Channel Allocation in 802.11-based Mesh Networks

Bhaskaran Raman
Department of CSE, IIT Kanpur, INDIA 208016
braman AT cse DOT iitk DOT ac DOT in

***Abstract*— IEEE 802.11 (WiFi) has been used beyond its original intended purpose of a tether-free LAN. In this paper, we are interested in the use of 802.11 in mesh networks. Specifically, we consider those which involve directional antennas and long-distance point-to-point links. In recent work, the 2P MAC protocol has been designed to suit such a network architecture. In this paper, we assume the use of the 2P MAC protocol in the links of the network, and consider the problem of link channel allocation. We first formulate the problem of minimizing the mismatch between link capacities desired by the network operator and that achieved under a channel allocation. We show that this problem is NP-hard. We then explore several heuristics for channel allocation and find a set of heuristics that achieve the optimal allocation in most scenarios.**

I. Introduction

IEEE 802.11 [1] (WiFi) was originally designed as a flexible and cost-effective extension to ethernet-based LANs. Although such operation is still the dominant use of WiFi, the popularity and low-cost of the technology has motivated its use in several other scenarios. In this paper, we are interested in its use in *long-distance mesh networks*, with *point-to-point links* [2], [3], [4], [5]. Such mesh networks are used, or are being planned, in developed countries [5] as well as developing countries [2], [3] alike, to provide low-cost broadband Internet access to remote rural locations. The long-distance (up to 40km or more) point-to-point links are formed using high-gain directional antennae [6].

An important issue in such use of 802.11 is that the CSMA/CA-based MAC protocol is not suited for the point-to-point links. Our prior work [7], [8] describes a TDMA-style MAC protocol called 2P, which utilizes the links in the network optimally. 2P assumes a network model where each node in the network has multiple radios, one for each point-to-point link at the node. This is shown in Fig. 1. The figure also shows a *landline* node which acts as the node which provides Internet connectivity to the rest of the nodes. 2P operates by switching each node in the network between Tx and Rx phases (two phases, hence the name 2P) alternatively. In such a scheme, the entire network can operate in a single channel, while still keeping each link active in either direction at any given time.

The above model of 2P operation has an important unresolved issue. 2P as described in [8] is agnostic about how the capacity in a (half-duplex) link is apportioned across the two directions of the link. In [8], for simplicity, we simply apportion the available capacity equally in either direction.

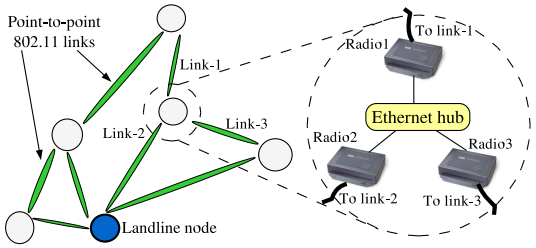

Fig. 1. 802.11 mesh: nodes have one radio per link

Now, apportioning the link capacity equally in either direction is clearly inappropriate. This is especially so since in access networks we expect the download requirements to be significantly higher than the upload requirements. In this paper, we address this important issue. We stipulate a model where the network operator has the freedom of dividing up a link's limited capacity in a flexible manner in either direction. Each link in the network can have an arbitrary *desired* capacity allocated for a given direction.

In the process of addressing the above issue, we also do away with another assumption made by 2P. The 2P MAC protocol assumes that the network topology is bipartite. In this paper, we consider an arbitrary network topology. We use multiple channels to divide the network into *channel subgraphs*. A channel subgraph is a contiguous part of the network using the same channel. 2P is used independently within each channel subgraph.

We consider the problem of allocation of channels to the links in the network. We view this problem as one of breaking up the network into channel subgraphs. In this process, our objective is two-fold. First, the channel subgraphs should be bipartite, so that 2P can operate within each channel subgraph. Next, more importantly, the channel allocation should be such that the *achieved* link capacity is as close as possible to the desired capacity.

We show that even for a simple case, finding an optimal channel allocation that minimizes the mismatch between desired and achieved link capacities is NP-hard. We then propose and evaluate several possible heuristics for channel allocation. We find that a simple set of heuristics can achieve optimal channel allocation in most scenarios.

The key contributions of this paper are thus: (a) formulation of the channel allocation problem to minimize the mismatch between desired and achieved link capacities, and (b) heuris-

tics for solving it optimally. The rest of the paper is organized as follows. The next section (Sec. II) presents a brief overview of the 2P MAC protocol. In Sec. III we present the overall system model, and describe the channel allocation scheme in relation to the 2P MAC. Subsequently, in Sec. IV and Sec. V we consider the issue of channel allocation to best fit the desired link capacities. Sec. VI discusses prior work in related settings. Finally, we present several open issues and future directions in Sec. VII and conclude in Sec. VIII.

II. Background: The 2P MAC Protocol

We now briefly describe the relevant details of the 2P MAC protocol detailed in [7], [8]. The 2P MAC is a TDMA-style protocol for mesh networks. It considers a network where each node has multiple point-to-point links, each with a separate directional antenna, as shown in Fig. 1.

Although the links use directional antennae, the links at a node cannot really operate independently due to the presence of side-lobes. That is, when a node is transmitting along a link, it cannot simultaneously receive along another link, since the reception will face interference from the transmitting radio at that node.

However, it is possible to have *synchronous operation (SynOp)* where the links at a node are all transmitting *(SynTx)*, or all receiving *(SynRx)*. In 2P, each node in the network simply switches between these two phases. When a node is in SynTx, its neighbours are in SynRx, and vice-versa. Further, when a node switches from SynRx to SynTx, its neighbours switch from SynTx to SynRx, and vice-versa. This is illustrated in a simple example in Fig. 2 for a 4-node topology. Transmissions are shown above the time-line and receptions are shown below the time-line for the four links in the topology.

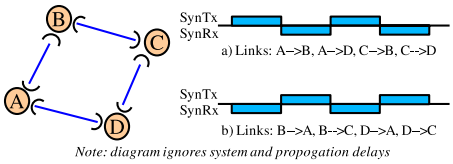

Fig. 2. 2P Illustration

2P achieves maximal efficiency by operating each link in one direction or another at all times. It effectively makes the half-duplex wireless point-to-point link behave like a wired link, by eliminating all contention. Reference [8] also describes how 2P can operate without tight time synchronization. This detail is not important for our discussion, and we skip this here.

