Distributed Channel Management in Uncoordinated Wireless Environments

Arunesh Mishra[1] Vivek Shrivastava[1] Dheeraj Agarwal[1] Suman Banerjee[1*] Samrat Ganguly[2]

[1]University of Wisconsin, Madison, USA. {arunesh,viveks,dheeraj,suman}@cs.wisc.edu

[2]NEC Laboratories, Princeton, NJ, USA. samrat@nec-labs.com

ABSTRACT

Wireless 802.11 hotspots have grown in an uncoordinated fashion with highly variable deployment densities. Such uncoordinated deployments, coupled with the difficulty of implementing coordination protocols, has often led to conflicting configurations (e.g., in choice of transmission power and channel of operation) among the corresponding Access Points (APs). Overall, such conflicts cause both unpredictable network performance and unfairness among clients of neighboring hotspots. In this paper, we focus on the fairness problem for uncoordinated deployments. We study this problem from the channel assignment perspective. Our solution is based on the notion of channel-hopping, and meets all the important design considerations for control methods in uncoordinated deployments — distributed in nature, minimal to zero coordination among APs belonging to different hotspots, simple to implement, and interoperable with existing standards. In particular, we propose a specific algorithm called MAXchop, which works efficiently when using only non-overlapping wireless channels, but is particularly effective in exploiting partially-overlapped channels that have been proposed in recent literature. We also evaluate how our channel assignment approach complements previously proposed carrier sensing techniques in providing further performance improvements. Through extensive simulations on real hotspot topologies and evaluation of a full implementation of this technique, we demonstrate the efficacy of these techniques for not only fairness, but also the aggregate throughput, metrics.

We believe that this is the first work that brings into focus the fairness properties of channel hopping techniques and we hope that the insights from this research will be applied to other domains where a fair division of a system's resources is an important consideration.

Categories and Subject Descriptors

C.4 [**Computer Systems Organization**]: Performance of Systems; C.2.1 [**Computer Communication Networks**]: Network Architecture and Design—*Wireless Communication*

[*]Work partially supported by NSF Award CNS-0520152.

MobiCom'06, September 23–26, 2006, Los Angeles, California, USA.

General Terms

Measurement, Algorithms, Experimentation, Performance.

Keywords

IEEE 802.11, channel assignment, partially overlapped channels, channel hopping.

1. INTRODUCTION

Wireless hotspots have grown rapidly as a customary attraction at restaurants, cafes and shops. Such wireless networks are typically small in size (often, a single AP) and are independently owned and managed. Such uncoordinated installations often manifest themselves in highly variable AP densities in many urban areas. For these reasons, we refer to them as 'uncoordinated' wireless deployments (also referred to as 'chaotic' networks by Akella et. al. in [1]) to distinguish them from 'centralized' deployments typically found in offices and campus buildings. The uncoordinated nature of hotspots has led to unsatisfactory and unpredictable network performance for its clients.

Uniqueness of uncoordinated deployments: In this paper we primarily focus on the fairness problem in uncoordinated deployment scenarios. While association control approaches (such as [2, 3, 4]) are particularly attractive to achieve fairness goals in centralized deployments, they are not feasible in uncoordinated deployment for the following reason. Clients of a given AP (from one hotspot, say a coffee-shop) may not be authorized to connect to another AP (of a neighboring hotspot, say a nearby restaurant). In particular, the model of uncoordinated deployment poses a few other constraints on the nature of the problem as well as the solutions that are practical. They are:

- (a) It is not possible to improve client performance in this environment through careful AP placement or site surveys, an approach which is quite commonly employed for WLAN deployments in offices.

- (b) The proposed solution should be simple, i.e., require minimal to zero coordination between APs or their clients. This is because the APs and their clients are effectively in different management domains and may have no explicit way of interacting with one another.

- (c) We posit that the desired notion of fairness in uncoordinated deployments is fairness aggregated at the AP level, and not at the level of individual clients. Essentially if two APs (1 and 2) are in close proximity of each other, then we call the performance perfectly fair if the aggregated throughput of all clients of AP 1 is identical to the aggregated throughput of all clients of AP 2. We make this hypothesis due to the following two reasons, one philosophical and the other practical: (i) 802.11 hotspots operate in unli-

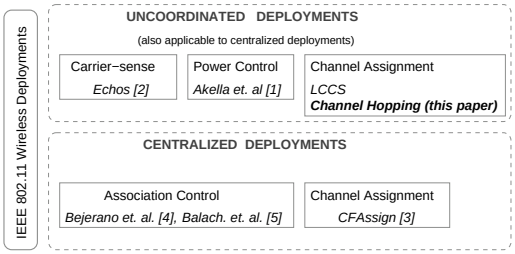


Figure 1: Complementary Methods of improving 802.11 performance. The focus of this paper is on uncoordinated deployments.

censed bands and hence, in absence of explicit market mechanisms, no hotspot should have greater priority on the the total bandwidth over another, irrespective of the number of clients, and (ii) Providing proportional fairness in these environment is certainly possible if APs (and their clients) estimate relative client set sizes of nearby APs. However, these mechanisms will require additional coordination between APs/clients across different management domains, which is not too practical. A specific danger of empowering APs with specific control knobs over its fair share is that it may itself escalate selfish behavior in the environment, requiring more complex game-theoretic formulations and solutions. In the rest of the paper, we therefore, focus on the AP-level fairness problem, as relevant to uncoordinated deployments.

