Channel Assignment for Multiple Interface Nodes in Wireless Ad hoc Networks

Minho Shin, Seungjoon Lee, Yoo-ah Kim

***Abstract*— In wireless networks, due to the broadcast property of the medium, nodes close to each other cannot simultaneously transmit over the same channel. One way to overcome this limitation is to use multiple independent channels available in the system. Although we can use a single wireless interface card to access multiple channels, such schemes can raise issues of compatibility (e.g., modification of the MAC protocol) and performance degradation (e.g., due to frequent channel switching). In this paper, we assume that nodes are equipped with multiple interface cards, and focus on the channel assignment problem for minimizing the total number of interferences among wireless links. We show that the problem is NP-hard and present distributed heuristics. We also present two centralized algorithms and show that the algorithms give constant factor approximation guarantees. We perform simulation experiments for the proposed distributed heuristic. The results show that compared to one-channel scenarios, our proposed algorithm can reduce the number of interferences by up to 85% when nodes are equipped with four interface cards. Through detailed packet-level simulation experiments, we also show that depending on the scenario, the resulting channel assignment actually achieves up to seven times throughput improvement over the single-channel case.**

I. Introduction

In wireless networks, due to the broadcast property of the medium, nearby nodes *interfere* with each other and cannot simultaneously transmit over the same wireless channel. One way to overcome this limitation is to use independent channels available in the system. For example, the IEEE 802.11b standard [1] defines three independent channels (or non-overlapping frequency ranges), and administrators of typical wireless local area networks (WLANs) assign independent channels to neighboring access points. This channel assignment allows multiple nodes in the system to transmit and receive data packets, and leads to higher system performance. Although such centralized channel assignment works well in the infrastructure-based environment, this scheme is not applicable to ad hoc networks, which typically lack in infrastructure. The goal of this paper is to develop a scheme that exploits independent channels available to wireless devices and enhances the throughput of ad hoc networks.

We can consider two different strategies for using multiple channels to achieve higher network performance. The first strategy is to enable a single wireless interface card to access multiple channels [2], [3], [4]. For example, So and Vaidya [2] propose a scheme that allows wireless devices to communicate on multiple channels using a single interface card. However, the scheme requires the modification of the MAC protocol. It also results in frequent channel switching, which incurs considerable overhead in the current hardware []. The other strategy is to use multiple interface cards [5], [6], [7], [8]. Suppose that all nodes have two 802.11b interface cards as shown in Figure 1.[1] In this example, node A is using two independent channels: channel 1 for e_1 and channel 2 for e_2. With this channel assignment, we can use all three wireless links at the same time. If all nodes have packets to transmit, then the network throughput using the above channel assignment will be three times as high as that of the single-interface case. In this paper, we assume that there exist nodes equipped with multiple interface cards, and focus on the channel assignment problem to enhance the overall network throughput.

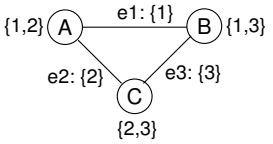


Fig. 1. In this ad hoc network, each node has two interface cards. We can assign a distinct channel to each link as shown above. With the above channel assignment, there is no interference among wireless links.

As previously observed in [9], [10], *interference* among wireless links is a major factor that limits overall network throughput. For example, the channel assignment shown in Figure 1 leads to no interference among links, while the use of only one channel would result in interference between all the three pairs of edges (e_1-e_2, e_1-e_3, and e_2-e_3) [10]. We consider the problem of channel assignment to minimize the total number of interferences in an ad hoc network. Our contributions are as follows:

- We formulate the channel assignment problem considering the effect of interference on the network throughput. We also show that even in its simplest setting, the channel assignment problem is *NP-hard*.
- We propose a *distributed heuristic* for channel assignment in wireless ad hoc networks. By utilizing as many independent channels available in the system as possible, our proposed scheme significantly reduces the total number of interferences among links.
- When all nodes have the same number of interfaces, we

M. Shin, S. Lee, Y. Kim are with Department of Computer Science, University of Maryland, College Park, MD 20742. E-mail: {mhshin,slee,ykim}@cs.umd.edu.

[1] In the IEEE 802.11b standard, we use independent channels 1, 6, and 11. However, to simplify the notation in this paper, we use $\{1, 2, \cdots, k\}$ for a set of k independent channels.

prove that a simple greedy algorithm gives a solution that is essentially best possible. We also present two centralized algorithms for more general networks and prove that they have constant factor approximation guarantees.
- Through *packet-level simulations* we show that the decrease in the total number of interferences indeed leads to throughput improvement. Our results show that compared to the one-channel case, our scheme using four interfaces can increase the network throughput by up to seven times.

The rest of this paper is organized as follows: We review related work in Section II and describe our design goals and problem formulation in Section III. We present our distributed heuristic in Section IV and discuss theoretical aspects of the channel assignment problem in Section **??**. In Section VI, we present our simulation results, and Section VII concludes.

II. Related Work

Many schemes have been proposed to exploit multiple channels for performance improvement in wireless networks. One class of work is to enable only one network interface card to operate in multiple channels. Multinet [3] exploits Power Saving Mode in the IEEE 802.11, and a node switches channels and checks TIM messages to see if there are incoming data packets. The MMAC (Multi-channel MAC) protocol also uses a single interface to access multiple channels [2]. In MMAC, time is divided into beacon intervals, and as in the PSM operation of the IEEE 802.11 standard [1], nodes buffer data packets until next beacon interval. In the beginning of each beacon interval, nodes with buffered data packets exchange a number of control messages with destinations, so that they can agree on the channel for data transmission. However, this scheme requires the change in MAC protocols and thus cannot be readily used in current wireless networks. Also, MMAC can increase packet latency due to channel switching. Nasipuri et al. [11] also propose a protocol, which assumes a node can continuously monitor all available channels. However, most of current wireless interfaces do not have such capability.

Another line of work is to use multiple interfaces available at each wireless nodes, which is closest to our work. In Multiraiod Unification Protocol (MUP) [5], when a node has k network interface cards, it only uses k channels (channels 1, 2, $\cdots$, k). Each node statically assigns a channel to each interface card, and when a node needs to transmit a packet, it checks the channel condition and uses the channel with the best condition at that time. We discuss the difference between MUP and our scheme later in this paper. Raniwala et al. [6] consider the channel assignment problem combined with routing in the context of static mesh networks. They assume that long-term traffic load between source and destination pairs is known a priori, and based on the information, and present a centralized heuristic for throughput improvement. Our work proposes a distributed algorithm for channel assignment in general networks.

Draves et al. [12] propose a new routing metric when nodes have multiple interface cards. They use fixed channel assignment to find high-throughput paths, but do not consider the channel assignment problem itself. Jain et al. [10] consider the problem of finding an optimal path, given the network topology and workload specification. They show that the problem is NP-hard, and provide a centralized algorithm based on an integer program, which can be generalized to the case where nodes have multiple interfaces.

III. Model and Problem Formulation

In this section, we first describe our design goals for our approach. Then, we present the network and interference model used in this paper. Finally we formulate our channel assignment problem, which is a variant of the well-known edge-coloring problem.

A. Design Goals

The goal of this paper is to develop a distributed scheme for channel assignment to improve network throughput when there exist nodes with multiple interface cards. Our design goals are as follows:

- The proposed scheme should use currently available hardware and MAC protocols without the modification.
- The proposed scheme should inter-operate with existing upper-layer protocols with minimal modification.
- The channel assignment should be done in a distributed way using only local information.

As shown in [10], [6], even when we use a centralized algorithm based on the knowledge of traffic pattern, finding an optimal channel assignment and path selection is NP-hard. However, in general we do not know the traffic pattern in ad hoc networks, and the use of distributed algorithms is desirable in ad hoc networks. We use a different approach to improve network throughput as described below.

We consider the number of *conflicts* between two edges where concurrent transmissions would interfere with each other. Our channel assignment scheme attempts to minimize the number of such conflicts. (In this paper, we use interference and conflict interchangeably.) We assume that (1) we use a contention-based MAC protocol (e.g., the IEEE 802.11 standard), and (2) the access probability is identical for all links. For example, we expect that the second assumption is true when the network load is high. Under these assumptions, a smaller number of interferences results in the increase in expected network throughput. Our channel assignment schemes attempt to improve the network throughput by minimizing the number of interferences. In Section VI, we use simulations to show that the decreased interference level by our channel assignment scheme indeed leads to network throughput improvement.

In the rest of this section, we define the network and interference model and formally define the channel assignment problem.

B. Network and Interference Model

We represent a network as an undirected graph $G = (V, E)$, where V is a set of nodes, and E a set of links. Let C_G be the number of channels available in the system and C_v be the number of network interface cards at node v. Without

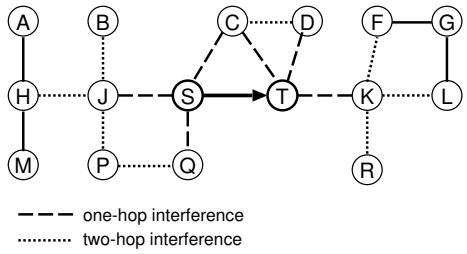


Fig. 2. Illustration of one-hop and two-hop interference when the wireless link from S to T is used.

loss of generality, we assume $\forall v, C_v \leq C_G$. Although each network interface can change the channels for transmission and reception, it can operate only one channel at a given time. We assume that there is no interference between independent channels.