For the purposes of the rest of our discussion, we note the following. First, 2P operation requires that the topology be bipartite. Next, in 2P, the capacity of a link is divided across the two directions of a link. 2P itself is agnostic with respect to how the apportioning happens, albeit with the following conditions. The outgoing links from a node all have the same capacity, say a fraction f of the total capacity of the link. The incoming links all have the same capacity, and it is $1 - f$.

III. Overall System Design

Our design combines the 2P MAC protocol and a channel allocation scheme. We now present the system design and the rationale behind our approach.

2P operation in a bipartite graph (with all links using the same channel) is illustrated in Fig. 3. (V_1, V_2) is the bipartition of the nodes. We alternate between scheduling traffic along edges in the direction $V_1 \rightarrow V_2$ for a fraction f of the time, and in the direction $V_1 \leftarrow V_2$ for a fraction $1 - f$ of the time.

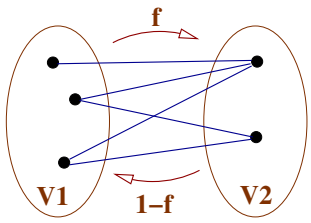

Fig. 3. Two-phase scheduling in a bipartite graph, or in a "channel subgraph"

Now, 802.11b/g defines at least 11 channels, as depicted in Fig. 4. Of these, at least three are completely non-overlapping (channels 1, 6, and 11). Now, the issue of medium contention arises only when we are using the same channel or overlapping channels. An important observation in our design is that if two links are allocated independent (non-overlapping) channels, they can be scheduled independent of one another.

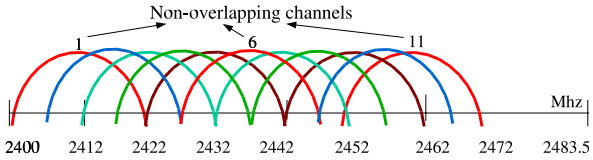

Fig. 4. 802.11 channels: schematic

For operation, each of the links in the network must be assigned a particular channel. The pair of transceivers for a particular link are tuned to the same channel for transmission in either direction (i.e., there is no directionality in channel assignment). And for our purposes, we consider the three mutually non-overlapping channels 1, 6, and 11. It may be possible to squeeze a fourth channel as explained in [9]. However, we reserve this for local 802.11b/g access at a node (e.g. local access within a village).

We combine channel allocation with 2P transmission scheduling as follows. We define a *channel subgraph* to be a maximally connected subgraph (of the original network graph) where all the edges are allocated the same channel. A particular channel allocation for the links in the network graph results in a partitioning of the links (or equivalently edges) into various channel subgraphs. An example is shown

in Fig. 5. The different channel subgraphs are marked with different line patterns and their channel allocations are shown alongside. Note that two edges allocated the same channel need not be in the same channel subgraph – they may be separated by links of other channels.

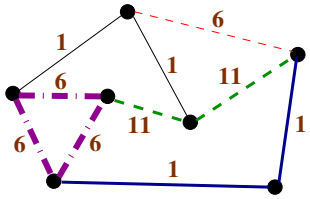

Fig. 5. Channel subgraphs: an example

Based on this definition, we observe that we only need to worry about transmission scheduling within channel subgraphs. Scheduling across channel subgraphs can be independent of each other.

Combining the observations above, even if the given network graph is not bipartite, we can break it up into smaller channel subgraphs that are bipartite. And the 2P MAC protocol is possible in the entire graph if we can allocate channels to the original network graph such that all the channel subgraphs are bipartite.

Define a channel allocation that results in bipartite (BP) channel subgraphs to be a *BP-proper* channel allocation. We also call this a *BP-proper* edge-colouring since channel allocation is essentially graph edge-colouring (we use the terms channel-allocation and edge-colouring interchangeably). Our example in Fig. 5 is not a BP-proper channel allocation, although a BP-proper 3-edge-colouring does exist for the graph. The question now is whether a given original network graph has a BP-proper 3-edge-colouring; we consider three colours since we have three non-overlapping channels in 802.11.

Clearly, not all graphs have a BP-proper 3-edge-colouring (K_9, a complete graph on 9 nodes, is an example[1]). Instead of trying to characterize the set of graphs that have a BP-proper 3-edge-colouring, we focus on a simple class of graphs that do have this property. In our design, we choose this class to be the set of graphs for which a proper 6-edge-colouring exists. (A proper edge-colouring is one in which any pair of adjacent edges have different colours).

The reason behind this design decision is that if a graph is 6-edge-colourable, then it has a BP-proper 3-edge-colouring. We give a constructive proof for this. A simple algorithm to arrive at a BP-proper 3-edge-colouring given a proper 6-edge-colourable is as follows. First colour the edges with C_0, C_1, C_2, C_3, C_4, and C_5. Next, merge the colours in pairs, say $C_0\&C_3$, $C_1\&C_4$, and $C_2\&C_5$, resulting in a 3-edge-colouring. This 3-edge-colouring is a BP-proper colouring. In fact, each channel (or colour) subgraph is either a path or an even-cycle, and in either case bipartite. This is because, in the edges of each channel subgraph, the merged colours C_i and C_{i+3} ($i = 0, 1,$ or 2) must alternate in the original 6-edge-colouring of the graph.

We consider an even restricted class of network graphs to ensure that it is 6-edge-colourable. We consider graphs with $\Delta \leq 5$, where Δ is the maximum node degree in the graph. By Vizing's theorem [10], a graph with $\Delta \leq 5$ definitely has a proper 6-edge-colouring.

The design decision of choosing this constraint is driven by the following reason. With this constraint, there is a known algorithmic mechanism to arrive at a BP-proper 3-edge colouring. This consists of two steps: (1) proper 6-edge-colouring the graph using Vizing's algorithm [10], and (2) merging colours in pairs, as described above, to result in a BP-proper 3-edge-colouring.

We believe that the constraint of $\Delta \leq 5$ is not restrictive in our setting. Since we are working in a mesh network setting, it is unlikely that a node will have degree over 5. In fact, a degree of 2 or 3 at a node is expected to be the common case. This is sufficient to create a useful mesh network since the nodes are spread out geographically. It is only for nodes close to or at the landline access points are we likely to have relatively high connectivity and thus higher node degrees. Hence we make the simplifying assumption that $\Delta \leq 5$ for the network graph. This assumption buys us a simple, efficient scheduling mechanism based on 2P operating throughout the network. Given this, future 802.11 mesh networks can be constructed under this (rather unrestrictive) constraint.