There are multiple complementary ways of addressing performance issues in 802.11 networks, as summarized in Figure 1. First is power control. Proper management of transmit power reduces interference and improves network performance on the whole. Akella et. al. [1] propose a dynamic power management technique to reduce interference in chaotic deployments, while the *Echos* system [5] uses careful carrier sense mechanisms at the receiver to eliminate unnecessary interference. A second, is to perform careful assignment of wireless channels to mitigate interference among neighboring APs, e.g., LCCS (described later in this paper). A third approach is to use association control, which balances client-load across a set of APs [2, 3, 4]. However, as pointed out, this approach is not applicable in the context of uncoordinated deployments of APs.

Given prior work along the power control (and carrier sensing) dimension of performance optimization in uncoordinated deployments, it is natural to next examine the complementary dimension of channel assignment both in isolation, and in tandem, which is the focus of our paper. While fairness is our primary metric in this work, we also examine the (positive) impact of our proposed mechanisms on the aggregate throughput metric as well.

The following are some of the key components of this work:

Channel Hopping. In this paper, we explore the idea of channel hopping for improving fairness. APs spend a fixed amount of time on a single channel, called a *slot* and switch to a subsequent channel as given by its *hopping sequence*. All clients associated to an AP switch channels along with it. The specific channel hopping algorithm proposed in this paper, called MAXchop, is easy to implement through trivial extensions to existing client-AP synchronization mechanisms that are already available in the 802.11 standard.

Due to lack of a sufficient number of channels in the crowded unlicensed spectrum, even an optimally computed Max-Min static assignment of channels will end up being favorable to some APs over the others. This causes unfairness in throughput among competing APs. Our application of the channel hopping technique allows the network as a whole to timeshare between different static channel assignments. As a result, the long-term throughput obtained at an AP becomes an average of the throughputs achieved during individual channel assignments used by the network as a whole. In this manner, no single AP suffers for long due to an unfavorable channel assignment. Intuitively, this improves fairness in throughput among interfering APs. It is this unique property of channel hopping that we bring into focus in this paper and show that it can provide significant improvement in fairness over existing channel assignment mechanisms for uncoordinated deployments. We study this in Section 3.

We note that in [6], Bahl et. al. propose a protocol called SSCH, that uses channel hopping for capacity improvement in a wireless ad-hoc network where nodes have a single interface. In their work, channel hopping allowed nodes operating on different channels to synchronize due to overlap in their hopping sequences and yet utilize multiple non-overlapping channels for improved network throughput. In contrast, our work in the context of uncoordinated deployments of 802.11 WLANs, is the first that explicitly exposes *the fairness properties* of channel hopping techniques as opposed to the *performance properties* of channel hopping exploited in [6]. The fact that channel hopping can improve fairness in a fully distributed manner over some good centralized techniques is the central idea explored in this paper, that we hope will be employed by future research in other wireless domains.

Switching Overhead. Channel hopping requires APs and associated clients to switch channels at the end of a slot which has an overhead associated with it. Common wireless hardware takes about 6 — 20 ms to switch to a different channel based on our measurements. This latency goes into initializing the radio and synchronizing with the new channel at the physical layer. We amortize this cost by reducing the frequency of channel hopping, that is, by increasing the duration of a single slot. For example, by using a slot period of one second or more, this amortizes the channel switch overhead over a large number of packets, such as a thousand data packets over one second assuming 11 Mbps data-rate and 1024 bytes per packet. We also note that emerging wireless cards such as Intel's Pro-Wireless [7], report a channel switch latency of under 100 μs which greatly reduces the overhead of implementing channel hopping. Through an implementation over commercial hardware, we show that the practical overheads of channel hopping are minimal and far outweigh its gains.

Impact on TCP. Channel switching can have a negative impact on TCP performance. This is because occasional packet losses while switching channels, can affect TCP's congestion window thereby reducing throughput momentarily. We address this issue as follows: (i) Channel switching is triggered by an AP during relatively low periods of activity to minimize packet losses, and (ii) channel switching is performed at a low frequency thereby reducing the overall impact of channel hopping on TCP. A slot period of one second ensures that such losses are infrequent. Through our evaluation in Sections 6.3 and 7, we show that these techniques have negligible adverse effects on TCP performance.

Partially Overlapped Channels. The popular 2.4 GHz band provides for only three non-overlapping channels which are insufficient for dense wireless hotspots. Recent work [8] has shown that *by careful measurement and modeling* interference among the 11 "partially-overlapped" channels along with *careful channel assignment and coordination* between interfering APs it is possible to achieve significant improvement in performance. Such an approach would normally be difficult in uncoordinated deployments. It turns out that the specific nature of the proposed channel hopping approach is a very effective way to leverage partially overlapping

channels in uncoordinated deployments with zero coordination, which otherwise would require significant coordination and planning.

Client-driven Assignment. Prior work [4] has shown that client participation in estimating interference is crucial in finding APs that are 'hidden' and yet interfere at clients positioned unfavorably. While not a focus, In our paper we also explore a variant of MAXchop that uses client-based estimates of interference from neighboring APs when building hopping sequences. We show that MAXchop can take advantage of client-feedback in estimating hidden APs in uncoordinated wireless hotspots. This provides improvement in both throughput and fairness. We study these extensions in Section 6.3.

Key Contributions

- We show that our proposed idea of channel hopping has good fairness properties and is thus well-suited for uncoordinated wireless environments. Based on these insights, we design a channel hopping algorithm called MAXchop that hops between good static channel assignments and also utilizes partially overlapped channels. The algorithm operates in a fully distributed fashion and we show it converges provably and rapidly.
- We evaluate our MAXchop algorithm alongside existing centralized and distributed channel assignment techniques which apply to centralized deployments. We perform these comparisons through packet level simulations over real hotspot network topologies for multiple different cities obtained from the *Wigle* hotspot database. Through such simulations, we observe that MAXchop improves client fairness in throughput as measured by the Jain's fairness index by 35 - 60 % depending on the density of the hotspots. We also study how MAXchop can be deployed in an incremental fashion and quantify the benefits obtained by hotspot providers with increase in the deployment population.
- We implement the MAXchop algorithm over an 802.11 testbed built from off-the-shelf WLAN chipsets, which have been commonly used in commercial APs and clients [1]. Through carefully constructed testbed topologies that closely mimic dense hotspots, we quantify the benefits and tradeoffs of implementing channel hopping algorithms and contrast them with the existing methods.