In this paper, we consider the following two simplified interference models. Clearly, a node cannot simultaneously transmit and receive over a single channel, and we represent such constraint using the *one-hop* interference model [13]. Therefore, an edge e interferes with the following set of edges: $I(e) = \{e' | e'$ has a common endpoint v with $e\}$. For example, in Figure 2, when node S sends a packet to node T, all neighboring nodes of S and T (e.g., C, D, J, K, and Q) cannot send data packets to S and T (shown in dashed lines). This one-hop interference model is simple and uses only one-hop neighborhood information, it sometimes does not capture the real wireless communication environments. For example, a transmission from B to J may collide with the transmission from S and T.

In the *two-hop* interference model [14], in addition to the constraint in the one-hop model, edge e cannot be used if there exists an ongoing transmission over an edge e' where either endpoint of e is a neighbor of either endpoint of e'. That is, $I(e) = \{e' | e$ and e' have a common endpoint or are adjacent to a common edge $e''\}$. For example, in Figure 2, $I(e = (S,T)) = \{e' | e'$ is incident to C, D, J, S, T, K, and $Q\}$ (shown in dotted lines). However, any edge in solid line (e.g., from G to L) can be used together with $e = (S,T)$. Since transmissions from J to H can concur with transmission from S to T, this two-hop interference model can be seen as too restrictive. However, if we use RTS/CTS control message exchanges as in the 802.11 MAC protocol [1], the two-hop model approximates the actual wireless interference well. Although other more complicated interference models are used in the literature [9], in this paper we focus on the one-hop and two-hop interference models when we assign the channels to interface cards at each node.

C. Channel Assignment Problem

Let us define the *interference number* (i_e) of an edge $e \in E$ to be the number of other edges that interfere with e depending on the interference model [9]. In other words, i_e is the number of edges in $I(e)$ that use the same channel as e. Our goal is to minimize the following metric:

$$i_G = \frac{1}{2} \sum_{e \in E} i_e. \quad (1)$$

One obvious way to reduce the metric is to disable some edges while maintaining network connectivity [15], [16]. However, we assign channels such that we retain all wireless edges in E. We can still combine the two approaches by applying our proposed scheme onto network topologies obtained using the topology control schemes [15], [16].

Our channel assignment problem is a variant of the edge coloring problem. Recall that the objective of edge coloring is to minimize the number of colors when we assign a color to each edge, such that no two adjacent edges have the same color. However, our problem is different from the original edge coloring problem in two aspects. First, the coloring need not be proper, but two adjacent edges are allowed to use the same color, and the goal is to minimize the number of such conflicts (also known as *soft* edge coloring []). Second, each node has its local color constraints, which limit the number of colors that can be used by the edges incident to the node. For example, if a node has two interfaces, it can use only up to two colors. Last, we consider a more general model for edge conflicts (i.e., two-hop interference model).

In the next sections, we present various algorithms for this channel assignment problem. In the remainder of this paper, we mean *channels* by *colors* and use edge coloring and channel assignment, interchangeably. We also use conflict and interference interchangeably.

IV. Distributed Heuristics

In this section, we present distributed heuristics for the channel assignment. We assume that neighbors exchange information such as their interface counts and channel assignment. We first focus on the initial color assignment procedure and discuss the adaptive operation in Section IV-C. Table I summarizes the notations we use in the algorithm description.

Our proposed heuristics consist of two steps. In the first step, each node v chooses its own set of colors $S(v)$ of size C_v, such that any two neighbors have a common color. More formally, we ensure that $\forall e = (u,v), S(u) \cap S(v) \neq \emptyset$. In the second step, for each $e = (u,v)$ node v chooses a color from $S(u) \cap S(v)$ to assign the color to e. In this step, each node attempts to balance the numbers of colors assigned to its edges. Formally, a node attempts to minimize:

$$\delta_v = \max_{c \in S(v)} \lambda_v(c) - \min_{c \in S(v)} \lambda_v(c), \quad (2)$$

where $\lambda_v(c)$ denotes the number of edges in $E(v)$ that are assigned color c. Intuitively, the above heuristic tries to minimize the one-hop interference using local information.

Color assignments are performed in a distributed fashion. when node v wants to color its edges, v first requests its neighbors that they do not change colors until v finishes. v waits until every neighbor acknowledges the request. Node v determines the colors of $E(v)$ only with neighbor information, and once v determines the color of edge $e = (u,v)$, u cannot change the color of e.

Notation	Definition
C_G	the number of available channels in the system
C_v	the number of colors that node v can use
$S(v)$	the color-set of node v ($C_v = \|S_v\|$)
$E(v)$	the set of edges incident to v
$N(v)$	the set of v's neighbors
$N_i(v)$	the set of v's neighbors with i interfaces
$N_L(v)$	$\bigcup N_i(v) \quad \forall i < C_v$
$U(v)$	the set of neighbors u such that edge (v,u) is not determined
$T(v)$	the set of v's neighbors with $S(v) \neq \emptyset$
$\bar{U}(v)$	$E(v) - U(v)$

TABLE I
NOTATIONS

We present two heuristics that use different mechanisms for these two steps. We begin with a simpler approach and then present a heuristic that further improves the performance.

A. BASIC-COLORING *algorithm*

In the BASIC-COLORING algorithm, each node v simply chooses $S(v) = \{1, 2, \ldots, C_v\}$, which is similar to MUP [5]. For example, nodes with two interfaces use $\{1, 2\}$ for their color-sets. In this assignment, any pair of nodes share at least one common color (i.e., color 1). In the original MUP scheme, a node does not fix a channel to a link, but opportunistically uses the channel with best quality. In this section, we assume that based on the chosen sets, a node fixes a channel to each link, and discuss this issue later in Section IV-C.

We now discuss the channel assignment to edges, based on the chosen color sets. In *homogeneous* networks, where every node has k interface cards, $\forall v, S(v) = \{1, 2, \cdots, k\}$, and assigning a color uniformly at random is likely to lead to small δ_v. However, in *general* networks, nodes can have different numbers of interfaces, and this random channel assignment does not perform well. For example, suppose that only two neighbor nodes have two interfaces, while the other nodes in the network have only one interface and use channel 1. In this scenario, the two nodes should use channel 2 for higher performance, but the random assignment may assign channel 1 with the probability of 0.5.

We present a channel assignment scheme in which node v attempts to minimize δ_v in the case of general networks. Instead of randomly choosing colors, we assign colors in round robin. Let $U(v) \subset N(v)$ be the set of nodes u such that (u, v) is not colored as yet when v tries to assign colors to $E(v)$, and subset $U_i(v) \subset U(v)$ denote a set of nodes u which have i *interfaces*. In addition, we define $\bar{U}(v)$ to be $N(v) - U(v)$, and subset $\bar{U}_c(v) \subset \bar{U}(v)$ to be a set of nodes u such that (u, v) have chosen *color* c.

Fig. 3 illustrates BASIC-COLORING at node v with four interface cards. The set of v's neighbors is $N(v) = \{A, B, \ldots, G\}$, and their color-sets are shown in the figure. For example, node C has three interfaces, thus $S(C) = \{1, 2, 3\}$. If a neighbor already colored edges to node v (nodes B and E in Fig 3), we show the chosen colors next to the edges (colors 2 and 3 for edges (v, B) and (v, E), respectively). We

Algorithm 1 BASIC-COLORING**: Color Assignment at node** v

We consider color $c = 1$ to C_v in round robin and repeat the following until $U(v)$ becomes empty.
if $\bar{U}_c(v)$ is not empty **then**
  we remove one node from $\bar{U}_c(v)$.
else
  we choose node u from $U(v)$ with smallest C_u such that $C_u \geq c$. If such u exists, we assign c to edge (v, u) and remove u from $U(v)$.
end if

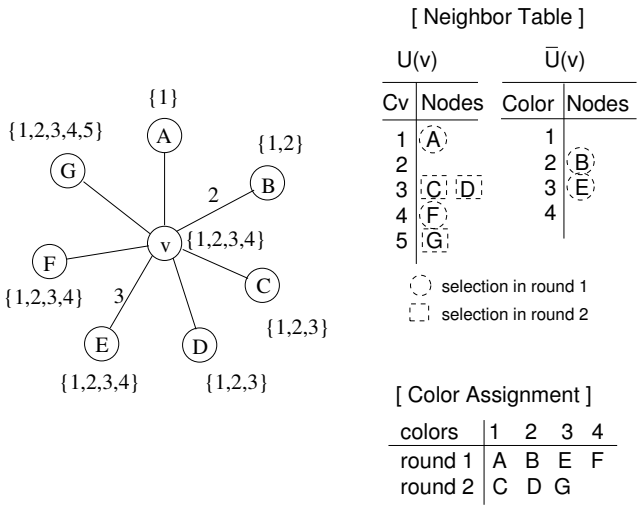


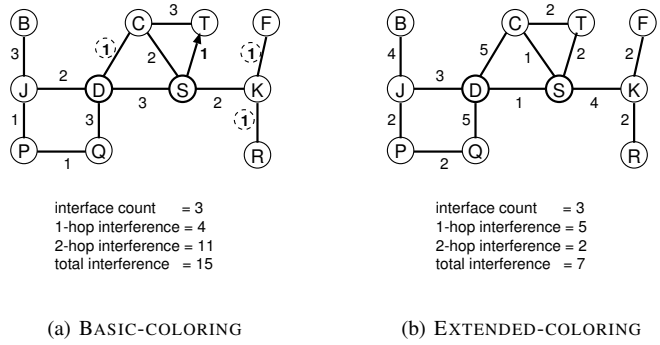
Fig. 3. Example of BASIC-COLORING

sort $U(v)$ by their interface counts (C_u), and sort $\bar{U}(v)$ by their chosen colors.