So far we have discussed a channel allocation mechanism in relation to the 2P MAC scheme. We now consider channel allocation in combination with link capacities.

IV. Channel Allocation

In the long-distance 802.11b network, suppose that we engineer the power levels to achieve 11Mbps on all the links. This 11Mbps represents the raw bandwidth that can be achieved in both the directions combined. Now, in a wired ISP network, the various links are provisioned (incrementally) to suit the expected traffic on them. However, there is an upper bound (11Mbps for 802.11b) on the achievable link capacity for our 802.11 network operator. Even this 11Mbps has to be shared between either direction.

Now, under the scheduling and channel allocation scheme described in the previous section, it is straightforward to achieve 5.5Mbps (half the 11Mbps raw bandwidth) for all the links in each direction. However, since one of the intended uses of the 802.11 mesh network is a wireless access network, there will likely be asymmetry in traffic in different directions. For instance, traffic to a village node (e.g., HTTP traffic) may be significantly higher than traffic from it. Similarly, traffic in the direction towards a landline node may be lower than the traffic from it. This motivates us to consider a model where we give

[1]To see this, K_3 does not have a BP-proper 1-edge-colouring. Hence K_5 does not have a BP-proper 2-edge colouring. Suppose it does, then consider the bipartition into (V_1, V_2) produced by the edges of the first colour. At least one of the partitions must have three nodes, and hence the K_3 between them must be coloured the second colour, which is a contradiction. Arguing similarly, K_9 does not have a BP-proper 3-edge-colouring.

the network operator the flexibility of achieving a particular *split* of the total 11Mbps for either direction. The operator can thus specify a *desired fraction (DF)* f of the 11Mbps for one particular direction and the remaining fraction $1-f$ for the other direction of the link.

Under the above definition, there is directionality associated with the specification of DF for an edge. If an edge between nodes v_1 and v_2 has a DF f in the direction $v_1 \rightarrow v_2$, it has a DF $1-f$ in the direction $v_1 \leftarrow v_2$. Hence for an edge, exactly one of $DF(v_1 \rightarrow v_2)$ and $DF(v_1 \leftarrow v_2)$ needs to be specified, and the other is computed accordingly.

Now, channel allocation and link capacities are closely related as follows. Consider a particular channel allocation as described in the previous section, and a channel subgraph S_G under this allocation. Let (V_1, V_2) be the bipartition of S_G, as in Fig. 3. Suppose the set of edges in S_G is $S_E = \{e_1, e_2, ...e_k\}$. Represent the edge e_i in the direction $V_1 \rightarrow V_2$ as $\overrightarrow{e_i}$ and in the direction $V_1 \leftarrow V_2$ as $\overleftarrow{e_i}$. We now observe that in a two-phase transmission schedule, the fraction of time for which all $\overrightarrow{e_i}$'s are scheduled is the same, say f. And the fraction of time for which all $\overleftarrow{e_i}$'s are scheduled is thus $1-f$.

We term f to be the *achieved fraction (AF)* for each of the $\overrightarrow{e_i}$'s. The achieved fraction for the $\overleftarrow{e_i}$'s is $1-f$. Of course, one has the freedom to choose the f in the above example. However, unless the channel subgraph is such that all the $\overrightarrow{e_i}$'s have the same DF, for any choice of f for the subgraph, some edges will have $AF \neq DF$. Note that for an edge $\overrightarrow{e_i}$ if $AF > DF$ then for $\overleftarrow{e_i}$, $AF < DF$. That is, in one direction, the edge has achieved link capacity smaller than the desired value. We term $|AF - DF|$ to be the *mismatch* for an edge.

Intuitively, we would like to minimize the mismatch between AF and DF. Suppose that we are able to come up with a channel allocation such that for each channel subgraph, the desired fractions of all $\overrightarrow{e_i}$'s are the same, say f_1. We can then schedule traffic in $V_1 \rightarrow V_2$ for a fraction f_1 of the time, and there would be no mismatch between the DF and AF of the edges. We call this a *zero-mismatch* channel allocation (ZMCA). It groups edges into various channel subgraphs such that all the $\overrightarrow{e_i}$'s are the same.

The question now is whether such a channel allocation is possible. We also term the problem of determining if a graph has a zero-mismatch channel allocation as ZMCA. Of course we are interested in a 3-channel allocation and hence consider ZMCA with three edge colours.

NP-Completeness of ZMCA: We show that ZMCA is NP-Complete, even when we restrict: (a) the graph to have $\Delta \leq 4$, and (b) the desired fractions to be chosen from a set of only five distinct values, with four of these being two pairs of $(f, 1-f)$. Hence, the general problem without these restrictions is also NP-Complete. We show that ZMCA is NP-complete by reducing an arbitrary instance of 3SAT (the satisfiability problem with at most three literals per clause) to an instance of ZMCA. (ZMCA is clearly in NP). Our proof mimics that in [11], where it is shown that the problem of determining the edge-chromatic number of a three-regular graph is NP-Complete. We outline the proof below, and refer the reader to [11] for further details.

Outline of the Proof: We now prove that ZMCA as defined in Sec. IV is NP-Complete. Given an instance of 3SAT, we construct an instance of ZMCA, with the graph having $\Delta \leq 4$, and choosing DFs from a set of five distinct values. For ease of exposition, say we choose DFs from the set $\{\frac{1}{4}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{3}{4}\}$. Our proof mimics that in [11] by constructing the inverting component, the replicating component, and the clause testing component. In the figures showing the various components, the edges are marked in one of the directions. This direction corresponds to the one that has the *smaller* DF (among f and $1-f$). The edges with DF=$\frac{1}{2}$ are not given any direction. The edges have various thicknesses corresponding to their DFs. The legend describes this notation.

Like in [11], a "true" value is represented by two edges having the same colour, and a "false" value by them having different colours. And like in [11], the construction uses inverting components, replicating components, and clause-testing components. We now show the construction of these components in our context.