Roadmap: The rest of this paper is organized as follows. We first discuss the notion of non-overlapping and partially overlapped channels in 802.11 and the existing channel assignment techniques for WLANs in the next section. In Section 3, we present the basic concepts of channel hopping and study its fairness properties. Next, in Section 4, we discuss the MAXchop algorithm and show its convergence. We present our evaluation of MAXchop in Section 6.3 through simulations and a testbed based implementation in Section 7. We discuss related work in Section 8. We conclude in Section 9.

2. BACKGROUND

Wireless hotspots are typically implemented using 802.11 APs. Each AP typically operates on a *single,* administrator-configured wireless channel. Each client *associates* with a single AP and subsequently interacts with this AP alone, on the AP's configured channel.

As a basic design rule, APs within radio frequency (RF) range of each other are set to different 'non-overlapping' channels. Proper assignment of channels to APs is important so that the network can take full advantage of the total wireless bandwidth offered by the multiple channels.

Channel Assignment Approaches. Most APs are initialized for their channel of operation through manual input. This is inefficient as it is based on human judgment of which channel is the best. Apart from this, some AP vendors implement a simple distributed method commonly called the least congested channel search (LCCS) algorithm [9]. In LCCS, upon initialization, the AP scans and selects the channel that offers the least amount of congestion, for example, the channel on which there was least amount of traffic belonging to other APs and clients. Since in LCCS, APs perform a scan without involving client feedback, it is possible that two APs might not detect each other and utilize the same channel. As a result, it is possible for clients to be positioned unfavorably so as to suffer considerable interference as a hidden terminal. Figure 2(a) shows this problem. It is possible to mitigate such interference through client feedback, as shown by the (centralized) CFAssign algorithm [4] shows that client feedback in finding interfering APs can mitigate this problem. In Section 6.3, we explore a variant of our channel-hopping technique that is uses such client feedback to detect and mitigate effects of such 'hidden APs'.

Using Partially Overlapped Channels. IEEE has divided the 2.4 GHz band into 11 channels. Successive channels are spaced 5 MHz apart while each channel occupies about 44 MHz of bandwidth on either side of the center frequency. This implies that most of the channels overlap in the frequency domain, leaving only channels 1, 6 and 11 as non-overlapping. Recent work in [8] has shown that through careful assignment of channels among interfering APs, it is possible to take advantage of the 11 partially overlapped channels available in the 2.4 GHz band for further throughput gains over the three non-overlapping ones. Here, we summarize the concepts presented in [8] that enable us to extend our channel hopping algorithm to take advantage of partially overlapped channels.

Overlap among channels is captured by the notion of *Interference Factor* or I-factor for short. Denoted by $I(i, j)$ this indicates the extent of overlap in signal power among the channels i and j. Conceptually, if P_i is the received signal power of a transmission on channel i, then the same receiver would obtain a signal power of $P_j = I(i, j)P_i$ on channel j for the same transmission. I-factor can be computed in a theoretical fashion using expressions for a transmitted signal's power spectral density, or the spread of a signal's power across the frequency domain. It can also be estimated using signal strength measurements as performed in [8].

Obtaining benefit from partially overlapped channels requires explicit interaction and coordination among interfering APs [8]. This is necessary so that partially overlapped channels are assigned carefully to yield throughput gains. In this paper, we show that channel hopping can take full advantage of partially overlapped channels in a distributed manner without the need for any coordination between interfering APs. We study these properties in Section 4.

3. BASIC CONCEPTS

Uncoordinated wireless deployments exhibit certain important characteristics that affect how we design channel assignment algorithms for them. First, they have high and variable densities. Prior evaluations by [1] based on AP data from *Wifimaps.com* and Intel's Place Lab have shown that uncoordinated environments exhibit high AP densities especially in urban areas. Also, there is high variation in the densities. Urban deployments have highly dense 'pockets' of APs sporadically placed. We study these properties further in Section 6.3. Second, they are uncoordinated. This implies that channel assignment methods should be distributed. This

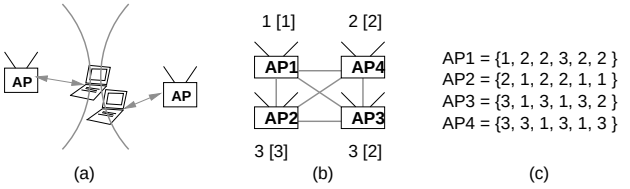


Figure 2: (a) Hidden AP problem in LCCS — Clients of the two APs interfere, but the APs, themselves, cannot detect such mutual interference; (b) an example topology and (c) its hopping sequence.

precludes possibilities such as roaming of clients to different APs [2] and balancing of client load among APs [3]. Finally, since these algorithms execute at the firmware level on APs they need to be simple and efficient.

We first present the unfairness inherent in static channel assignments when applied to such deployments. Next, we discuss the core concepts of channel hopping and show how it improves fairness among APs. We draw some important insights which form the basis for our MAXchop algorithm presented in the next section.

