mismatch.

Link Intervals. Before discussing the heuristic, let us take a closer look at the problem. Consider a link (u, v) in a bi-partite subgraph B_k, where $u \in B_k^0$ and $v \in B_k^1$. Let the load for the routing be xL from u to v, and yL in the reverse direction. (Obviously, $x, y \geq 0$, and $x + y \leq 1$.) For every f such that $x \leq f \leq 1 - y$, assigning a fraction f to B_k gives a mismatch of 0 at (u, v). This is because any fraction above x satisfies the requirement from u to v, and any fraction below $1 - y$ satisfies the requirement in the reverse direction. Thus, given the loads for both directions of a link e, we can define a link interval $I_e = [f_1, f_2]$ for link e such that, if the bi-partite subgraph containing e is assigned any fraction in I_e then the capacities required by e are satisfied in both directions, i.e., e has no mismatch. So, we redefine the mismatch for a link e as the distance of the link interval I of e from the fraction that is assigned to the bi-partite graph containing e. (The distance between a fraction f and an interval I is defined as $min_{f_1 \in I}|f - f_1|$.) Note that, if the sum of the loads for the two directions of a link is equal to its capacity L, then the corresponding link interval is simply a single fraction value.

Median Intervals. Next consider all links in a bi-partite subgraph B_k. Clearly, if intervals of all links in B_k have a non-empty intersection, then assigning any fraction in that intersection to B_k results in a total mismatch of 0 for B_k. However, if there is no such intersection, we would like to assign a fraction to the bi-partite subgraph such that the total mismatch is minimized. We find such a fraction as follows.

For a fraction f and an interval $I = [f_1, f_2]$, we say that I is lower than f if $f_2 \leq f$ and I is higher than f if $f \leq f_1$. The *median interval* is the set of fractions f such that the number of links with intervals lower than f is equal to the number of links with intervals higher than f.[5] The definition of the median interval suggests an obvious algorithm to find such an interval: in ascending order, sort the list of fractions that are either the start fraction or the end fraction of the link intervals. In this sorted list, let f_a be the first fraction such that the number of end fractions lower than or equal to f_a is equal to the number of start fractions higher than or equal to f_a. Also, let f_b be the next fraction in the sorted list. Then the median interval is $[f_a, f_b]$.

For each bi-partite subgraph B_k, we denote the median interval by MI_k. We now show that any fraction in the *median interval* MI_k minimizes the total mismatch for B_k. Consider any fraction $f \in MI_k$, and any other fraction $f' < f$. (The argument for $f' > f$ is symmetric.) Let E_1 be the set of links with intervals higher than f, E_2 be the set of links with intervals lower than f, and let E_3 be the set of links with intervals that contain f. Also, let $m(e)$ and $m'(e)$ denote the mismatch of link e when f and f' are assigned as fractions to B_k, respectively. Now, for each $e \in E_3$, $m(e) = 0$ (because the interval of e contains f), and $m'(e) \geq 0$. Also, for each $e \in E_1$, $m'(e) = m(e) + (f - f')$, and for each $e \in E_2$, $m'(e) \geq m(e) - (f - f')$. As f is in the median interval, we have $|E_1| = |E_2|$. Thus, the total mismatch $\sum_e m(e)$ when f is assigned to B_k is at most the total mismatch $\sum_e m'(e)$ when f' is assigned to B_k.

A Greedy Heuristic. Now consider the problem of reducing the total mismatch of LS_K. We use a simple greedy heuristic to reduce the total mismatch. We iteratively do the following. For each bi-partite subgraph B_k, we calculate the median interval MI_k, and define the cost of a link $e \in B_k$ as the mismatch of e when the midpoint of MI_k is assigned as the fraction of B_k.[6] In addition, we define the cost of a bi-partite subgraph as sum of the costs of all its links, and the cost of LS_K as the sum of costs of all K bi-partite subgraphs. We then select q links with the q highest costs and look at all K^q possible assignments of these links to the K bi-partite subgraphs (where q is a small positive integer). We call an assignment valid if each B_k remains bi-partite after the assignment. We then choose a valid assignment with the lowest total cost and update the

[5]This set is actually an interval because if two fractions f_1 and f_2 are in this set then all fractions between f_1 and f_2 also belong to the set.