In the first round, we choose A, B, E, and F (in dashed circles) for color $1, 2, 3$, and 4, respectively. Then we remove A, B, E, and F from $U(v)$ and $\bar{U}(v)$. In the second round, since there is no neighbor such that $C_u = 1$ or 2 in $U(v)$ or $c_{v,u} = 1$ or 2 in $\bar{U}(v)$, we assign colors 1 and 2 to edges to C and D, respectively. Finally, we choose G for color 3. The resulting colors are shown in the figure. Note that $\delta_v = 1$ in this example.

Drawback of BASIC-COLORING*:* The BASIC-COLORING algorithm uses channels only from $\{1, \ldots \max_v C_v\}$. Usually the number of available independent channels is larger than $\max_v C_v$; for example, IEEE Std. 802.11a provides up to twelve independent channels while most mobile stations are equipped with a few interface cards. However, utilizing more available channels for wireless links can reduce 2-hop interference significantly.

Consider a channel assignment by BASIC-COLORING algorithm in Figure. 4-(a) where all the nodes have three interface cards. The given channel assignment allows only four 1-hop interferences. However, when node S transmit a packet to T, three other edges ((C, D), (K, F), and (K, R)) cannot be used because of 2-hop interference with the edge (S, T). We can remove these 2-hop interferences by introducing a new color, say color 4. If we assign color 4 to those three edges,



(a) BASIC-COLORING (b) EXTENDED-COLORING

Fig. 4. Comparison of BASIC-COLORING and M-coloring by example graph

$((C, D)$, (K, F), and $(K, R))$, we can eliminate three 2-hop interferences without violating color constraints.

B. EXTENDED-COLORING *algorithm*

To address the limitation of BASIC-COLORING algorithm which uses $\max C_v$ colors, we propose a new heuristic, which we call EXTENDED-COLORING algorithm. In this algorithm, we use $(2 \max C_v - 1)$ colors without violating color constraints. Since we use more channels in the system, the total number of interferences significantly decreases as shown later in this paper.

We begin with the case of homogeneous networks. Consider two nodes u and v with the same number of interfaces, say K. In EXTENDED-COLORING, they randomly choose K colors from $P = \{1, 2, \ldots, 2K-1\}$. Then, by the pigeonhole principle, $S_u \cap S_v \neq \emptyset$. ($2K$-1) is the maximum number when we use the pigeonhole principle, and as shown in Section VI, the use of a smaller set for P leads to a smaller amount of performance improvement. Although this simple algorithm works for homogeneous networks, in general it does not guarantee $S_u \cap S_v \neq \emptyset$. For example, if $C_u = 2$ and $C_v = 3$, and u picks $S(u) = \{1, 2\}$ from $\{1, 2, 3\}$ and v picks $S(v) = \{3, 4, 5\}$ from $\{1, 2, \ldots, 5\}$, $S(u) \cap S(v) = \emptyset$. However, we can simply solve this problem as follows: since v has a neighbor u with $C_u = 2$, it chooses two colors from $\{1, 2, 3\}$. Then, the remaining color can be chosen randomly. We next present our heuristic that guarantees a common channel between neighbor in the case of general networks.

In EXTENDED-COLORING, each node v with k interfaces waits for all neighbors in $N_L(v)$ (see Table I for definition) to select their color sets and notify v. Then, v chooses k colors from $P = \{1, 2, \ldots, 2k-1\}$ as follows. Let $T(v)$ be the set of neighbors that have already selected their color sets. Note that $T(v)$ includes all the nodes in $N_L(v)$ and some nodes in $N_k(v)$ since some nodes in $N_k(v)$ may have already chosen their color sets depending the order of running the algorithm. Algorithm 2 is the max-color algorithm for choosing color set at node v.

Theorem 4.1: Suppose $S(v)$ is the color set chosen by the above algorithm. Then, for any neighbor node $u \in T(v)$, $S_(u) \cap S(v) \neq \emptyset$

Algorithm 2 EXTENDED-COLORING: Choose color set at node v

At i-th iteration ($i = 1$ to C_v), we do the following.
k_i = the number of colors chosen after $(i-1)$-th iteration
if $N_i(v)$ is not empty or $i = C_v$ **then**
  if $i = 1$ **then**
    choose color 1
  else
    From set $\{\max(1, 2k_i), \ldots, 2i-1\}$, choose $(i - k_i)$ most frequent colors in color sets of nodes in $T(v)$.
  end if
end if

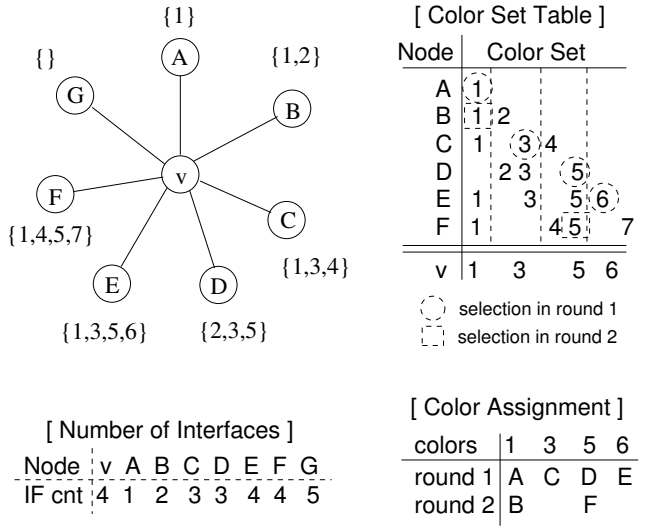


Fig. 5. Example of BASIC-COLORING

Proof: See Appendix A. ■

Figure 5 illustrates how a node chooses its color set according to Algorithm 2. We list the number of interfaces at each node in the figure. The *color-set table* at each node records the color set of each neighbor, which is sorted by the number of interfaces. Since $C_G > C_v$, G chooses its color set after v, and the color set of G is empty. In this example, $T(v) = \{A, B, C, D, E, F\}$. When $i = 1$, we choose color 1 since $C_A = 1$. When $i = 2$, $k_i = 1$ and we have to choose $i - k_i = 1$ color out of $\{2, 3\}$. In this example, we choose color 3 because more neighbors have chosen color 3 than color 2. Also, when $i = 3$, $k_i = 2$ and we choose $i - k_i = 1$ color. Color 5 is chosen because 5 occurs more than 4. Similarly, we choose 6 when $i = 4$. In this way, node v choose $S(v) = \{1, 3, 5, 6\}$, as shown at the bottom of color set table in the figure.

Once node v chooses its color set $S(v)$, it assigns the colors to the edges incident to nodes in $T(v)$. The other edges will be colored by neighbors in $N(v) - T(v)$. The algorithm 3 describes the procedure in which a node assigns colors to edges in round robin fashion as in BASIC-COLORING.

We again use Figure 5 to illustrate the color assignment using EXTENDED-COLORING. As a result of algorithm 2, $S(v) = \{1, 3, 5, 6\}$. Suppose that we choose c in an increasing

Algorithm 3 EXTENDED-COLORING: Color assignment at node v

repeat
  We choose $c \in S(v)$ in round robin.
  We choose a node, $u \in T(v)$ with smallest C_u s.t. $c \in S(u)$.
  if u exists **then**
    We assign c to edge (u, v) and remove u from $T(v)$.
  end if
until $T(v)$ becomes empty

order from 1 to 6. When $c = 1$, color sets of nodes A, B, C, E, and F contains color c. However, A has the smallest $C_A = 1$, and therefore we color edge (v, A) in color 1 and remove A from $T(v)$. When $c = 3$, nodes C, D, and F has color 3 in their color sets, but C has the smallest $C_C = 3$ and we color edge (v, C) in color 3. Similarly, we color edges (v, D) and (v, E) in color 5 and 6, respectively. After the first round, we have $T(v) = \{B, F\}$. In the second round, when $c = 1$ we choose B since $1 \in S(B)$. We cannot color when $c = 3$ since no node in $T(v)$ contains 3 in its color set. Finally, we choose F for $c = 5$. In Figure 5, we show the resulting color assignment in the color assignment table.

Fig 4 compares the performance of BASIC-COLORING and EXTENDED-COLORING. The total number of interferences is 15 for BASIC-COLORING, and 7 for EXTENDED-COLORING. Although the number of one-hop interferences for EXTENDED-COLORING (5) is slightly larger than that of BASIC-COLORING (4), EXTENDED-COLORING significantly decreases the number of two-hop interferences, and as shown in Section VI, this results in substantial improvement in network throughput.

C. Discussions

1) Comparison with Opportunistic Schemes : In our proposed heuristics, we fix a channel to each wireless link. However, an obvious alternative is to use an opportunistic scheme similar to MUP [5], where after each node chooses its color set, two neighbors agree on a common set of channels and select the best one according to the current network condition. Such an opportunistic approach requires channel quality estimation, which usually uses frequent probe message (e.g., two probes per second) [5]. We also expect that in high-load scenarios, the performance of our fixed-channel strategy will be similar to the opportunistic case.