3.1 Drawbacks of Static Techniques

We use the term static assignments to refer to techniques that assign a fixed channel to an AP for a relatively long duration of time (such as hours or days). Such techniques typically model channel assignment as a graph coloring [10] problem and build distributed or centralized algorithms for it. Traditionally, throughput maximization and client-fairness has been the primary goal for such schemes rather than per AP-fairness. Here, we study why static channel assignment can cause unfairness in uncoordinated environments.

Consider the example shown in Figure 2(b). There are four APs which interfere with each other and we have three channels to allocate in the 2.4 GHz band. Figure 2(b) models this as a standard graph coloring problem where the vertices are the set of APs and edges represent interference. We call this the interference graph, defined next:

Definition: An undirected graph $G = (V, E)$ denotes an interference graph when V is the set of APs under consideration and an edge $(u, v) \in E$ indicates that APs u and v interfere with each other. Two APs interfere with each other if either the APs interfere directly or clients associated to these APs interfere with each other. Client participation in capturing interference addresses the hidden AP problem discussed earlier in Section 2. Based on this graph model, static channel assignment becomes a coloring problem. Next, we define the k-colorable property for a graph:

k-coloring property: Given a graph $G = (V, E)$ and a coloring $\chi : V \rightarrow \{1 \ldots k\}$ where k is the number of colors used, χ is said to have the k-coloring property iff for each edge $e = (u, v) \in E$, $\chi(u) \neq \chi(v)$. For the example of Figure 2(b), the interference graph is a four-clique as shown. Given that this graph is not three-colorable, a good channel assignment using three colors would attempt to minimize the number of APs that interfere. That is, it would minimize the number of edges $e = (u, v) \in E$ such that $\chi(u) = \chi(v)$. For Figure 2, there are multiple such assignments possible and two such solutions are shown. The channel numbers are indicated above the APs and in square brackets for the first and the second solution, respectively.

In the first assignment of Figure 2(b), APs 2 and 3 operate on the same channel. The 802.11 distributed coordination function (DCF) will allow them to share the same channel using MAC level backoffs and the RTS-CTS handshakes. However, this happens at the cost of significant reduction in throughput. For simplicity, we assume that 802.11 provides fair sharing of a channel's bandwidth [11]. Based on this, APs 2 and 3 each get $1/2$ of the normalized channel's bandwidth, while APs 1 and 4 get unit bandwidth. Thus, the first solution of Figure 2(b) performs an unfair assignment to APs 2 and 3 when compared to APs 1 and 4. However, in the second assignment shown in Figure 2(b) APs 3 and 4 get $1/2$ while APs 1 and 2 get unit band with. From this example, it is apparent that static channel assignments tend to be biased towards a certain set of APs over the others.

This unfairness in optimal static assignments happens only when the interference graph is not k-colorable, where k is the number of available channels. This follows from the fact that when the graph is k-colorable, optimal static assignment finds such a k coloring and each AP gets unit bandwidth, which is optimally fair. When such a graph is not k-colorable, optimal static assignments attempt to minimize the number of edges that use the same color/channel. The very fact that some edges use the *same channel* for a long duration of time shows that the corresponding APs suffer unfairly when compared to other APs that do not have such edges (or have a lower number of such edges).

Practically, it is hard to find an optimal static coloring. Finding such a k coloring is NP-hard for general graphs and it is even NP-hard to find a constant approximation [12]. Thus, even if a graph is k-colorable, a good static assignment algorithm will probably not find such a coloring. So, much like before, there will be edges that violate the coloring property and the corresponding APs will suffer degraded throughput when compared to others because of sharing the same channel. Thus, even if a graph is k-colorable, as long as the static channel assignment algorithm does not find such a k coloring, there could still be unfairness in the static assignments.

Based on these observations, we ask the following question : How severe will this unfairness be, or in other words, will practical interference graphs *not* be k-colorable? Considering the dense and uncoordinated nature of hotspot deployments, we expect that the interference graphs representing them will not be k-colorable, for practical values of k (three for 802.11b). Intuitively, this is because the high density in such environments makes these graphs very dense in nature. This increases the number of colors/channels needed for satisfying the k-colorable property. Thus, static channel assignments for most uncoordinated environments will tend to be unfair to some APs over others. Next, we discuss how channel hopping can improve the fairness among APs over such static channel assignments.

3.2 Channel Hopping

We have seen how static assignment can cause unfairness. In our method of channel assignment, APs use a sequence of channels and transition between them over time. We build channel hopping sequences for APs such that at any time instant, the network as a whole uses a 'good' static assignment, like the first assignment shown in Figure 2(b). But, in order to not cause any one AP to suffer for long, the network switches to a different global assignment, such as the second assignment shown in Figure 2(b). In this manner, over a long period of time, the throughputs of interfering APs get averaged out to equal values. Below, we discuss this intuition in greater detail.

Basic channel hopping is performed as follows: Time is divided into slots. Let t_s denote the duration of a time slot. APs use static channels for the entire duration of a slot. At a slot boundary, APs switch channels to their respective assignments for a subsequent slot. This requires all APs to have a synchronized notion of time

AP #	Slot Number						Avg
	1	2	3	4	5	6	
1	1	1	1/2	1/2	1	1/2	0.75
2	1	1/2	1/2	1	1/2	1	0.75
3	1/2	1/2	1	1	1	1/2	0.75
4	1/2	1	1	1/2	1/2	1	0.75

Table 1: Aggregate and per-slot throughput for Figure 2(b).

and equal slot sizes. We assume that this is the case for ease of exposition. Later in Section 5, we relax these constraints for the MAXchop algorithm. The sequence of channels that an AP uses across successive slots defines its *channel hopping sequence*. The *periodicity* of a hopping sequence is the number of slots after which the sequence repeats itself.