[6]The total mismatch of B_k is the same for any fraction in MI_k.

bi-partite subgraphs accordingly. We repeat the above steps with updated bi-partite subgraphs if the reduction in cost of LS_K is larger than some constant ϵ. Otherwise, we terminate our algorithm by assigning each bi-partite subgraph B_k the fraction f_k corresponding to the midpoint of MI_k. Each link $e \in B_k$ is assigned a capacity of $f_k L$ from B_k^0 to B_k^1 and $(1 - f_k)L$ in the reverse direction.

VI. Numerical simulations

In this section we carry out extensive numerical simulations to evaluate our techniques for decomposing the input graph into K-bi-partite graphs and clustering links with similar demands into the same bi-partite graphs. For our simulations, we generate synthetic topologies that aim to match the geographical structure of village clusters. We now describe our simulation setup.

Generating synthetic graph topologies. We consider a circular plane with a radius of 50Kms. We iteratively place nodes on this plane. We assume that a link can exist between any two nodes only if they are within 10Kms of each other. We choose node positions randomly, with a restriction – that a node be within range of at least 1 node already placed on the plane. This is to ensure that we get a connected graph. For each node we compute the number of its candidate neighbors (i.e., nodes within a 10km radius of the node). We consider nodes in decreasing order of the number of their candidate neighbors, and select 1 or 2 nodes with the greatest number of candidate neighbors to be *gateway* nodes. We also fix the maximum degree (Δ) of the nodes in the graph. For every node, we then randomly select and create a link with at most Δ other nodes within a 10km radius. The sum of the capacities of both directions of a link is fixed at 11Mbps (the maximum capacity of an 802.11b channel). In our simulations we cover two main aspects of this work. First, assuming symmetric demands, we compare the max flow on the bi-partite subgraphs obtained by LS algorithm with the analytical worst-case bound for bi-partite subgraphs obtained by max-cut. Second, for asymmetric demands, we evaluate the performance of our heuristic for clustering links with similar fractions.

Max flow with symmetric demands. In Section IV, we had analytically demonstrated that the max flow over the union of K max-cuts is within $\Omega(\frac{2^K}{2^K+log(|E|)})$ of the max flow over the entire graph. However, since max-cut is, in general, NP-Hard to compute, we had also considered the LS algorithm to obtain the cuts (which is a $1/2$ approximation to max-cut). We now evaluate the performance of the LS algorithm in terms of the max flow, computed over the union of K-cuts determined using this algorithm.

For our simulations we create 25 instances of graphs with $N = 75$ nodes. We consider that all nodes have symmetric demands of 8Mbps both to and from the gateway nodes (initially we assume only 1 gateway node). We compute the max flow,[7] λ_1, over the *entire* input graph, G. Next, we recursively apply the LS algorithm on the graph G for values of $K = 1, 2, ..., 11$ to identify K bi-partite subgraphs within G. We then compute λ_2, the max flow over the union of these bi-partite subgraphs. We are interested in the ratio $\frac{\lambda_2}{\lambda_1}$. We compute the mean and the standard deviation of this ratio over all graph instances G and plot it in Figure 4 for different values of K.[8]

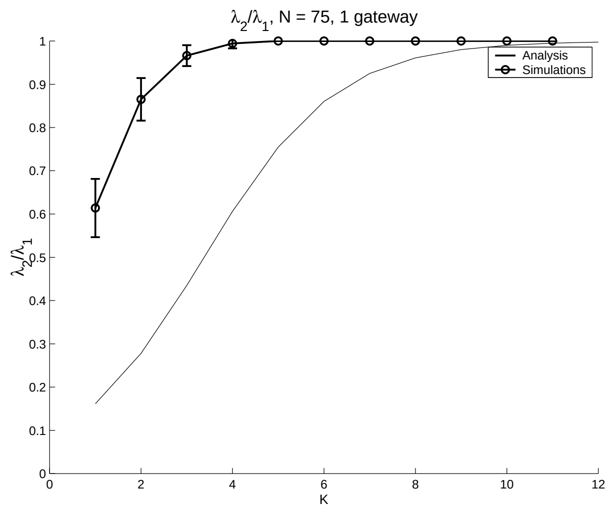

Fig. 4. Ratio of λ_2/λ_1 for different values of K. Comparing the max flow from the LS algorithm with the flow predicted analytically, if max cuts had been used instead.

Previously, our analysis has demonstrated that if each of the K-cuts are max-cuts then the worst-case value of λ_2/λ_1 ought to be $\Omega(\frac{2^K}{2^K+log(|E|)})$. We also plot the value of this analytical expression for different values of K. (On average, the number of edges $|E| = 1345$ for these graph instances.) It is clear that the max flow achieved over the union of K-cuts computed using our LS algorithm always meets and exceeds the predicted worst-case performance if we had used max-cuts instead. (In our simulations the average value of λ_1 was 0.33.) Specifically, we observe that for $K = 3$, $\frac{\lambda_2}{\lambda_1}$ is already > 0.95.