Another related question is that the cardinality of common channel set is larger in BASIC-COLORING than in EXTENDED-COLORING. Suppose that all nodes have k interfaces. If we randomly selects k colors out $(2k-1)$ colors, in EXTENDED-COLORING, the average size of common channel set between two neighbors is $\frac{k^2}{2k-1} = k - \frac{k^2-k}{2k-1} \leq k$. In BASIC-COLORING, all neighbors share k channels, and it may appear that using the above opportunistic approach, BASIC-COLORING provides a better chance to use a channel with high quality. However, as shown in Section VI, the fundamental limiting factor is the total number of utilized channels, and the smaller number of utilized channels in BASIC-COLORING leads to network saturation at lower data rate than EXTENDED-COLORING.

2) Adaptivity to Dynamic Ad Hoc Networks : In ad hoc networks, nodes can join, leave, and even change their positions. Our proposed algorithm can be adaptively used for such dynamic environment as follows. When a new node n tries to join the network, it first identifies its neighbors, $N(n)$, by scanning all available channels. Since $\forall v \in N(n)$, $S(v) \neq \emptyset$, n can immediately run EXTENDED-COLORING algorithm to choose its color set $S(n)$ and determine colors of edges incident to neighbors with less or equal number of interfaces. We color the remaining edges later by the following process. Also, each node v periodically updates its neighbor list $N(v)$ and checks if there exists $u \in N(v)$ such that $S(v) \cap S(u) = \emptyset$. If there exists such u, v runs the EXTENDED-COLORING algorithm. Even when no such u exists, v may find that the channel assignment is not balanced (e.g., due to node movement). In this case, v re-assigns the channels to edges using Algorithm 3. Note that the update of $S(v)$ at node v may require neighbor node w to update its color set if $c_w > c_v$. However, the maximum interface in the network is likely to be small in practice (e.g., 3 or 4), and we expect that the number of affected nodes is small. We plan to further investigate the control overhead of our scheme in dynamic ad hoc networks in the future.

In the next section, we consider theoretical aspects of this channel assignment problem.

V. FURTHER THEORETICAL RESULTS

In this section, we discuss theoretical results on the one-hop interference model. We present a distributed greedy algorithm for homogenous networks and prove that the algorithm gives a solution that is essentially best possible. We also present two centralized algorithms for 1-or-k networks[2] with constant approximation factors.[3]

In the one-hop interference model, a node experiences the minimum number of conflicts when the colors of its incident edges are evenly distributed. In other words, the number of conflicts is minimized at node v when d_v/C_v edges are chosen for each color, where d_v denotes the degree of node v. Therefore, the average number of one-hop interferences of each edge $e \in E(v)$ is lowerbounded by $d_v/C_v - 1$. This lower bound plays an important role in proving our approximation guarantees. For the two-hop interference model, however, the lowerbound is not enough to prove a non-trivial approximation guarantees.

A. Greedy Algorithm

Although the channel assignment problem appears easy at first glance, we can show that it is NP-hard even in the simplest setting. Consider the homogenous network ($\forall v, C_v = k$). We can easily show that the problem of minimizing one-hop interferences is NP-hard for arbitrary K by the reduction

[2]Nodes in the network have either one or k interface cards.

[3]An algorithm is said to give an approximation factor α when the cost of the solution produced by the algorithm is at most α times the optimal.

from edge coloring problem [17]. However, a very simple greedy algorithm gives a solution in which the total number of interferences is at most $OPT + |E|$. The result is essentially best possible since it is NP-hard to approximate within additive term of $o(|E|^{(1-\epsilon)})$ for a given $\epsilon > 0$.

The greedy algorithm works as follows. We use colors from $\{1, \ldots, k\}$. For any uncolored edge $e = (u, v)$, we choose a color for edge e that introduces the smallest number of interferences. More formally, let $n(c, v)$ be the number of edges in $E(v)$ that are already colored with c. We choose color c such that $n(c, u) + n(c, v)$ is minimized. We repeatedly color edges until all edges get colored.

We can show that the average number of interferences over all edges in $E(v)$ is at most d_v/k in the solution obtained by the greedy algorithm. As mentioned above, lowerbounds can be obtained when each edge $e \in E(v)$ experiences $d_v/k - 1$ interferences and therefore, we have the following theorem.

Theorem 5.1: When we use the greedy algorithm, the total number of one-hop interfences in homogeneous networks is:

$$i_G = \frac{1}{2} \sum_{e \in E} i_e \leq OPT + |E|. \tag{3}$$

Proof: From the viewpoint of node v, the number of one-hop conflicts is minimized when edges incident on the node are evenly distributed. In that case, each edge $e \in E(v)$ experiences $d_v/k - 1$ interferences at average when all nodes have k interface cards. Therefore, the total number of conflicts is lower bounded by:

$$LB = \frac{1}{2} \sum_{v} (\frac{d_v}{K} - 1) d_v = \frac{1}{2} \sum_{v} \frac{d_v^2}{K} - |E|. \tag{4}$$

To upperbound the number of intereferences by the greedy algorithm, consider an edge $e = (u, v)$. Let $n(e)$ be the number of interferences that e introduces when it gets colored. The total number of interferences in the final coloring is $\sum_e n(e)$. Since we choose a color for e such that it introduces the smallest interferences in node u and v at the time in which edge e gets colored, $n(e)$ is at most $(d_v(e) + d_u(e))/K$ where $d_v(e)$ and $d_u(e)$ are the number of edges in $E(v)$ and $E(u)$ that get colored *before* e.

Therefore, the total number of conflicts by the greedy algorithm is:

$$\begin{aligned}
\sum_e n(e) &\leq \sum_{e=(u,v)} (d_v(e) + d_u(e))/K \\
&= \sum_v \sum_{e \in E(v)} d_v(e)/K \\
&= \frac{1}{K} \sum_v d_v(d_v - 1)/2 \\
&\leq \frac{1}{2} \sum_v \frac{d_v^2}{K} \\
&\leq OPT + |E|.
\end{aligned}$$

■

The approximation ratio is best possible by the following theorem.

Theorem 5.2: It is NP-hard to approximate the channel assignment problem in the one-hop interference model within an additive term of $o(|E|^{1-\epsilon})$.

Proof: See Appendix B. ■

In the next two sections, we further examine theoretical aspects of the channel assignment problem for more general cases.

B. Centralized Greedy Algorithm

We consider the networks where each node can have either one or k interfaces. These cases are interesting for the following reasons. (1) As shown below, even when $k = 2$, the problem of minimizing the number of one-hop interferences is NP-hard. (2) It best captures the current realistic setting, in which most of mobile stations are equipped with one interface card and some of them have two (or k, in general) interface cards. We present two centralrized algorithms for this type of networks and analyze the approximation factors of the algorithms.

We show that the problem of minimizing the number of one-hop interferences in 1-or-k networks is NP-hard. We prove this by the reduction to 3SAT [17]. The reduction is simliar to the reduction from edge coloring to 3SAT [18].

Theorem 5.3: The channel assignment problem to minimize the number of one-hop interferences is NP-hard even when $C_v = 1$ or 2.

Proof: See Appendix C for the proof. ■

We first present a centralized greedy algorithm for 1-or-k networks. The algorithm generalizes the idea of the greedy algorithm for homogenous networks. The difficulty in this case comes from the fact that an edge cannot choose its color locally since the color choice of an edge can affect colors for other edges to obey color constraints and connectivity.

Before describing the algorithm, we define some notations. Let $V_i \subset V$ be the set of nodes v with $C_v = i$ (i.e., we have V_1 an V_K). We partition V_1 into several clusters $V_1^1, V_1^2, \ldots V_1^t$, such that nodes $u, v \in V_1$ belong to the same cluster if and only if there is a path composed only of nodes in V_1. Let E_1^i be a set of edges either of which endpoints is in V_1^i. Note that all edges in E_1^i should have the same color. We also define $B_1^i \subset E_1^i$ to be a set of edges where one endpoint is in V_1^i and the other is in V_K. We can think of B_1^i as a set of edges in the boundary of cluster V_1^i. See Fig 6 for example.

In the greedy algorithm for homogenous networks, each edge greedily chooses a color so that the number of interferences it creates (locally) is minimized. Similarly, in 1-or-k networks, edges in the same cluster E_1^i choose a color so that the number of interfaces it creates is minimized. Note that, however, it cannot be done efficiently in distributed manner since all edges in the cluster should agree to have the same color. Once edges in E_1^i for all i choose their colors, the remaing edges (edges not belonging to any cluster) greedily chooses their colors. Algorithm 4 describes the centralized greedy algorithm.

Since we choose a color for a group of edges at a time, the

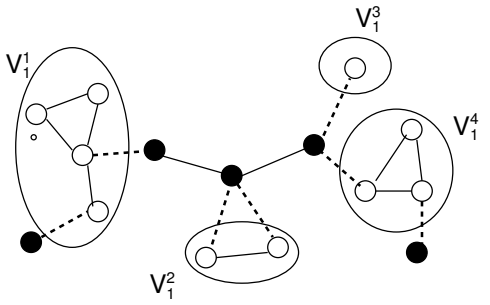


Fig. 6. The figure show an example of clusters V_1^i in 1-or-k networks. White nodes have only one interfaces and black nodes have k interfaces. Dotted lines belong to B_1^i.

Algorithm 4 Centralized Greedy Algorithm

for each cluster V_1^i **do**
  choose a color for edges in each E_1^i as follows.
  if B_1^i is empty **then**
    choose any color for E_1^i.
  else
    for each color $c \in \{1, \ldots, k\}$ **do**
      count the number of interferences to be created when we choose color c for E_1^i. Formally, count $\sum_{e=(u,v)\in B_1^i} n_c(v)$ where $n_c(v)$ is the number of edges $e' \in E(v)$ with color c.
    end for
    choose a color c that minimizes $\sum_{e=(u,v)\in B_1^i} n_c(v)$.
  end if
end for
for each edge that does not belong to any E_1^i **do**
  choose a color using the simple greedy algorithm in Section V-A.
end for

approximation factor is not as good as the greedy algorithm for homogenous networks.