Consider the example of Figure 2(b). Suppose, we assign hopping sequences as shown in Figure 2(c) which utilize three channels:

Here, the periodicity of the sequences is six, i.e., the hopping pattern repeats after six slots.

Notation: Let λ denote the hopping sequences. In particular, let $\lambda(u)$ denote the hopping sequence used by AP u. In the above example, $\lambda(AP1) = \{1, 2, 2, 3, 2, 2\}$. Let λ_i denote the channel assignment for all APs in slot $i, (i = 1 \ldots N_s)$, where N_s is the periodicity of the hopping sequence. Also, let $\lambda_i(u)$ denote the channel used by AP u in slot i. For example, $\lambda_1(AP1) = 1$.

The hopping sequence shown in Figure 2(c) has been engineered such that in any fixed slot i, the channel assignment used by the network as a whole, namely λ_i, attempts to minimize interference in that slot i. In fact, this sequence utilizes one of the optimal channel assignments in each slot. Over the six slots, the network hops between all six possible optimal assignments for the topology of Figure 2(b). Thus, channel hopping methods allow this flexibility of utilizing multiple different global channel assignments at different time slots. This important property improves the per-AP fairness in throughput.

Table 1 shows the normalized throughput obtained by the setup of Figure 2(b) using the hopping sequence shown in Figure 2(c). It is evident that in each slot, there is one edge that violates the k-coloring property and the corresponding APs suffer as a result. For example, in slot 1 APs 3 and 4 get $1/2$ the throughput while in slot 3 APs 1 and 2 get $1/2$. Over six slots, each edge would have suffered interference once and each AP twice. At the end of the hopping period, each AP gets equal throughput of $3/4$.

The above example illustrates how channel hopping can improve fairness in uncoordinated and dense environments. This intuition extends to a general graph, as hopping between different 'good' channel assignments improves the fairness in the system over each individual assignment.

Security Properties. Although not the focus of this paper, channel hopping techniques have interesting security properties. By making the hopping sequences pseudo-random in nature, it becomes hard for an eavesdropper to predict which channel would be used by an AP next. The clients and APs can use a common pseudo-random generator which can be initialized based on secret keys. This can improve resistance to denial-of-service attacks as the attacker will have to spend considerable time in scanning each channel to find a specific AP. By the time the attacker determines the next channel, the AP would have switched to a subsequent channel. This technique requires a good number of channels, for example, three non-overlapping channels in the 2.4 GHz band might be insufficient. Taking into account the 5 GHz band and the partially overlapped channels available there, this brings the total number of channels to about 56 making it practical. This makes an interesting direction for future work.

Algorithm 1 MAXchop

Notations Used:
V = set of access points
$x \in V$ = current AP executing MAXchop
$N(x)$ = set of APs interfering with x
N_s = periodicity of hopping sequences
λ = represents hopping sequences
k = number of channels/colors
$\rho(a, b)$ equals one if $a = b$, zero otherwise
$\eta(j)$ = Number of interfering APs if the current AP (x) were to use channel j
t_s = Duration of a slot, eg:- one second

Initialize(x):
1: Set $\lambda_i(x) = random(1 \ldots k)$, for $i = 1 \ldots N_s$

Hop(x):
1: **for** $i = 1 \ldots N_s$ **do**
2: **for** $j = 1 \ldots k$ **do**
3: $\eta(j) = \sum_{u \in N(x)} \rho(j, \lambda_i(u))$
4: **end for**
5: $\eta_{min} = \underset{j=1 \ldots k}{MIN} \eta(j)$ /* Minimum interference value*/
6: Let $C = \{c : \eta(c) = \eta_{min}\}$ /* Set of all colors that yield η_{min} */
7: $\lambda_i(x) = ComputeMinMax(C, i)$
8: **end for**

4. THE MAXchop ALGORITHM

In this section, we describe our MAXchop algorithm based on the concepts of channel hopping presented in the previous section. This distributed algorithm utilizes information from neighboring (or interfering) APs to compute a 'good' hopping sequence. The algorithm does not require any explicit communication among neighboring APs. Also, we show that the collective set of hopping sequences for such an uncoordinated deployment of APs converges provably in $O(\delta)$ rounds where δ is the AP's degree in the interference graph – an undirected graph over the set of APs with edges denoting interference (Section 3). Later in Sections 6.3 and 7 we evaluate the performance of this algorithm through extensive simulations and a testbed based implementation.

At a high level, the algorithm works in a distributed fashion as follows. Each AP first obtains the hopping sequences of other interfering APs. It then computes its hopping sequence that maximizes its throughput. This done for each slot much like the way static channel assignments are computed. That is, for each slot, the AP chooses the channel that minimizes the number of edges which violate the k-coloring property (discussed in Section 3) and thus maximizes its throughput. It is possible that multiple channels might yield the same throughput. Now the AP can either (i) select a channel uniformly at random from all such channels, or (ii) it selects the one that distributes interference as evenly as possible among its neighbors. We show that with either approaches the algorithm converges to the same assignment in $O(\delta)$ rounds. However, with approach (ii) the convergence rate is faster. Because of space limitations, we describe approach (i) here. A reference algorithm for approach (ii) is presented in the Appendix.

The MAXchop algorithm is shown as Algorithm 1. Assume that time at an AP is divided into slots of duration t_s seconds. We define *hopping period* as the time after which a hopping sequence repeats itself. Say, if this happens after N_s slots, the hopping period is equal to $N_s t_s$. In other words, the hopping sequence used by an AP specifies the channel in use for each consecutive slot of duration t_s. This happens for a total of N_s slots, after which the sequence repeats. The algorithm consists of three routines which we discuss below.