We have also observed a similar trend for variations of this experiment with different node demands, and also for graph instances with two gateway nodes.

Max flow over cuts with asymmetric demands. Recall from Section II that, the link scheduling protocol that we use (2P) requires all links within a bi-partite graph to be activated for an equal fraction of time in any given direction. In the previous simulation, we assumed that all nodes have symmetric demands. This ensures that there is a routing that achieves max flow such that it assigns equal loads to both directions

[7]We formulate the max flow problem as an LP using AMPL [11] and solve it with the CPLEX LP-solver.

[8]Since, we want to evaluate the performance of the LS algorithm (vs. our analysis) for different values of K, we allow the nodes in these graphs to have a maximum degree of 36. This ensures that close to $K = 6$ cuts will be required for the LS algorithm to cover the entire graph.

of each link. Hence, for all links to carry the load assigned to them, they have to simply be activated for an equal fraction of time (0.5) in both directions.

However, if nodes have *asymmetric* demands, max flow routing can result in different loads on two directions of a link. Also then, the fraction of time a link is required to be activated in any given direction can differ across links within the same bi-partite graph. Since the fraction of time a link is active in any given direction is the same for all links within a bi-partite graph, this mismatch may cause a link to not be able to carry the load that was assigned to it by max flow. To deal with this problem, in Section V, we described a heuristic that can re-arrange links across bi-partite graphs to minimize the mismatch. We now evaluate this heuristic.

We generate 50 instances of the mesh graphs with $N = 50$ nodes. We fix the maximum degree a node can have to be 5 (the average number of edges $|E| = 124$). We assume that the nodes have an asymmetric demand of 2Mbps to 1 selected gateway node, and 10Mbps from the gateway node. (We set ϵ to 0.1. Recall that the heuristic stops when the reduction in cost between two iterations is less than or equal to ϵ.)

As before, we first compute λ_1 on the original graph instance G. Then we run the greedy heuristic for different values of the parameter $q = 0, 1, ..., 5$ (as described in Section V) to rearrange links with similar intervals into the same bi-partite subgraphs. Recall, that the parameter q determines the number of links that are considered together to be reassigned to different bi-partite graphs. We then fix the fraction for each bi-partite graph (as would be the case when the 2P scheduling protocol is run on this graph), by selecting the interval that will minimize the mismatch cost across the graph. This fraction then determines the available capacity on the links. We use this modified capacity to compute the max flow, λ_2, over the union of bi-partite subgraphs. As before, we are interested in the ratio $\frac{\lambda_2}{\lambda_1}$. We compute the mean and standard deviation of this ratio over all graph instances and plot it in Figure 5 for different values of q. Larger values of q will allow the heuristic to evaluate more links together to decide which bi-partite subgraph they should be placed in.

In our experiments the mean value of λ_1 over all graph instances is 0.09. As expected, the performance of the heuristic improves for higher values of q. Infact, for $q = 5$ we get a very high mean value (0.97) of the ratio $\frac{\lambda_2}{\lambda_1}$. We also observe that the standard deviation of this ratio decreases with increasing q.

A novel aspect of our approach is that we consider a (desired) *interval* associated with each direction of the link. If a link is assigned an operating fraction anywhere within this interval, it will be able to carry the entire load assigned to it by max flow. We believe this interval based approach gives greater flexibility in minimizing the mismatch when we assign links to bi-partite subgraphs.

In contrast, the heuristics proposed in [15] only consider the fixed desired *fraction* of a link. Their heuristics then aim to assign links to bi-partite subgraphs (based on local search on top of an edge-coloring algorithm) so that the operating fraction of the selected bi-partite graph is closest to the desired fraction of the link. We compare our *intervals* based heuristic to the *fixed fraction* local search approach proposed in [15]. Our simulations (Table I, using the same set of parameters and input graphs as in the previous simulation) find that for the graph instances with 1 gateway node the Intervals based approach performs, on average, 15% better than the fixed fraction approach, and 12% better when there are 2 gateway nodes. We have also carried out this comparison for sparser graphs (similar to those used in [15]) and have observed similar results.