Theorem 5.4: The approximation ratio of the centralized greedy algorithm in the one-hop interference model for 1-or-k networks is $(2 - 1/K)$.

Proof: See Appendix D. ∎

When $K = 2$, the centralized greedy algorithm finds a 1.5-approximation. In the following section, we use a more involved algorithm to obtain a better approximation ratio.

C. SDP-based Algorithm

In this section, we formulate the problem using semidefinite programming[4](SDP) relaxation, and obtain a solution by randomized rounding based on the solution of SDP. Semidefinite programming relaxation is one of the powerful tools to obtain approximate solutions for several NP-hard problems including MAX-CUT and Vertex Coloring [19], [20].

Consider the following vector programming (VP), which we will convert to SDP later. For ease of presentation, we only consider the case when $K = 2$ and $C_G = 3$. The results can further be extended for general K. We have two types of m-dimensional[5] unit vectors: X_v for each vertex v and Y_e for each edge e.

VP:

$$\min \sum_{e_1,e_2} \frac{1}{3}(2Y_{e_1} \cdot Y_{e_2} + 1) \text{ for } e_1,\, e_2 \in E(v) \quad (5)$$

$$|X_v| = 1 \quad (6)$$

$$|Y_e| = 1 \quad (7)$$

$$X_v \cdot Y_e = 1 \quad \text{if } C_v = 1,\ e \in E(v) \quad (8)$$

$$X_v \cdot Y_e = 1/2 \quad \text{if } C_v = 2,\ e \in E(v) \quad (9)$$

We can relate a solution of VP to a channel assignment as follows. Since C_G is 3, at most 3 colors will appear in a channel assignment. We can map each color to a m-dimensional vector so that the dot product of each pair is exactly $-\frac{1}{2}$. That is, the angle between any pair of vectors corresponding to different colors should be exactly $\frac{2\pi}{3}$. Y_e takes the vector that corresponds to the color of edge e. If C_v is one, all edges incident to v should have the same color. Therefore, X_v can take the same vector as Y_e where $e \in E(v)$. If C_v is 2, edges in $E(v)$ can have two colors. To ensure that all edges in $E(v)$ choose from only two colors, we restict the angle between X_v and Y_e ($e \in E(v)$) to be $\frac{\pi}{3}$. Therefore, X_v goes in the middle of the vectors corresponding to two colors used by edges in $E(v)$. It is easy to check that (X_v, Y_e) satisfy constraints 8 and 9. Moreover, the objective function is exactly the same as the number of one-hop interferences in the given channel assignment since if $Y_{e_1} = Y_{e_2}$ (e_1 and e_2 have the same color), it contributes one to the objective function, and 0 otherwise. Thus the optimal solution of the VP gives a lower bound of the optimal solution in channel assignment problem for the one-hop interference model.

We now convert the VP into SDP by letting m_{pq} be the dot product of vector p and q.

SDP:

$$\min \sum a_{pq} m_{pq} + b_{pq} \quad (10)$$

$$m_{pp} = 1 \quad (11)$$

$$m_{pq} = c_{pq} \quad (12)$$

$a_{pq} = \frac{2}{3}$ and $b_{pq} = \frac{1}{3}$ if $p = Y_{e_1}$, $q = Y_{e_2}$ and $e_1,\, e_2 \in E(v)$ for some v. For any other pairs of vectors, $a_{pq} = 0$ and $b_{pq} = 0$. $c_{pq} = 1$ if $p = X_v$, $q = Y_e$ ($e \in E(v)$) and $C_v = 1$ $c_{pq} = 1/2$ if $p = X_v$, $q = Y_e$ ($e \in E(v)$) and $C_v = 2$.

It is known that semidefinite programs (SDP) can be solved in polynomial time (within any desired precision) [21], [22], [23], [24], [25], and given a solution for the SDP, we can

[4]An $n \times n$ matrix A of reals is *symmetric positive semidefinite* if and only if there exists a matrix $m \times n$ B such that $B^T B = A$. It is known that given A, such B of full row-rank can be found in time polynomial in n. In an instance of semidefinite programming, we wish to optimize a linear function of a symmetric positive semidefinte matrix A subject to linear constraints. For any positive real ϵ, a semidefinite program can be solved within an additive error of ϵ in time polynomial in the size of the input and $\log \frac{1}{\epsilon}$.

[5]Thus the VP produces $m \times n$ matrix B where $n = |V| + |E|$.

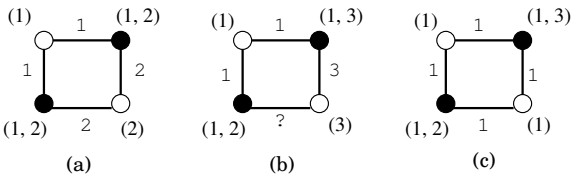


Fig. 7. Black nodes have two interface cards and white nodes have only one interface card. (a) The solution uses only two colors. The number of interferences is minimized (b) Three colors are used in the solution. There is one link that cannot be assigned to any color. (c) Three colors are used. The solution introduce more conflicts than (a).

find the solution to the corresponding VP, using imcomplete Cholesky decomposition [26].

It remains to round the solution to the VP into a feasible channel assignment. We use the same rounding technique used for MAX-CUT by Goeman and Williamson [19]. Given an optimal solution for VP, we obtain a coloring as follows. We select a vector r uniformly at random. For each edge e, assign color 1 if $Y_e \cdot r \geq 0$ and assign color 2 otherwise. We can show that the expected number of interferences in the solution obtained by the above rounding algorithm is at most 1.342 times the optimal channel assignment.

Theorem 5.5: The expected number of total conflicts by our algorithm is at most 1.342 times the optimal.

Proof: See Appendix E. ■

We can generalize the results to the case when $C_v = 1$ or k ($k > 2$) and $C_G = k$, the approximation guarantee in this case is $1 + \frac{k}{5(k-1)^2}$.

Note that in all three algorithms (greedy, centralized greedy, and SDP-based algorithm) the total number of different colors used in the network is only $\max C_V$ rather than C_G. This is because if we allow them to use colors more than $\max C_v$, it often lose connectivity or gives a solution that has more one-hop interferences. (see figure 7 for example). After obtaining the solution by the algorithm, we may further improve the solution by allowing more colors to be used while retaining all edges in the graph. We could not prove that such improvements give a better approximation gaurantee for the one-hop interference model, but in practice, we expect that it should reduce the number of two-hop interferences significantly.

VI. Simulation Study

In this section, we use simulation experiments to evaluate our proposed algorithm. We first show that compared to the single-channel case and BASIC-COLORING, the channel assignment using EXTENDED-COLORING significantly reduces the total number of interferences. Then, we use packet-level simulations to show that the decrease in the number of interferences actually can lead to network throughput improvement.

A. Channel Assignment and Interference

In this section, we first describe simulation scenarios and present the results about the number of interferences when we use EXTENDED-COLORING. We place N nodes in 1000m by 1000m square area uniformly at random, and we vary N to experiment with different network size. We also vary the transmission range R, and there is an edge between two nodes if the distance between them is less than R. We also vary the number of interface cards at each node. Table II summarizes parameters use in the simulations. For each experiment, we use the average of 20 runs with different node placement.

parameters	Values
Network Dimension	$1000\ m$ by $1000\ m$
Network Size (N)	$\{50, 100, 150, 200\}$
Tx Ranges (R)	$\{200m, 250m, 300m, 350m, 400m\}$
Interface Counts (K)	$\{1, 2, 3, 4\}$

TABLE II
PARAMETERS USED IN SIMULATIONS

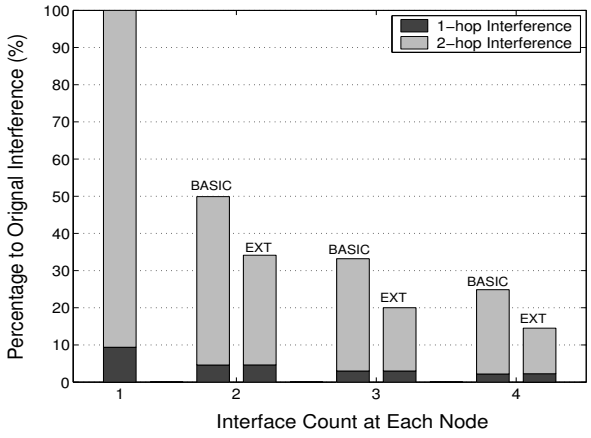


Fig. 8. The amount of interference for each scheme as the number of interfaces varies.

Experiments in Homogeneous Networks: We first experiment with homogeneous networks, where all nodes have k interface cards. In Figure 8, we plot the relative number of interferences for each scheme when compared to the single-channel case. In this set of experiments, we use 100 nodes and the transmission range of 300 meters while we vary the value of k as shown on the horizontal axis. When $k \geq 2$, EXTENDED-COLORING reduces the number of interferences significantly more than BASIC-COLORING. Specifically, when $k = 4$, EXTENDED-COLORING decrease the number to around $14.5\% = 1/6.89$, while the reduction using BASIC-COLORING is around $24.9\% = 1/4.01$. In this case (k=4), EXTENDED-COLORING uses seven channels in the system, and BASIC-COLORING uses only four channels. We can observe that the use of more channels in EXTENDED-COLORING is closely related to the amount of reduction. We also plot the number of one-hop interferences in Figure 8, and EXTENDED-COLORING and BASIC-COLORING have similar values.