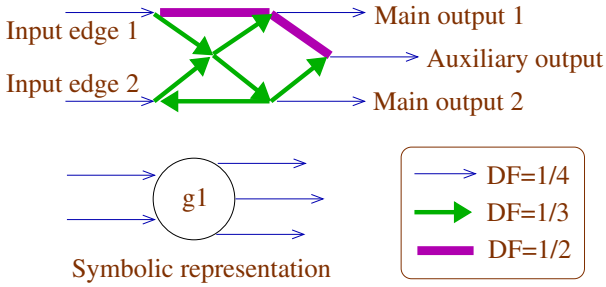

Fig. 6. Inverting component: version-1

Fig. 6 shows an inverting component, labeled as version-1, as explained below. Under any zero-mismatch 3-colouring the component behaves as an inverting component. To see this, observe that two adjacent edges can be given the same colour only if they have the same desired fraction in the same direction away from their common node. We leave it to the reader to convince himself/herself that under all possible BP-proper 3-colourings, this behaves as an inverting component.

As in [11], this component has two "input" edges, two "main output" edges, and an "auxiliary output" edge. This inverting component behaves almost the same as that in [11]. One difference is that when the input to the inverting component is "true" (the two input edges coloured the same), say C_0, this inverting component has the property that the auxiliary output edge is also forced to be C_0. In [11], there is a flexibility that the auxiliary edge can be any of C_0, C_1, or C_2 in such a scenario. It turns out that this flexibility is required for the clause-testing component. This is the reason why we labeled this inverting component as version-1. An inverting component with the required flexibility is created easily by concatenating five of the version-1 components as shown in Fig. 7.

The replicating component is constructed from several in-

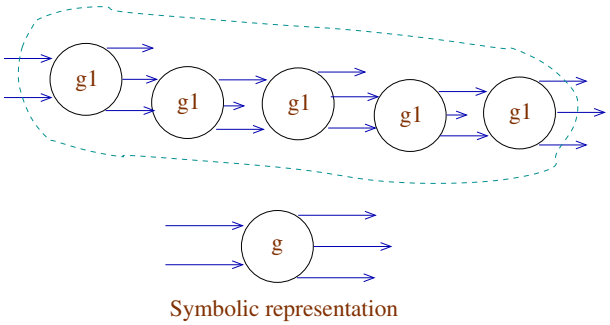

Fig. 7. Inverting component

verting components just as in [11]. However, for this to work, we need to observe that in our inverting component, the auxiliary output can be directed either way (i.e. its DF can be 1/4 or 3/4) – the behaviour of the inverting component remains the same irrespective.

The clause testing component construction is also similar to that in [11]. We however need to be careful about the choice of DFs in our construction. This is shown in Fig. 8.

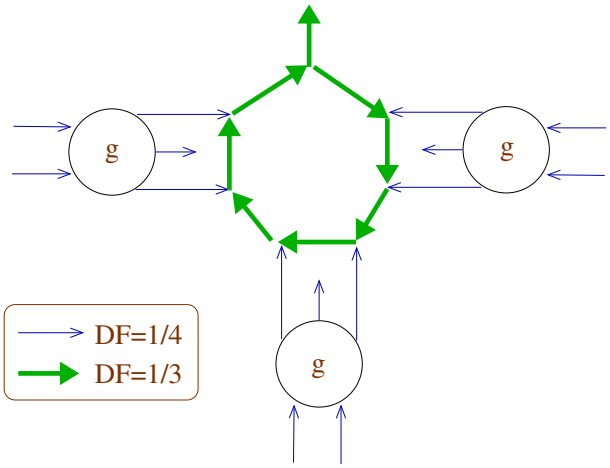

Fig. 8. Clause testing component

With such a construction, the original 3SAT instance is satisfiable if and only if the resulting construction has a ZMCA. Note that in our construction, the maximum node degree is 4.

V. Heuristics for Optimal Channel Allocation

We would ideally like to achieve zero-mismatch channel allocation (ZMCA) for a given graph. But a ZMCA may not even exist for the graph and the given set of DFs for the edges. Given this, we try to allocate channels such that the $\overrightarrow{e_i}$'s within each subgraph have the same DF "as far as possible". In more precise terms, we consider the additive metric of the sum of all mismatches in the graph under a channel allocation. We term the problem of finding a channel allocation that minimizes this metric as the *minimum-mismatch* channel allocation (MMCA) problem. Given that ZMCA is NP-Complete, the corresponding optimization version MMCA is NP-hard. We now explore heuristics for MMCA under the overall 2-step scheme (step-1: Vizing colouring, followed by step-2: colour merging in pairs) for finding a BP-proper channel allocation.

We also have a third step after the BP-proper channel allocation. This is the assignment of the fraction f to each channel subgraph, as shown earlier in Fig. 3. In ZMCA, this step was trivial since all we had to check was if all the $\overrightarrow{e_i}$'s had the same DF f_1, and if so assign the $\overrightarrow{e_i}$'s an AF of f_1. In MMCA too, this step is straightforward as we explain below.

Suppose the $\overrightarrow{e_i}$'s $(i = 1..k)$ of a channel subgraph have DFs $\{0 \leq f_1 \leq f_2 ... \leq f_k \leq 1\}$. The fraction assignment $\hat{f}$ that minimizes the cost in the subgraph $C = \sum_{i=1..k} |f - f_i|$ has to be one of the f_i's. This is because C is piece-wise linear in f in $[0, 1]$ – i.e., linear in each of $[0, f_1], [f_1, f_2], ...[f_k, 1]$. If $\hat{f}$ is in $[0, f_1]$ it has to be f_1, and if it is in $[f_k, 1]$, it has to be f_k. If it lies in $[f_i, f_{i+1}]$, then it has to be one of f_i or f_{i+1}.

We can thus find the f for the channel subgraph to be the f_i that minimizes C as above. The assignment of f for each subgraph can be performed independently, and this third step thus gives the AFs for the edges under a BP-proper 3-edge-colouring. We have the corresponding mismatch cost associated with the channel allocation.

For our heuristics, we consider the freedom offered to us in the first and second steps. In the second step of colour merging, the freedom we have is that we can merge any pairs of the six colours $C_0 - C_5$. We subsume this freedom in our first step of 6-edge-colouring as follows. We decide the colours that are going to be merged, say C_i & C_{i+3} $(i = 0, 1, 2)$, before we begin the 6-colouring (i.e. we restrict our freedom in the second step). However, we consider a swap of C_i with C_j, $i \neq j$, in all edges of the graph (after the 6-edge-colouring) to result in a different 6-edge-colouring. This effectively compensates for the fact that we do not consider all possible cases of colour merging. As an example, let us see why we do not need to consider the merging of say colours C_0&C_4. Such a case would be similar to the merging of C_0&C_3, after a swap of C_3 and C_4 in a 6-edge-colouring. We thus consider heuristics only in the first step: 6-edge-colouring using Vizing's algorithm.