Initialize: The Initialize routine is executed at an AP x during bootup, or periodically after a long amount of time, such as on a weekly basis. This is done in order to re-initialize the state maintained by the MAXchop algorithm after a long period of time. This routine initializes the channel assignment with a pseudo-random hopping sequence.

Hop: The second routine, Hop, is executed at the end of a hopping period, that is, after $N_s t_s$ time duration, where N_s is the hop periodicity and t_s is time duration of a single slot. This routine computes a 'new hopping sequence' to be used for the subsequent hopping period based on information about the hopping sequences of interfering APs. Implementation issues such as obtaining hopping sequence information from neighboring APs is discussed in Section 5.

ComputeMinMax: The third routine, ComputeMinMax, computes the color/channel that divides the interference equally among interfering APs, out of a set of colors that provide the best throughput for the given AP x. For ease of exposition in this section, we assume that this routine returns any such color uniformly at random. A reference algorithm for dividing the interference in a min-max fashion is presented in the Appendix.

The Hop routine gets executed at the end of each hopping period and is the central procedure of MAXchop. We explain this in further detail. Lets assume that the Hop function executes on an AP $x \in V$. In Steps 2-4, the Hop function computes the quantity $\eta(j)$, which measures the number of interfering APs if the current AP x were to use channel j. An AP would attempt to select a channel that offers the minimum interference, that is, with the minimum value of $\eta(j)$ over all channels j. Before we compute $\eta(j)$, lets define the following functions: The function $\lambda_i(u)$ denotes the hopping sequence for AP u. Specifically, for $i \in 1 \ldots N_s$, $\lambda_i(u)$ gives the channel assigned to AP u in slot i. Let $N(x)$ denote the set of APs that interfere with AP x (and are thus its neighbors in the interference graph). Based on these, $\eta(j)$ for AP x can be calculated as the number of APs in $N(x)$ which are on the same channel as AP x, that is, channel j. Thus, the quantity $\eta(j)$ for an AP x is equal to the number of APs $u \in N(x)$ such that $\lambda_i(u) = j$, where $N(x)$ is the set of APs interfering with x, and i is the current slot under consideration. This is computed in Step 3 of the algorithm.

Step 5 computes the minimum value of η as η_{min} over all colors/channels $1 \ldots k$ for that slot for AP x. This quantity represents the minimum amount of interference that AP x will suffer regardless of which channel is selected during slot i. Step 6 computes the set C of all colors that yield the minimum value of η_{min}. If $|C| > 1$, implies that the AP x has a choice of more than one channel that would yield the same amount of interference. As discussed before, at this point, the AP can either choose a color randomly out of set C or it could do it in a manner that divides the interference equally among neighboring APs. We discuss the second possibility in detail next.

Step 7 calls the function ComputeMinMax that returns a color from C such that it distributes the interference equally among all neighbors of x (see Appendix). This 'distribution' of interference among neighbors is done in a min-max fashion. Define the amount of interference between two APs over a hopping period as its *L-value*, computed as $L(u, v) = \sum_{i=1 \ldots N_s} \rho(\lambda_i(u), \lambda_i(v))$. Here, $u, v \in V$ are two APs that interfere and ρ is as defined in Algorithm 1. Out of all colors in C that yield the same amount of aggregate interference, ComputeMinMax selects the color that best distributes the L-values among the edges in a Min-Max fashion. This algorithm is provided in the Appendix. For simplicity, we can assume that ComputeMinMax returns a color/channel from C uniformly at random.

As a note, the problem of finding the optimal hopping sequence that is min-max fair is NP-hard for general graphs. This follows from the observation that a method of finding a fair solution could be used to solve the decision version of the general graph coloring problem. The algorithm presented here is a practical technique of providing better fairness compared to static channel assignments. The contribution of this paper lies in the novel usage of channel hopping techniques to improve fairness over static channel assignments as opposed to finding provably good channel assignment algorithms. It is possible to combine our channel hopping method with more sophisticated and provably good channel assignment algorithms to yield better results. This is a promising direction which is nevertheless outside the scope of this paper.

4.1 Convergence

Each AP executes the MAXchop algorithm periodically at the end of every hopping period. Even with this periodic execution, we say that MAXchop has converged if the APs uses the same hopping sequence in successive hopping periods. This assumes that the network dynamics such as client associations, ambient noise levels stay constant over this period. Convergence properties of distributed algorithms show a definitive direction towards which the network is evolving itself and thus assume significant importance. We argue convergence for MAXchop based on the following invariant: Let $\hat{L}_E$ denote the L-values of all edges arranged in non-decreasing order.

Invariant: Consider an AP x that is about to execute the Hop function in the MAXchop Algorithm. Let $\hat{preL}_E$ denote the value of $\hat{L}_E$ before execution of the Hop function. Let $\hat{postL}_E$ denote the value of $\hat{L}_E$ after execution of Hop at the AP x. Then, $\hat{preL}_E \geq_{lex} \hat{postL}_E$. This inequality follows from observing that Hop alters the value of $\hat{L}_E$ in the following manner: by considering only those colors that yield minimum interference of η_{min} in Step 6, Hop minimizes the sum of all L-values. That is, Steps 2-6 compute $\underset{j=1 \ldots k}{MIN} \sum_{\forall u \in N(x)} \rho(\lambda_i(u), k)$. A more detailed proof of convergence is sketched in the Appendix.

4.2 Utilizing Partially Overlapped Channels

In this section, we discuss how channel hopping can take advantage of partially overlapped channels without requiring explicitly interaction among interfering APs. The interested reader is referred to Section 2 for a discussion on the notion of I-factor and how partially overlapped channels can provide throughput gains in general.