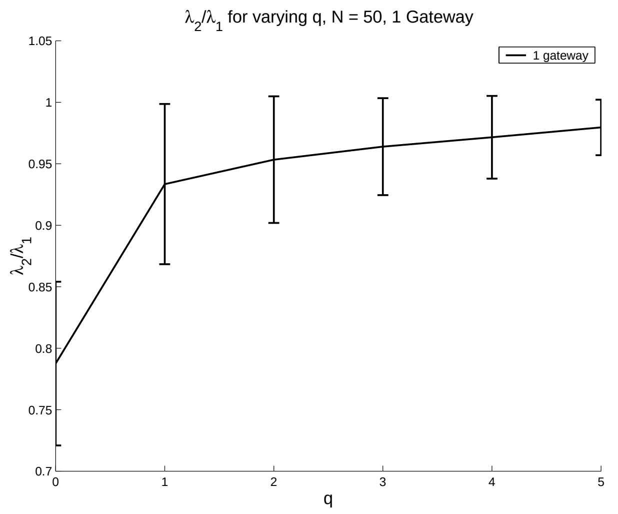

Fig. 5. Ratio of λ_2/λ_1 for different values of q: Evaluating the performance of the intervals based greedy heuristic.

(λ_2/λ_1), Mean (std. dev.)	1 gateway	2 gateways
Interval heuristic	0.98 (0.02)	0.94 (0.05)
Fixed fraction LS	0.85 (0.06)	0.84 (0.07)

TABLE I
MEAN AND STANDARD DEVIATION OF $\frac{\lambda_2}{\lambda_1}$ FOR THE INTERVALS VS. FIXED-FRACTION BASED HEURISTICS

To summarize, through extensive simulations we have demonstrated that our proposed approaches perform well within the analytical predictions and also come close to the best possible flow in our graph instances.

VII. Conclusions and Future Work

In this paper we have proposed channel allocation and routing algorithms for long-distance IEEE 802.11 based wireless mesh networks. Our work builds on 2P, a recently proposed MAC protocol that is well-suited for the interference pattern found within these networks. Given that 2P has the restriction that it can only operate on bi-partite graphs, we have made the following contributions in this work.

Given K non-interfering 802.11 channels, we proposed a cut-based algorithm to compute K bi-partite subgraphs (on each of which the 2P protocol can be run separately). We showed that a large class of graphs can thus be completely covered by K bi-partite subgraphs. For the remaining set of graphs, we analytically established what fraction of the max flow of the original mesh graph can be routed over the bi-partite subgraphs.

For a bi-partite graph, the 2P protocol also requires that the links are active for equal fraction of time in a given direction, and thus, restricts the capacities of links in a given direction. Therefore, we also studied the problem of mismatch between the load on a link (as computed by max flow) and it's effective capacity under a given channel allocation. To reduce this mismatch, we considered the interval of a link, i.e., the set of fractions of time for which the link can be scheduled in a given direction without compromising on it's assigned load. To limit the decrease in max flow due to this restriction on link capacities, we proposed a simple heuristic based on clustering links with similar intervals into the same bi-partite graphs. We showed through comprehensive numerical simulations that our heuristic comes very close to the best possible flow.

There are several interesting future directions for this work. In this paper we have assumed that adjacent links have sufficient angular separation to allow SynOP. This may not be true in practise. We have also assumed that nodes are connected only through point-to-point directional antennas. An alternate model, to reduce cost, would be that a node connects to several other nodes through one *sectoral* antenna. Sectoral antennas again introduce further spatial interference problems both between each other and with other directional links that overlap with them. Hence, a new distributed link scheduling protocol may be required that can allow spatially overlapping links to operate without interference.

We have also observed that Mix-Rx-Tx interference implies that only a bi-partite subgraph can be active on any given channel. We would also like to explore alternate distributed scheduling strategies such as, for example, switch across multiple bi-partite subgraphs that share an 802.11 channel.

REFERENCES

[1] R. Ahuja, T. Magnanti, and J. Orlin. *Network Flows - Theory, Algorithms and Applications*. Prentice-Hall, 1993.

[2] I. F. Akyildiz, X. Wang, and W. Wang. Wireless mesh networks: a survey. *Computer Networks*, 47:445–487, 2005.

[3] M. Alichery, R. Bhatia, and L. Li. Joint channel assignment and routing for throughput optimization in multi-radio wireless mesh networks. In *Proceedings of MobiCom*, 2005.

[4] Y. Aumann and Y. Rabani. An O(log k) approximate min-cut max-flow theorem and approximation algorithm. *SIAM J. Computing*, 27(1), 1998.