Vizing's algorithm works as follows. It colours edges one after another, in each stage choosing a colour that is absent (so far) at either end-point v_1 and v_2 of the chosen edge e. If no such common unused colour between v_1 and v_2 is found, there is a simple recolouring process that is guaranteed to terminate. We use this Vizing's algorithm with the small modification that we always use 6 colours, even if the graph were colourable with less colours.

We consider two hooks in the Vizing algorithm for heuristics: (1) the choice of colour for an edge, when there is freedom to do so, and (2) the order in which edges are coloured. We explore heuristics based on these choices. We now present our evaluation methodology (Sec. V-A), followed by the heuristics that use the above two hooks (Sec. V-B& V-

C). We then present a local search heuristic in Sec. V-D that acts on top of the above heuristics.

A. Evaluation methodology for heuristics

We study the effectiveness of our heuristics by applying them to randomly generated graphs that are constructed to resemble expected long-distance 802.11 WiFi networks.

We first generate the nodes at random locations on a rectangular area (we choose 100km X 70.7km – 70.7 is 100/sqrt(2)). The number of nodes is a parameter in this procedure. For each node, we compute its "neighbouring density" as the number of nodes within a rectangle 40km X 28.3 km ($\frac{2}{5}$'ths of the overall dimensions). This roughly captures how close a node is to other nodes. Proceeding in order of increasing density value of the nodes, we designate a *desired node-degree* of 1 for the first 15% of nodes, node-degree of 2 for the next 35%, 3 for the next 35%, 4 for the next 10%, and 5 for the next 5%. The percentage values chosen above are meant to capture a realistic 802.11 long-distance network – majority of the nodes with degree 2 or 3, and some with degree 1, 4, or 5.

We next form a spanning tree T among the nodes. The purpose of this step is to ensure that we end up with a connected graph. We start with an empty set T and at each stage choose an edge e between T and $V - T$ such that e had the least physical distance among all such possibilities. We then add e to T, and repeat the process until T is a spanning tree. In the final stage, we add more edges to the tree T to result in a graph G. We start with nodes that are have a *desired node-degree* of 2 (since with the spanning tree all nodes have at least degree 1), and then choose nodes with desired node-degree 3, and so on. For each node, we satisfy its desired degree by choosing among its closest neighbours. The resulting graph may have a higher degree for a few nodes than originally desired. We however ensure that $\Delta \leq 5$.

After generating the graph, we also specify randomly chosen desired fractions (DFs) for each edge. The DFs are chosen from the set $\{\frac{1}{4}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{3}{4}\}$. This set allows the network operator reasonable flexibility for choosing the desired link capacities.

B. Heuristics for colour choice

There are two heuristics we consider for choice of colour while colouring an edge.

Greedy-Col: This first heuristic is a simple, greedy approach. While colouring an edge e with end-points v_1 and v_2, suppose that we have the freedom to choose from a set of colours (each unused so far at both v_1 and v_2). For each colour possible for e, we consider the subset S_E of edges coloured so far (including e). We then perform colour merging in S_E and find the cost of the channel subgraph that contains e. We then simply choose the colour that produces the minimum such cost. Thus at each stage of the Vizing colouring, we greedily try to pick a colour that would add the minimum mismatch cost to the graph.

Match-DF: Recall that in a ZMCA all the $\overrightarrow{e_i}$'s in a subgraph have the same DF. This heuristic, Match-DF, explicitly tries to achieve this. Suppose we merge colours C_i and C_{i+3} ($i = 0, 1,$ or 2, as in Sec. III) after the 6-colouring, we define C_i and C_{i+3} to be *counterpart* colours of each other. While colouring edge e between nodes v_1 and v_2, we give preference to a colour such that: (a) it is among the Greedy-Col colours as in the previous heuristic, and additionally (b) its counterpart colour is already among the set of coloured edges at v_1 and/or v_2, and importantly (c) the edge(s) with the counterpart colour at v_1 and/or v_2 have the same DF as $\overrightarrow{e}$ or $\overleftarrow{e}$, considered in the appropriate direction. (If no colours satisfying (b) and (c) exist, the fall-back would be Greedy-Col).

To explain this with an illustrative example, suppose e is directed $\overrightarrow{e}$ in the direction $v_1 \rightarrow v_2$, and v_1 has an already coloured edge $\overrightarrow{e_1}$ in the direction $v_1 \rightarrow v_3$. If $\overrightarrow{e_1}$ has the same DF as $\overrightarrow{e}$, then the counterpart colour of e_1 is preferred for colouring e. In doing such matching, we prefer colours for which we are able to match at both end points v_1 and v_2 over colours for which a match happens on only one of the end-points.

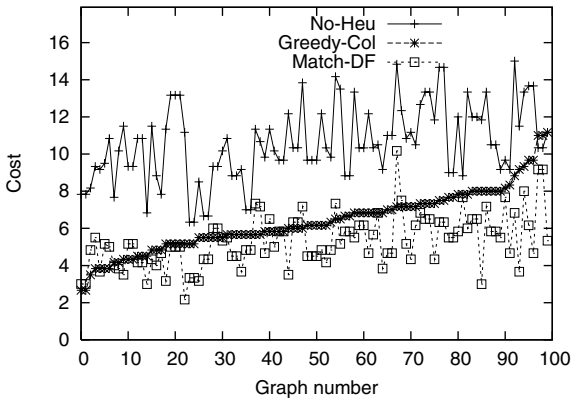

Fig. 9. Performance of Greedy-Col and Match-DF

Performance of Greedy-Col and Match-DF: To compare the performance of these heuristics, we generate 100 random graphs, each with 50 nodes, as described earlier. A network with 50 nodes represents a medium-sized 802.11 rural network. Fig. 9 compares the mismatch cost achieved in the three cases of comparison. The case without use of any heuristics is labeled "No-Heu". The 100 random graphs are sorted in the order of their costs with the Greedy-Col heuristic – this makes a visual comparison easier. We clearly see that the Greedy-Col heuristic performs significantly better than the No-Hue case. And the use of Match-DF brings in further improvement in most cases. In some cases, Match-DF has higher cost than Greedy-Col, since after all Match-DF is a locally applied heuristic and can lose out globally sometimes. The costs for the three cases averaged across the 100 graphs are: 10.58 (No-Heu), 6.38 (Greedy-Col), and 5.32 (Match-DF).