In Figure 9, we plot the number of interferences of each scheme when we vary the number of nodes in the network. We fix the transmission range at 300 meters, and all nodes have three interface cards. Since we use the fixed area size of 1000m by 1000m and transmission range, the increase in the number of nodes corresponds to increase in the average

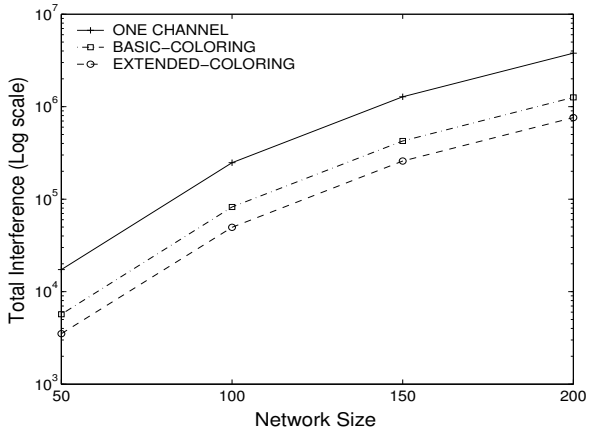


Fig. 9. The number of interferences when we vary the number of nodes in the network. We fix the transmission range at 300m, and all nodes have three interface card. Note that the Y-axis in log-scale.

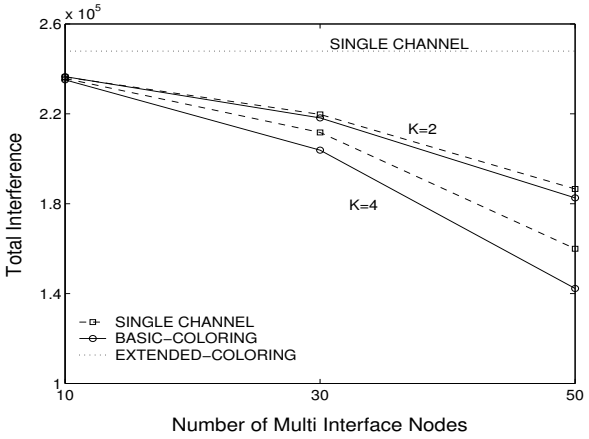


Fig. 11. The number of interferences as we vary the percentage of nodes with multiple interfaces.

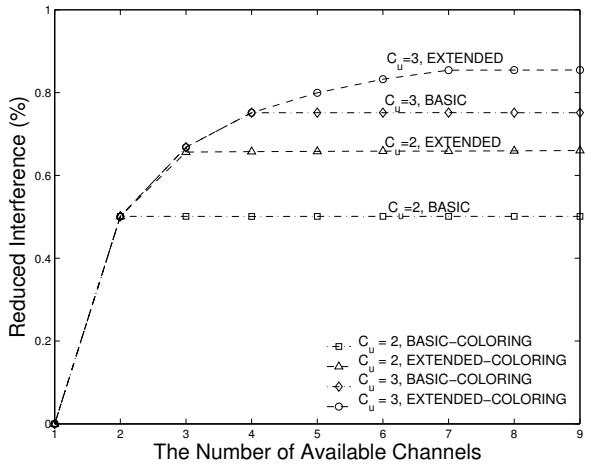


Fig. 10. The Percentage of reduced interferences as the number of available independent channels increases.

number of neighbors. As a result, the total number of conflicts increases as the network size increases. In the simulation, compared to the single-channel case, the percentage of conflict reduction using both schemes stays almost constant, regardless of the network size: 20.4% for BASIC-COLORING and 33.3% for EXTENDED-COLORING. We performed experiments using different network settings, and for a fixed k, the performance gain using EXTENDED-COLORING was similar across various settings.

In the next experiments, we examine the performance when there is not a sufficient number of channels for EXTENDED-COLORING. In Figure 10 we show the reduced interference by the channel assignment algorithms as the number of available independent channels (denoted by C_G) increases from 1 to 9. The lower two lines represents the case when all the nodes are equipped with two interface cards ($K = 2$), and the higher two lines represents when the nodes have four interface cards ($K = 4$). When $K = 2$, although BASIC-COLORING reduces up to 50% of original interferences until $C_G = 2$, the reduction of interference does not grow more as C_G increases more. However, with the same number of interfaces per node , EXTENDED-COLORING can use up to three channels, resulting more reduction of interferences up to 66%. When $K = 4$, similarly, the performance of BASIC-COLORING stops growing at 75% reduction when $C_G = 4$ while EXTENDED-COLORING reduces the interference until $C_G = 7$ reducing interference by up to 85 %. For example, when eight independent channels are available in the network such as in 802.11a, and when the mobile stations are equipped with four interface cards, EXTENDED-COLORING algorithm utilizes seven different channels and reduces 10% more interferences than BASIC-COLORING algorithm.

Experiments in General Networks: We briefly present the results when we use EXTENDED-COLORING in general networks. Due to the space constraint, we report the results when we use 1-or-k networks, where all nodes have either one interface or k intefaces. In Figure 11, we plot the number of interferences when we vary the proportion of nodes with k interfaces. We use $k = 2, 4$ and 100 nodes while the transmission range is 300 meters. The dotted line shows the original interference, which is reduced to dashed lines by BASIC-COLORING and to solid lines by EXTENDED-COLORING algorithms. In this figure, we observe that EXTENDED-COLORING always outperforms BASIC-COLORING, and the performance gap between EXTENDED-COLORING and BASIC-COLORING grows as more nodes have multiple interface cards. We also experimented using different scenarios: random assignment of interface count between 1 and $k = 4$ with varying proportions for each count. In all cases, EXTENDED-COLORING outperformed BASIC-COLORING as well as the single-channel case.

B. Packet-level Simulations

In this section we present the results of simulation experiments using *ns-2*.[6] The main goal of this packet-level simulation is to see the relationship between our proposed metric in Eq. 1 and the network performance in practice.

[6]Available at http://www.isi.edu/nsnam/ns.

We have modified the simulation code such that each node has multiple wireless interfaces. Also, each node periodically broadcasts the information on the number of its interfaces and the channel usage. Depending on this information obtained from neighbors, a node assigns a channel to each wireless link as described in Section IV. We modify the simulation code for the logical link layer such that packets sent to neighbors use the appropriate network interfaces according to the above channel assignment. In our simulations, we assume that sufficient number of channels are available in the system, and focus on the case where all nodes are equipped with the same number of interface cards.

We use the AODV (Ad hoc On Demand Distance Vector) routing protocol to find end-to-end paths [27]. According to one of our design goals in Section III, we do not modify the AODV protocol in our simulations. However, upon receiving broadcast control messages (e.g., route request messages), our modified link layer replicates and sends them using all wireless interface cards the node has. We report results when we place 50 stationary nodes uniformly at random on a 1000m by 1000m square and use a fixed transmission range of 250 meters for all nodes. We also experimented in different environments (e.g., node density, network size), and the results were similar. We use stationary nodes to minimize the impact of routing overhead due to route failures.

We select ten source-destination pairs uniformly at random among the pairs that require at least two intermediate nodes along the path (i.e., the distance between source and destination is more than 500 meters). Each source generates a CBR (Constant Bit Rate) flow using 1024-byte UDP packets. In the simulations, we vary the packet frequency of each source and examine the average data delivery ratio and end-to-end latency with different network traffic load. For example, if each source sends ten packets per second, then the aggregate end-to-end data rate becomes 800 Kbps. Each flow starts between 10 and 30 seconds chosen uniformly at random and ends at 90 seconds.[7] We use the IEEE 802.11b standard for the underlying MAC layer protocol [1], and the underlying data transmit rate is fixed at 2 Mbps for all nodes. We use average values of ten runs with different node placement.

In the first experiment scenario, all nodes have four interface cards, and we vary the packet rate of each source and investigate the data delivery ratio of each scheme. In Figure 12, we plot average data delivery ratios when we increase the total packet transmission rate. In the signle-channel case, the used channel becomes saturated even with low data rate. As a result, the delivery ratio becomes lower than 80% in the case of 320Kbps data rate, and around 20% when the data rate becomes 1.2Mbps. In contrast, BASIC-COLORING and EXTENDED-COLORING use four interfaces available at each node and maintain high delivery ratios for significantly increased traffic load.

In Figure 12, we can observe that EXTENDED-COLORING outperforms BASIC-COLORING. When the data rate is more than 800Kbps, BASIC-COLORING delivers less than 90% of data packets. In contrast, using EXTENDED-COLORING, we

[7]We also experimented using longer durations, and the results were similar.

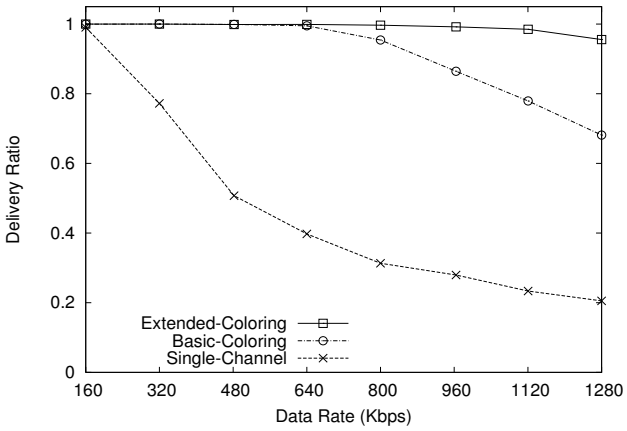


Fig. 12. The average data delivery ratio when we vary the data rate. Each node has four interface cards.