The MAXchop algorithm requires minor modifications to incorporate partially overlapped channels. Recall that Step 3 in the Hop routine of Algorithm 1 calculates the η function for the current AP x as follows:$\eta(j) = \sum_{u \in N(x)} \rho(j, \lambda_i(u))$. For slot i assuming AP x were to use channel j, this function determines the number of APs $u \in N(x)$ that would interfere with AP x. With non-overlapping channels, this is calculated by the ρ function which states that they interfere iff both APs use the same channel.

We modify the ρ function to capture interference from a partially overlapped channel as follows. Specifically, define $\rho(u, i, x, j)$ to

indicate if AP u on channel i interferes with AP x on channel j. $\rho(u, i, x, j)$ could return a binary value indicating that either the two APs 'interfere' or they don't. Or, $\rho(u, i, x, j)$ could return an accurate estimate of interference as the amount of signal or noise received by one AP from another AP on a partially overlapped channel. Below, we discuss how either of these functions can be computed. Note that the MAXchop algorithm can incorporate both models of interference.

If $P_x(u)$ denotes the received power of a transmission from AP u received at AP x assuming both are on the same channel, then $P_x(u)I(i, j)$ denotes the received power for the same transmission if the transmitter was on channel i and the receiver was on channel j. Based on this, we construct the ρ function by requiring that this received power should be above a certain threshold to cause interference. Clearly, this is a binary interference model and the ρ function returns a binary value. The binary ρ function can be readily replaced with a more realistic interference model such as the following $\rho(u, i, x, j) = P_x(u)I(i, j)$. This model gives the exact amount of signal from AP u that would interfere at AP x. With these modifications, the semantics of Step 5 which uses ρ still stays the same and all other properties of the algorithm continue to hold. In the next Section, we discuss some of the practical considerations in implementing MAXchop.

5. PRACTICAL CONSIDERATIONS

In this section, we discuss the practical issues with implementing MAXchop in an uncoordinated wireless environment over off-the-shelf commodity hardware.

5.1 Implementing Channel Switching

Implementing channel switching brings up two important issues which we discuss below.

1. *Client-AP Coordination:* First, an AP has to inform all its associated clients to move to a different channel. This can be implemented as a special beacon message. Implementing it as a part of the beacon message provides the following properties. The beacon is sent periodically and at the lowest bit-rate supported by the AP. This maximizes the probability that all associated clients receive the beacon. Also, clients in power-save mode wake up to receive beacon messages. In this manner, such clients will also be aware of channel changes. Such constructs are already in place in the upcoming IEEE 802.11k draft standard on Radio Resource Management. Also, much like the way channel information is available in a beacon message today, the exact hopping sequence computed by an AP can be included as well. This allows the neighboring interfering APs to obtain each other's hopping sequence for executing MAXchop. In our implementation discussed in Section 7, we use dedicated broadcast messages to trigger channel changes.

2. *Channel Switch Overhead:* Second, we measured the channel switch latency for some of the popular 802.11 vendors. We instrumented open source drivers in GNU/Linux for the following cards, and measured the channel switch latencies in the drivers:

Vendor	Chipset	Type	Driver	Latency
ZoomAir	Prism 2.5	802.11 b	hostap	20 ms
Cisco	Aironet	802.11 b	airo_cs	10 ms
Netgear	Atheros	802.11 a/b/g	madwifi	6 ms
Intel	Intel	802.11 b/g	ipw	200 μs

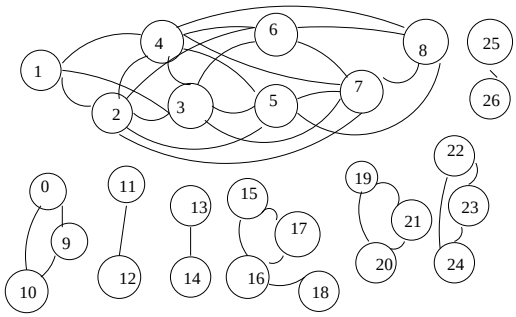


Figure 3: AP Interference graph of the 27 AP sample topology.

Although newer chipsets such as the Intel's ProWireless provide low channel switch latencies, for most others the latency is high enough to trigger packet delays and potential losses. In Sections 6.3 and 7, we study the impact of these on TCP through testbed experiments and simulations. We provision our channel hopping algorithm to perform the following: First, trigger channel switch is triggered during low periods of activity. This minimizes such losses. Second, the slot duration is chosen to be sufficiently large enough (such as one second or more) such that the overhead of switching channels gets amortized over an entire slot. Finally, considering the other significant gains obtained by channel hopping, this overhead becomes relatively small.

5.2 Estimating the APs that interfere

The MAXchop algorithm requires information on the hopping sequences of interfering APs. This can be implemented in two ways, namely, client-driven and AP-driven. Prior work by Mishra et. al. [4] has shown that client-driven estimation of interfering APs is crucial to find what are called hidden APs. We have discussed this briefly in Section 2. Our MAXchop algorithm takes as input the set of APs that interfere and their hopping sequences. Thus, both approaches can be implemented with the MAXchop algorithm. In the client-driven approach APs request associated clients to perform a scan of all channels and report interfering APs (and their hopping sequences). Also, APs themselves perform a scan to find interfering APs. Constructs to implement client-driven scanning have been included in the IEEE 802.11k draft. In the AP-driven approach, only APs perform this scan and collect necessary information. These scan operations are performed with relatively low periodicity (such as once per 30 minutes) since AP positions and topologies do not change frequently. In our evaluation presented in Section 6.3, we contrast both methods.