C. Heuristics for edge ordering

While the previous subsection described various heuristics for colour choice, we now explore heuristics for edge ordering.

We explore two different heuristics for edge ordering. Since we found above that the performance of the Match-DF heuristic was good, both these heuristics try to help the Match-DF heuristic in different ways.

Sum-Diffs: This heuristic is based on the intuition that some edges are "more difficult" to colour than others. It tries to colour these first, when there is maximum flexibility in terms of choice of colours (there is less flexibility in choice of colours as more and more edges are coloured). To capture a notion of "more difficult" to colour, we define a metric *Sum-Diffs(e)* for each edge e. This is the sum of the (absolute) differences between the DFs of e and each of its neighbours. Intuitively, the more this metric, the more difficult it is to match up DFs with neighbours in the Match-DF heuristic, while colouring e. We thus order the edges in decreasing order of this metric.

BFS: The second edge ordering heuristic is based on a Breadth-First-Search (BFS) ordering of the edges. The BFS ordering is obtained simply by performing a BFS traversal starting with an arbitrarily chosen node. During the traversal, the order in which the edges at a particular node are chosen is also arbitrary. In such a BFS ordering of edges, when it is turn for an edge to be coloured, most likely it will have some of its neighbours coloured, but not all. The fact that some neighbours are coloured helps in the application of the Match-DF heuristic, and the fact that some neighbours are not coloured helps in flexibility of colour choice.

Performance of Sum-Diffs and BFS: The edge ordering heuristics are applied in addition to the Match-DF heuristic for colour choice. We apply them to the same set of 100 random graphs as earlier, and for each graph, we find the additional improvement due to an edge ordering heuristic, as compared to using no edge ordering (with only the Match-DF heuristic). Fig. 10 shows these values. The graphs are sorted here in increasing order of their cost using the Match-DF heuristic. Since the edge-ordering heuristics are applied in addition to the Match-DF heuristic, we append a "Match-DF" to their labels in the plot.

We can see from the plot that there are a significant number of cases where the edge-ordering actually ends up performing poorer than just the Match-DF heuristic (negative improvement). However, for graphs with higher cost with the Match-DF heuristic (towards the right-hand side of the plot), we find that either of the edge orderings is able to show improvement over using just the Match-DF heuristic. The costs after applying the different edge ordering heuristics averaged over the 100 graphs are: 4.78 (Sum-Diffs::Match-DF), and 4.47 (BFS::Match-DF) – recall that the average cost was 5.32 for Match-DF for the same set of graphs.

D. Local search heuristic

While the above heuristics show significant improvement over No-Heu (a factor of 2 or more), we do not yet have an idea of how well these perform in comparison to what is optimally possible. In order to see this, we ran an (exponential) exhaustive search algorithm enumerating all possible colourings. We used smaller graphs, with 20 nodes each for this purpose – we could not find the optimal cost of graphs much larger than this (within reasonable amount of time) due to the exponential nature of the exhaustive search. We considered 20 such graphs and found the average costs of the various cases to be: 3.72 (No-Heu), 2.03 (Greedy-Col), 1.55 (Match-DF), 1.31 (Sum-Diffs::Match-DF), 1.40 (BFS::Match-DF), and only 0.43 for the optimal case. Hence the above heuristics perform worse than the optimal possible colouring.

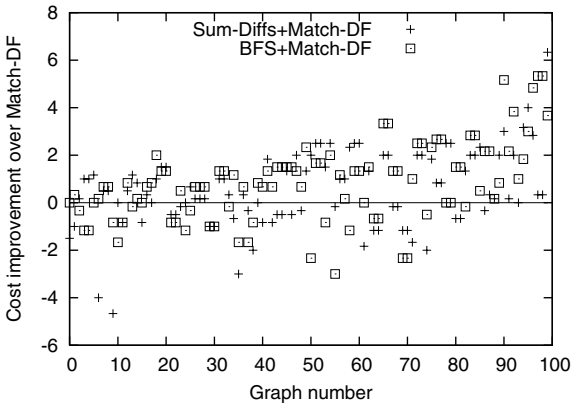

Fig. 10. Performance of Sum-Diffs and BFS

We visually compared the colouring produced by the exhaustive optimal search algorithm with the one produced by our heuristics (by programmatically generating an image file). We observed that the colouring matched for large parts of the graph, but were different in small parts, where the mismatch costs figured for the case of our heuristics. We however did not find any standard pattern in which our heuristic colouring could be altered to be made closer to optimal. This led us to try the following local search-based optimization heuristic.

After applying the previously described heuristics, we obtain a channel allocation. Some channel subgraphs in the resulting 3-edge-colouring have edges with mismatch between DF and AF. We now try to recolour these channel subgraphs, and the edges nearby, by means of an exhaustive enumeration of colouring possibilities. We however do not want to consider recolouring all these subgraphs at the same time, since that would increase the cost of exhaustive enumeration exponentially. Hence we consider them one after another. We proceed as follows.

Denote the channel subgraphs (before any recolouring, but after the initial colouring) as $S_1, S_2, ...S_l$, in decreasing order of mismatch cost. In the first step, we simply uncolour all the edges of S_1, and also all the neighbouring edges to this subgraph. We then perform an exhaustive search on the possible colourings of just this uncoloured part.

In a subsequent steps i, $i = 2..l$ of this optimization heuristic, we attempt to do similar recolouring for the subgraph S_i. But due to the recolouring done in previous stages, S_i may have been altered from what it was in the original graph. Hence we pick an edge e_i (arbitrarily) from each S_i before step-1. In

steps $i \geq 2$, we simply consider the channel subgraph (after colour merge, in the current colouring), that has edge e_i and check if it has non-zero cost. If so, we attempt to improve the colouring by a recolouring process like in the first step.