	Data rate (Kbps)					
	160	320	480	640	800	960
BASIC-COLORING	31.4	34.2	37.0	65.2	225.1	498.3
EXTENDED-COLORING	30.3	31.3	35.1	36.2	40.1	75.5

TABLE III

AVERAGE END-TO-END DELAY (IN MS) WHEN WE VARY THE DATA RATE.

can maintain the data delivery ratio higher than 95% up to the data rate of 1.2Mbps. This is because the use of more channels in EXTENDED-COLORING (seven against four in BASIC-COLORING) lowers the total number of interferences among links, which allows more concurrent transmission to succeed. For a similar experiment setting, compared to BASIC-COLORING, the results in Figure 8 show that EXTENDED-COLORING lowers the number of interferences to 14.5% = 1/6.89, while in Figure 12, EXTENDED-COLORING can support around seven times as high data traffic load as the single-channel case. From these results, we can see that there is close relationship between the reduction in the number of interference and network throughput improvement.

In Table III, we tabulate the average end-to-end delays for BASIC-COLORING and EXTENDED-COLORING using the same experiment scenarios as in Figure 12. When the data traffic load is relatively low, EXTENDED-COLORING achieves lower latency than BASIC-COLORING. Specifically, while both schemes deliver more than 99% of data packets, compared to BASIC-COLORING, the average latency of EXTENDED-COLORING is around 10% lower in the case of 320 Kbps, and around 45% lower in the case of 640 Kbps. We also can observe that when the data rate increases from 640 Kbps to 800 Kbps, the average latency of BASIC-COLORING grows four times as high, which shows that the network starts to become saturated.

Varying the Number of Interfaces: We next investigate the throughput performance when we vary the number of interfaces at each node. For a given number of interfaces, we increase the data rate by an increment of 160 Kbps and examine how much data traffic each scheme can support. We determine that the scheme can support the data rate if it

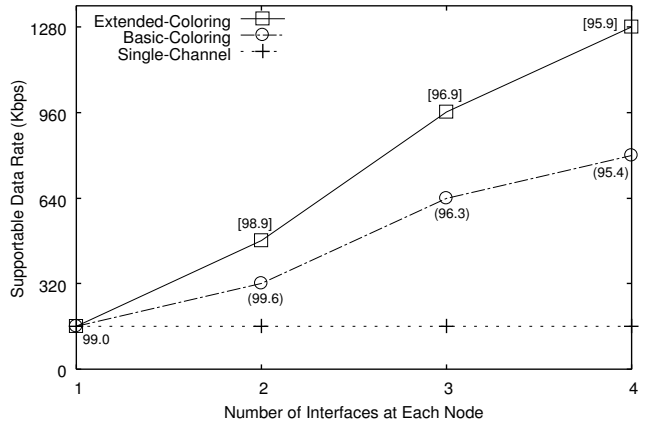


Fig. 13. The maximum data rate in which each scheme can deliver more than 95% of data packets. On the horizontal axis, we vary the number of interface cards at each node, and the numbers in the plot denote the corresponding delivery ratios in percentage.

delivers more than a certain threshold, and in our simulations, we use 95% as the decision threshold value.

In Figure 13, we show the maximum data rate that each scheme can support as we vary the number of interfaces at each node. The numbers in the plot denote the corresponding average delivery ratios (in %). As expected, the network can support more amount of data traffic as nodes have more interface cards. Again, EXTENDED-COLORING can support higher data rate than BASIC-COLORING, which again agrees with the results shown in Figure 8. Specifically, when there are two interface cards at each node, EXTENDED-COLORING uses three independent channels, and BASIC-COLORING uses only two. In Figure 13, compared to the single-channel case, EXTENDED-COLORING achieves three times as high network capacity, while the improvement ratio of BASIC-COLORING is around two. From this result we can infer that in EXTENDED-COLORING, the reduced number of interferences by using more channels is closely related to the improvement in network throughput.

To summarize, the packet-level simulation results show that our proposed scheme to reduce the total interference can lead to network throughput improvement in practice.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we have addressed the problem of minimizing radio interference in wireless ad hoc networks, where nodes are equipped with multiple interface cards. We formulate the problem as the channel assignment problem for minimizing the total number of interferences among communication links. We show that the problem is NP-hard and propose a distributed heuristic called EXTENDED-COLORING. In addition, we present centralized approximation algorithms with constant approximation factors. Through simulations, we show that our scheme significantly reduces the total number of interferences, which indeed leads to improvement in network throughput.

In the future, we will further investigate the performance of our adaptive algorithm (e.g., convergence time, control overhead) In our current EXTENDED-COLORING, we use only $(2k\text{-}1)$ channels when k is the maximum interface count number. We plan to investigate whether we can use more channels without violating the constraint in a distributed manner. Currently, our work treats all edges equivalently, and in the future, we want to generalize the current scheme to differentiate edges according to their weight. Also we plan to study the combination of our channel assignment algorithm with topology control schemes so that we can relax node connectivity constraints and allow more performance improvement.

REFERENCES

[1] IEEE, "Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," IEEE 802.11 Standard, 1999.

[2] Jungmin So and Nitin H. Vaidya, "Multi-channel MAC for ad hoc networks: handling multi-channel hidden terminals using a single transceiver," in *Proceedings of ACM MobiHoc*. 2004, ACM Press.

[3] Ranveer Chandra, Paramvir Bahl, and Pradeep Bahl, "Multinet: Connecting to multiple IEEE 802.11 networks using a single wireless card," in *Proceedings of Infocom*, March 2004.

[4] Jiandong Li, Zygmunt J. Hass, Min Sheng, and Yanhui Chen, "Performance evaluation of modified ieee 802.11 mac for multi-channel multihop ad-hoc networks," in *17th International Conference on Advanced Information Networking and Applications (AINA)*, March 2003.

[5] A. Adya, P. Bahl, J. Padhye, A. Wolman, and L. Zhou, "A multi-radio unification protocol for ieee 802.11 wireless networks," Tech. Rep., Microsoft Technical Report, MSR-TR-2003-41, June 2003.

[6] Ashish Raniwala, Kartik Gopalan, and Tzi cker Chiueh, "Centralized channel assignment and routing algorithms for multi-channel wireless mesh networks," *SIGMOBILE Mobile Computing and Communications Review*, vol. 8, no. 2, pp. 50–65, 2004.

[7] N. Jain and S. Das, "A multichannel csma mac protocol with receiver-based channel selection for multihop wireless networks," in *Proceedings of the 9th International Conference on Computer Communications and Networks (IC3N)*, Oct. 2001.

[8] Wing-Chung Hung, K. L. Eddie Law, and A. Leon-Garcia, "A dynamic multichannel mac for ad-hoc lan," in *21th Biennial Symposium on Communications*, Apr. 2002.

[9] Rajmohan Rajaraman, "Topology control and routing in ad hoc networks: a survey," *ACM SIGACT News*, vol. 33, no. 2, pp. 60–73, 2002.

[10] Kamal Jain, Jitendra Padhye, Venkata N. Padmanabhan, and Lili Qiu, "Impact of interference on multi-hop wireless network performance," in *Proceedings of MobiCom*. 2003, pp. 66–80, ACM Press.

[11] A. Nasipuri, J. Zhuang, and S. R. Das, "A multichannel CSMA MAC protocol for multihop wireless networks," in *Proceedings of IEEE WCNC*, September 1999.

[12] Richard Draves, Jitendra Padhye, and Brian Zill, "Routing in multi-radio, multi-hop wireless mesh networks," in *Proceedings of MobiCom*. 2004, pp. 114–128, ACM Press.

[13] Sven Oliver Krumke, Madhav V. Marathe, and S. S. Ravi, "Models and approximation algorithms for channel assignment in radio networks.," *Wireless Networks*, vol. 7, no. 6, pp. 575–584, 2001.

[14] V. S. Anil Kumar, Madhav V. Marathe, Srinivasan Parthasarathy, and Aravind Srinivasan, "End-to-end packet-scheduling in wireless ad-hoc networks," in *Proceedings of SODA*. 2004, pp. 1021–1030, SIAM.

[15] Roger Wattenhofer, Li Li, Paramvir Bahl, and Yi-Min Wang, "Distributed topology control for power efficient operation in multihop wireless ad hoc networks," in *IEEE INFOCOM*, Apr. 2001.

[16] Ning Li, Jennifer Hou, and Lui Sha, "Design and analysis of an MST-based topology control algorithm," in *IEEE Infocom*, Apr. 2003.

[17] M. R. Garey and D. S. Johnson, *Computer and Intractability: A Guide to the Theory of NP-completeness*, Freeman, 1979.

[18] Ian Holyer, "The np-completeness of edge-coloring," *SIAM J. Computing*, vol. 10, no. 4, pp. 718–720, 1981.

[19] M. X. Goemans and D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *Journal of the ACM*, vol. 42, pp. 1115–1145, 1995.

[20] D. Karger, R. Motwani, and M. Sudan, "Approximate graph coloring by semidefinite programming," *Journal of the ACM*, vol. 45, pp. 264–265, 1998.

[21] F. Alizadeh, "Interior point methods in semidefinite programming with applications to combinatorial optimization," *SIAM Journal on Optimization*, vol. 5, no. 1, pp. 13–51, 1995.