[5] P. Bahl, R. Chandra, and J. Dunagan. SSCH: Slotted seeded channel hopping for capacity improvement in ieee 802.11 ad hoc wireless networks. In *Proceedings of MobiCom*, 2004.

[6] P. Bhagwat, B. Raman, and D. Sanghi. Turning 802.11 inside-out. In *Second Workshop on Hot Topics in Networks (HotNets -II)*, 2003.

[7] J. Bicket, D. Aguayo, S. Biswas, and R. Morris. Architecture and evaluation of an unplanned 802.11b mesh network. In *Proceedings of MobiCom*, 2005.

[8] M. Burton. Channel overlap calculations for 802.11b networks. 2002. http://www.80211-news.com/publications/page289-376.asp.

[9] I. Chlamtac and S. Kutten. A spatial reuse tdma/fdma for mobile multi-hop radio networks. In *Proceedings of Infocom*, 1985.

[10] I. Chlamtac and S. Lerner. A link allocation protocol for mobile multi-hop radio networks. In *Proceedings of Globecom*, 1985.

[11] R. Fourer, D. M. Gay, and B. Kernighan. *AMPL A modeling language for mathematical programming*. Thomson, 2 edition, 2003.

[12] F. O. Hadlock. Finding a maximum cut of a planar graph in polynomial time. *SIAM Journal on Computing*, 4:221–225, 1975.

[13] M. Kodialam and T. Nandagopal. Characterizing achievable rates in multi-hop wireless mesh networks: the joint routing and scheduling problem. In *Proceedings of MobiCom*, 2003.

[14] P. Kyasanur and N. Vaidya. Capacity of multi-channel wireless mesh networks. In *Proceedings of MobiCom*, 2005.

[15] B. Raman. Channel allocation in 802.11-based mesh networks. In *Proceedings of IEEE Infocom*, Apr. 2006.

[16] B. Raman and K. Chebrolu. Design and evaluation of a new MAC for long distance 802.11 mesh networks. In *Proceedings of ACM Mobicom*, Aug. 2005.

[17] A. Raniwala and T.-C. Chiueh. Architecture and design for an IEEE 802.11-based multi channel wireless mesh network. In *Proceedings of Infocom*, 2005.

[18] J. So and N. Vaidya. Multi-channel MAC for ad hoc networks: handling multi-channel hidden terminals using a single tranceiver. In *Proceedings of Mobihoc*, 2004.

[19] R. van Drunnen, J. Koolhaas, H. Schuurmans, and M. Vijn. Building a wireless community network in the netherlands. In *Proceedings of Usenix/Freenix conference*, 2003.

[20] V. Vazirani. *Approximation Algorithms*. Addison-Wesley, 2001.

VIII. APPENDIX: A DETAIL IN THE PROOF OF THEOREM 7

In the proof of Theorem 7, suppose max flow is at least $\frac{(2^K-1)(1-x)L}{xL\ O(log(\frac{|E|}{2^K}))}$. We now show that, if $x = \Omega(\frac{2^K}{2^K+log(|E|)})$, then max flows is at least 1. (We will assume $|E| \geq 2^K$, otherwise, M_K anyway covers G.)

Let a denote $(\frac{|E|}{2^K})$. Then, there exists positive constants a_0 and c such that max flow is at least $\frac{(2^K-1)(1-x)L}{xLc.log(a)}$ when $a \geq a_0$. It follows that if $x = x_0 = \frac{2^K-1}{(2^K-1)+c.log(a)}$ then max flows is at least 1.

We consider two cases $c < 1$ and $c \geq 1$. Consider the first case. Then, $x_0 = \frac{2^K-1}{(2^K-1)+c.log(a)} \geq \frac{2^K-1}{(2^K-1)+log(a)} \geq \frac{2^K-1}{2^K+log(|E|)-(1+K)} \geq \frac{2^K-1}{2^K+log(|E|)} \geq \frac{2^K}{2(2^K+log(|E|))}$.

Suppose $c \geq 1$. Then, $x_0 = \frac{2^K-1}{(2^K-1)+c.log(a)} \geq \frac{2^K-1}{c(2^K-1)+c.log(a)} \geq \frac{2^K-1}{c(2^K+log(|E|))-c(1+K)} \geq \frac{2^K-1}{c(2^K+log(|E|))} \geq \frac{2^K}{2c(2^K+log(|E|))}$.

Thus, if $x = \Omega(\frac{2^K}{2^K+log(|E|)})$ then max flows is at least 1.