While the part of the graph over which we perform local exhaustive search at each step could theoretically include even all the edges of the graph in the worst case, in practice it has only about 10-20 edges – exhaustive search on this takes at most a few seconds in implementation. The optimization phase can in fact place a limit on the number of edges which are uncoloured and recoloured – over which exhaustive search is performed – we chose this limit to be 16 edges in our implementation.

We term this final heuristic as *L-Search* since it is based on several local exhaustive searches.

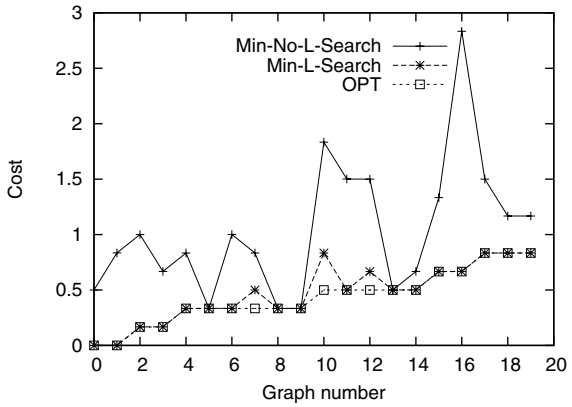

Fig. 11. Performance of L-Search

Performance of L-Search: Note that such a search is for optimization and is implemented on top of one of the heuristics presented earlier (to get the initial colouring). Fig. 11 presents a comparison of the performance of L-Search with the exhaustive optimal search (OPT) over the entire graph. We run L-Search on three different 6-edge-colourings – produced by Match-DF, Sum-Diffs::Match-DF, and BFS::Match-DF. We then choose the one that produces the minimum cost. This is labeled Min-L-Search in Fig. 11. We also show the minimum of the costs of the three possible colourings mentioned above, without the use of L-Search. This is labeled as Min-No-L-Search in the plot. As the plot shows, the Min-L-Search is almost always the same as OPT. The costs averaged across the 20 graphs for the various cases are: 1.2 (Min-No-L-Search), 0.47 (L-Search), and 0.43 (OPT).

We could not compare the performance of L-Search with that of OPT for the graphs with 50 nodes since the exhaustive search OPT does not complete within reasonable time on these graphs. However, L-Search shows significant improvement on top of the other heuristics even in the 50-node case. The cost averaged across the same hundred 50-node graphs as in the previous subsections is 1.51 for Min-L-Search, while it was 3.84 for Min-No-L-Search.

VI. Related Work

We now discuss prior work related to our research.

We have considered a combination of channel allocation and STDMA scheduling (2P) for controlling medium access in our 802.11 mesh network with point-to- point links. Channel allocation and frequency reuse is a well-studied issue in cellular networks [12]. However, the problem in cellular networks is quite different than in our context. While in cellular networks channel allocation is modeled as a node-colouring problem, in our setting, we have modeled it as a BP-proper edge-colouring problem. Our formulation of the minimum-mismatch channel allocation problem is also unique to our setting as compared to cellular networks.

STDMA has been considered by researchers for medium sharing in packet radio networks. Both link scheduling [13], [14], where individual graph links are scheduled, as well as broadcast scheduling [15], [16], [17], [18] where graph nodes are scheduled, have been considered. In these, the goal is to come up with a transmission schedule of time slots such that all links/nodes are scheduled within a minimum number of time-slots (i.e., with minimum schedule length). This problem is NP-complete for both the link and node scheduling variants [19], [15] and efforts have focused on other issues such as a distributed implementation [13], [15], [16], [17], [18]. Researchers have also considered restricted classes of graphs [19], [20] since the problems are NP-Complete on general graphs.

Another dimension that has been considered is scheduling to adapt to current/expected traffic patterns [14], [21], [22]. It is interesting to observe that bipartite graphs also figure in [21], [22], albeit in a context different from ours – in [21], [22] it is shown that non-bipartite graphs lead to significantly less efficient solutions. This is intuitive since scheduling around an odd cycle always leads to a conflict.

While most of the earlier work has considered only omni-directional antennas, more recent work has also considered directional antennas [23]. Link scheduling has also been considered specifically in the context of Bluetooth scatternets [22]. With directional antennas as well as in Bluetooth scatternets, more "reuse" is possible in STDMA scheduling, since there is lesser interference and many more transmissions can go in parallel.

Our work differs from past work on scheduling in two main aspects. First, *synchronous operation (SynOp)* at a node is possible in our setting. This is because (a) we use directional antennae, and (b) we know the exact locations of the nodes so the power levels of the links can be engineered with a careful link-budget analysis to reject the interfering transmission [8].

The relatively recent work on a unified framework for (T/F/C)DMA channel allocation [24] allows such a flexibility in theory. In [24], a generic algorithm is proposed for scheduling under a flexible set of constraints. However, [24] does not evaluate the performance of the generic algorithm under the flexibilities we have considered. Further, the following difference also holds with respect to [24].

The second main difference is that past work has considered scheduling in isolation and applies it to the entire network graph. However, we consider channel allocation in combination with scheduling. This allows us the flexibility of breaking up the network graph into small channel subgraphs, each of which is bipartite. This breakup in turn permits a simple and efficient two-phase scheduling in each of the channel subgraphs. Such a design is unique to our work. Another unique contribution of our work is our formulation of the minimum-mismatch channel allocation problem in terms of desired and achieved link capacities, and our heuristics to solve this.

VII. Discussion and Future Work

We now elaborate on a few points of discussion related to our design.

In our system, the channel allocations and schedules can be pre-computed centrally and passed on to all nodes. Another implementation aspect in our scheme is the granularity of scheduling. Using a granularity of one or more packet transmission lengths is feasible. This granularity must also be taken into account while the network operator species the DF values.

There are several dimensions for future research that arise out of our work. First, in our formulation of MMCA, we have considered mismatches between AF and DF with equal weightage for all links. However, in a real network, some links are more important than others. Also, it may be alright to achieve a capacity less than desired in a particular direction, but not in the other. Such considerations are natural extensions to our work.

While in our work we have assumed that the desired fractions are handed to us, guidelines for arriving at these values are required. Such guidelines may consider how routing is done in the mesh network, and where landline nodes are placed, to determine expected traffic volume on the links. Further, the DFs may be varied dynamically based on time-of-day dependent traffic patterns, or even more dynamic aspects such as link/node failures. The dynamic variation of DFs and dynamic channel allocation is a system design aspect that needs further study.