[22] M. Grotschel, L. Lovasz, and A. Schrijver, "The ellipsoid method and its consequences in combinatorial optimization," *Combinatorica*, vol. 1, pp. 169–197, 1981.
[23] M. Grotschel, L. Lovasz, and A. Schrijver, "Geometric algorithms and combinatorial optimization," *Springer-Verlag*, 1987.
[24] V. Nesterov and A. Nemirovskii, "Self-concordant functions and polynomial time methods in convex programming," *Central Economical and Mathematical Institute, U.S.S.R. Academy of Science, Moscow*, 1990.
[25] V. Nesterov and A. Nemirovskii, "Interior-point polynomial algorithms in convex programming," *SIAM*, 1994.
[26] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, MD, 1983.
[27] C. E. Perkins and E. M. Belding-Royer, "Ad hoc on-demand distance vector (AODV) routing," in *IEEE Workshop on Mobile Computing Systems and Applications*, Feb. 1999.
[28] V. G. Vizing, "On an estimate of the chromatic class of a p-graph (russian)," *Diskret. Analiz.*, vol. 3, pp. 25–30, 1964.

APPENDIX

A. Proof of Theorem 4.1

Proof: Note that $C_u \leq C_v$ since u belongs to $T(v)$. The color set $S(v)$ includes at least C_u colors from set $\{1, \ldots, 2C_u - 1\}$ since $N_{C_u}(v)$ is not empty. Also the size of $S(u)$ is C_u and $S(u) \subset \{1, \ldots, 2C_u - 1\}$. By the pigeonhole principle, $S(v) \bigcap S(u) \neq \emptyset$. ■

B. Proof of Theorem 5.2

Proof: Suppose that we have a simple graph $G = (V, E)$. It is known that finding the chromatic index $\chi'(G)$ of G is NP-hard (chromatic index is the minimum number of colors for edge-coloring G) [18]. By the Vizing's theorem [28], the chromatic index of G is Δ or $\Delta + 1$ where Δ is the maximum degree of any vertex $v \in V$.

Given ϵ, let $G' = (V', E')$ be the graph which has $|E|^{1/\epsilon - 1}$ copies of G. Note that $|E'| = |E|^{1/\epsilon}$. We set $C_G = C_v = \Delta$. If $\chi'(G) = \Delta$ then the optimal solution of the channel assignment problem in the one-hop interference model will be 0. Otherwise if $\chi'(G) = \Delta + 1$, then each of component of G' has at least one conflict and therefore, the optimal solution has at least $E^{1/\epsilon - 1}$ conflicts, which is the same as $|E'|^{1-\epsilon}$. Thus if we have an approximation algorithm with additive term of $o(|E'|^{1-\epsilon})$ for a graph $G' = (V, E')$, we can decide the chromatic index of G. Contradiction. ■

C. Proof of Theorem 5.3

Given an instance C of the problem 3SAT, we construct a graph G, in which each node has the balanced assignment if and only if C is satisfiable (by balanced assignment, we mean that colors are evenly distributed to edges in $E(v)$). In a channel assignment of G, a pair of edges is said to be $true$ if the edges are assigned the same channel, and $false$ if the edges are assigend different channels. The pairs of edges we consider are shown inside rectangles in Fig. 14 for example. We need three components for the reduction – inverting, variable setting, satisfaction testing. In the figures, black nodes have 2 interfaces and white nodes have only one interface.

In the inverting components (Fig. 14), one can easily see that black nodes have the balanced assignment if and only if the input pair of edges is true and the output pair is false, or vice versa. These components are used to obtain the negation of variables. In the variable setting components (Fig. 15), all the pairs of edges in the rectangles should be the same - either true or false. We can use as many output pairs as there are appearances of variable v_i or $\bar{v}_i$ (combined with the inverting components). For each clause c_j in C, we have one satisfaction testing component (Fig. 16). It can be checked that black nodes in the satisfaction testing components have the balanced assignment if and only if at least one of three pairs of edges corresponding to literals in c_j is true. Combining all the components, we can construct a graph G, in which all nodes have balanced assignments if and only if C is satisfiable.

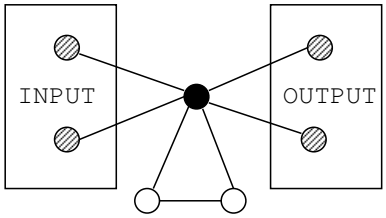


Fig. 14. Inverting componet

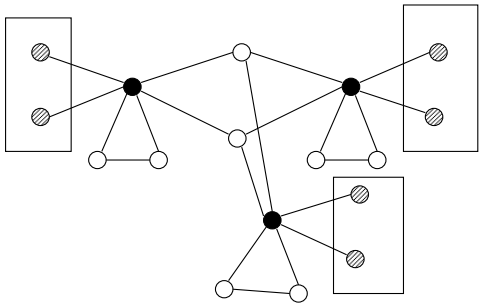


Fig. 15. Variable setting component

D. Proof of Theorem 5.4

Proof: Edges (u, v) both of which endpoints belong to V_1, should use the same color as any other adjacent edges no matter what algorithm we use. For edges (u, v) neither of which endpoints belong to V_1, we assign colors using the simple greedy algorithm discussed in Section V-A. Using the analysis similar to Section V-A, we can show that the average number of interferences each edge creates is at most one more than the optimal (therefore, at most $|E|$ more than the optimal in total).

We only need to analyze the number of conflicts created by edges in B_1^i (recall that we defined $B_1^i \subset E_1^i$ to be edges one of which endpoint is in V_k). Suppose that the algorithm assign color c to E_1^i. Recall that for an edge $e = (u, v)$ where $u \in V_1^i$ and $v \in V_k$, $n_c(v)$ is the number of edges $(u', v) \notin E_1^i$ with color c. Since we choose a color c that minimizes $\sum_{e=(u,v) \in B_1^i} n_c(v)$, we have

$$\sum_{e=(u,v) \in B_1^i} n_c(v) \leq \sum_{e=(u,v) \in B_1^i} (d(v) - e_i(v))/k$$

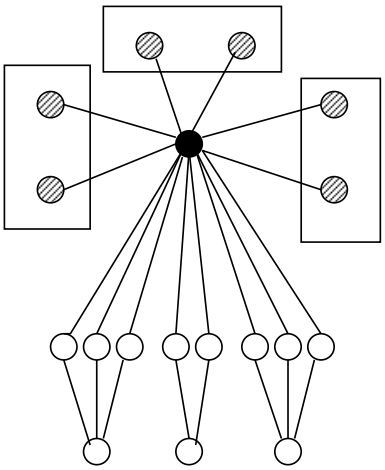


Fig. 16. Satisfaction setting component

where $e_i(v)$ be the number of edges in $E(v) \bigcap E_1^i$. Then, the number of conflicts created by edges in B_1^i is

$$\begin{aligned}
\sum_{e \in B_1^i} I_e &= \sum_{\substack{(u,v) \in B_1^i \\ v \in V_k}} (n_c(v) + e_i(v) - 1) \\
&\leq \sum_{\substack{(u,v) \in B_1^i \\ v \in V_k}} \left(\frac{d_v - e_i(v)}{k} + (e_i(v) - 1) \right) \\
&= \sum_{\substack{(u,v) \in B_1^i \\ v \in V_k}} \left(\frac{d_v}{k} - 1 \right) + (1 - \frac{1}{k}) \cdot \sum_{\substack{(u,v) \in B_1^i \\ v \in V_k}} e_i(v)
\end{aligned}$$

Note that both $\sum_{(u,v) \in B_1^i, v \in V_K} (d_v/k - 1)$ and $\sum_{(u,v) \in B_1^i, v \in V_k} e_i(v)$ are lower bounds of the optimal solution. Therefore we have $(2 - 1/k)$-approximation. ■

E. Proof of Theorem 5.5

Lemma 1.1: [19] For $-1 \leq t \leq 1$, $\frac{\arccos t}{\pi} \geq \frac{\alpha}{2}(1 - t)$. where $\alpha > .87856$.

proof of Theorem 5.5: Let X_{ij} be 1 if e_i and e_j have the same color for $e_i, e_j \in E(v)$. For any vertex v with one interface, all edges in $E(v)$ should be the same color in any solution. Therefore, we only consider edges $e_i, e_j \in E(v)$ when $C_v = 2$. The total number of such conflicts is $X = \sum X_{ij}$.

$$\begin{aligned}
E[X] &= \sum E[X_{ij}] \\
&= \sum Pr(C_{e_i} = C_{e_j}) \\
&= \sum (1 - Pr(C_{e_i} \neq C_{e_j})) \\
&= \sum (1 - 2Pr(Y_{e_i} \cdot r \geq 0,\ Y_{e_j} \cdot r \leq 0)) \\
&= \sum (1 - \frac{\arccos(Y_{e_i} \cdot Y_{e_j})}{\pi}) \\
&\leq \sum (1 - \frac{\alpha}{2}(1 - Y_{e_i} \cdot Y_{e_j})) \\
&= \sum (1 - \frac{3\alpha}{4} + \frac{3\alpha}{4}(\frac{2}{3} Y_{e_i} \cdot Y_{e_j} + \frac{1}{3}) \\
&\leq (2 - \frac{3\alpha}{2}) OPT + \frac{3\alpha}{4} OPT \\
&\leq 1.342 OPT
\end{aligned}$$

We obtain the first inequality by Lemma 1.1 and the second inequality comes from the fact that for any pair of edges in $E(v)$, the probability that they have the same color in an optimal solution is at least $1/2$ (as $C_v = 2$) and also from the lower bound given by SDP.