In our channel allocation algorithm and the heuristics, we have not paid attention to the angle of separation between two links. However, for more headroom in the link-budget analysis in [8], it is desirable if two links assigned the same channel are as far apart as possible. An algorithm that considers this aspect in channel allocation requires further exploration. This issue may also be addressed/alleviated by appropriate topology formation during rural network construction, by ensuring that links have enough angular separation. This and other considerations such as resilience to failures, closeness to points of landline connections, etc. would go into deciding the topology of a rural network. Topology formation is also an area for further study.

VIII. Conclusions

802.11 mesh networks are gaining popularity due to the easy and low-cost availability of the technology. In this paper, we have considered the issue of channel allocation in tandem with the 2P MAC protocol. We have addressed the important issue of flexible capacity allocation in mesh networks.

In our network, we have the flexibility of transmitting to or receiving from multiple directions simultaneously. The 2P MAC protocol fully utilizes this flexibility. We allocate channels such that we end up with bipartite channel subgraphs. This allows 2P to operate within each channel subgraph. We propose a 2-step algorithm to arrive at a BP-proper channel allocation: Vizing colouring, followed by colour merging. This works under the constraint $\Delta \leq 5$, which is not restrictive for sparse mesh networks.

To summarize, our contributions are in terms of (a) the overall system model for dividing a given network topology into bipartite channel subgraphs, so that 2P operation can be enabled throughout, (b) formulation of the problem of zero-mismatch channel allocation (ZMCA) and the minimum-mismatch channel allocation, (c) proof that ZMCA is NP-complete and thus MMCA is NP-hard, and finally (d) heuristics for achieving a close-to-optimal channel allocation and their evaluation.

Acknowledgment

This work was supported by Media Lab Asia (the RuralNet project: MLA/CS/20050014) and by the Ministry of Human Resources and Development (project: MHRD/CS/20030332). We thank Kameswari Chebrolu as well as the anonymous reviewers for their comments on earlier versions of this paper.

References

[1] "IEEE P802.11, The Working Group for Wireless LANs," http://grouper.ieee.org/groups/802/11/.

[2] Pravin Bhagwat, Bhaskaran Raman, and Dheeraj Sanghi, "Turning 802.11 Inside-Out," in *HotNets-II*, Nov 2003.

[3] Eric Brewer, Michael Demmer, Bowei Du, Kevin Fall, Melissa Ho, Matthew Kam, Sergiu Nedevschi, Joyojeet Pal, Rabin Patra, and Sonesh Surana, "The Case for Technology for Developing Regions," *IEEE Computer*, vol. 38, no. 6, pp. 25–38, June 2005.

[4] "Technology and Infrastructure for Emerging Regions," `http://tier.cs.berkeley.edu/`.

[5] "DjurslandS.net: The story of a project to support the weak IT infrastructure in an low populated area of Denmark," `http://djurslands.net/biblioteket/international/djurslands_net_english_presentation.ppt`.

[6] "HyperGain HG2424G 2.4 GHz 24 dBi High Performance Reflector Grid Antenna," http://www.hyperlinktech.com/web/hg2424g.php.

[7] Bhaskaran Raman and Kameswari Chebrolu, "Revisiting MAC Design for an 802.11-based Mesh Network," in *HotNets-III*, Nov 2004.

[8] Bhaskaran Raman and Kameswari Chebrolu, "Design and Evaluation of a new MAC Protocol for Long-Distance 802.11 Mesh Networks," in *11th Annual International Conference on Mobile Computing and Networking paper (MOBICOM)*, Aug/Sep 2005.

[9] M. Burton, "Channel Overlap Calculations for for 802.11b Networks," http://www.cirond.com/White_Papers/FourPoint.pdf.

[10] J. Misra and D. Gries, "A Constructive Proof of Vizing's Theorem," *Information Processing Letters*, vol. 31, no. 3, Mar 1992.

[11] I. Holyer, "The NP-Completeness of Edge-Colouring," *SIAM J. COMPUTING*, vol. 10, no. 4, pp. 718–720, Nov 1981.

[12] I. Katzela and M. Naghshineh, "Channel Assignment Schemes for Cellular Mobile Telecommunications: A Comprehensive Survey," *IEEE Personal Communications*, pp. 10–31, 1996.

[13] I. Chlamtac and S. Lerner, "A link allocation protocol for mobile multihop radio networks," in *Globecom*, Dec 1985.

[14] R. Ogier, "A decomposition method for optimal scheduling," in *24th Allerton Conference*, Oct 1986.

[15] R. Ramaswami and K. K. Parhi, "Distributed scheduling of broadcasts in a radio network," in *INFOCOM*, 1989.
[16] A. Ephremedis and T. Truong, "Distrbuted algorithm for efficient and interference-free broadcasting in radio networks," in *INFOCOM*, 1988.
[17] I. Cidon and M. Sidi, "Distrbuted assignment algorithms for multi-hop packet-radio networks," *IEEE Transactions on Computers*, vol. 38, no. 10, pp. 1353–1361, Oct 1989.
[18] I. Chlamtac and S. Kutten, "A spatial reuse TDMA/FDMA for mobile multi-hop radio networks," in *INFOCOM*, Mar 1985.
[19] S. Ramanathan and E. L. Lloyd, "Scheduling Algorithms for Multi-hop Radio Networks," *IEEE Transactions on Networking*, vol. 1, no. 2, pp. 166–177, Apr 1993.
[20] A. Sen and M. L. Huson, "A New Model for Scheduling Packet Radio Networks," in *INFOCOM*, 1996.
[21] L. Tassiulas and S. Sarkar, "Maxmin Fair Scheduling in Wireless Networks," in *INFOCOM*, 2002.
[22] S. Baatz, M. Frank, C. Kuhl, P. Martini, and C. Scholz, "Bluetooth Scatternets: An Enhanced Adaptive Scheduling Scheme," in *INFOCOM*, 2002.
[23] M. Sanchez, J. Zander, and T. Giles, "Combined Routing & Scheduling for Spatial TDMA in Multihop Ad hoc Networks," in *Wireless Personal Multimedia Communications (WPMC)*, 2002.
[24] R. Ramanathan, "A Unified Framework and Algorithm for Channel Assignment in Wireless Networks," in *INFOCOM*, 1997, pp. 900–907.