5.3 Asynchrony in Hopping

Different APs can have different periodicity in their hopping sequences. Also their time slots need not be aligned. This removes the need for any explicit synchronization between APs and also allows them to switch channel opportunistically (during low periods of activity). The MAXchop algorithm takes care of this as follows. The hopping patterns of an interfering AP can be transformed to best match the current APs hop cycle such that interference calculations can be performed (Step 3 of Algorithm 1). In our evaluation through simulations and a testbed based implementation, APs have an asynchronous notion of time, and thus their slot updates are not synchronized. We find that over a long period of time, MAXchop with asynchronous slots performs the same as with synchronization.

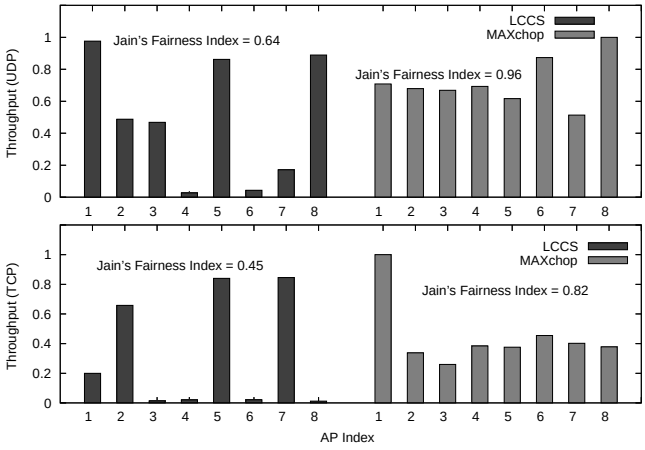


Figure 4: Throughput vectors for the 27 AP sample topology.

	Metric	*Fairness*			*Throughput*		
	Num Channels	3	4	5	3	4	5
NOV	LCCS	0.32	0.48	0.49	0.49	0.55	0.58
	RaC	0.46	0.59	0.66	**0.51**	**0.62**	**0.73**
	MAXchop	**0.48**	**0.62**	**0.69**	0.45	0.56	0.61
	Num Channels	11	16	21	11	16	21
POV	LCCS	0.37	0.51	0.65	0.72	0.74	0.81
	RaC	0.60	0.77	**0.82**	**0.89**	**0.96**	**1.00**
	MAXchop	**0.61**	**0.80**	0.81	0.80	0.86	0.99

Table 2: Normalized fairness and throughputs for the 27 AP sample topology. NOV = Using non-overlapped channels, POV= Using partially overlapped channels (TCP results).

6. SIMULATIONS

In this section, we evaluate our proposed channel hopping algorithm, MAXchop , through extensive packet-level simulations over real hotspot topologies derived from a popular hotspot database, *Wigle* [13]. We compare the performance of MAXchop against two channel assignment algorithms, namely, least congested channel search (LCCS) which is commonly implemented on APs, and a centralized algorithm, called CFAssign-Randomized Compaction (RaC) [4]. Being centralized, RaC benefits from possible interactions among APs and optimization such as load balancing. Their performance, thus, acts as a benchmark to compare against.

Through simulations over representative hotspot topologies derived from *Wigle*, we find that channel hopping improves fairness significantly over the static assignment performed by LCCS (35 — 60 % improvement). More importantly, we also observe that MAXchop achieves fairness and throughputs comparable to that achieved by the centralized RaC algorithm (executed on these topologies assuming full cooperation and coordination among APs). We also find that MAXchop is capable of utilizing partially overlapped channels in a fully distributed manner and provides additional fairness and throughput gains (42 — 60 % improvement). Due to space limitations, we present a representative set of results that bring out the above properties of channel hopping.

6.1 Hotspot Topologies

We obtained AP locations for a dense 14.7 $\times$ 1 km urban area as shown in Figure 5. Given the uneven distribution of the APs, it was easy to partition this urban area into 12 separate topologies

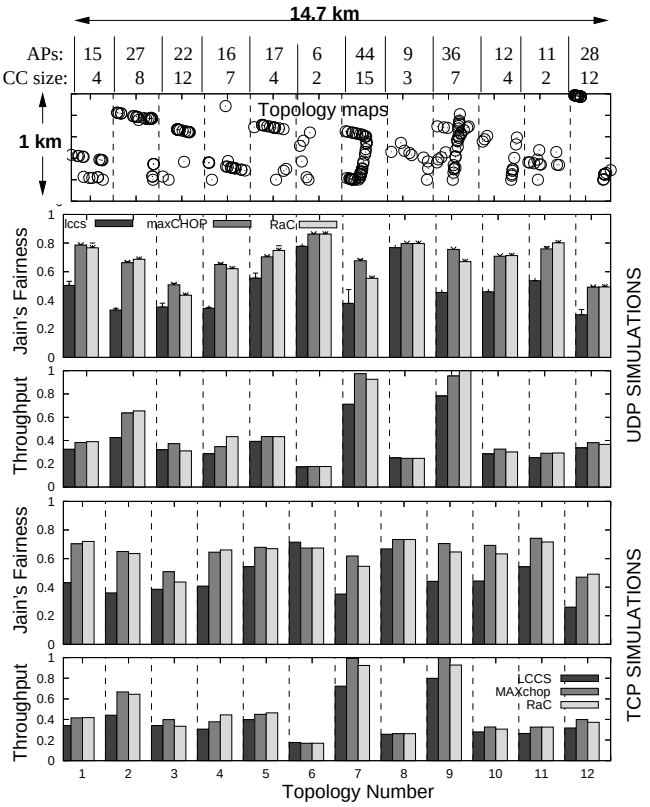


Figure 5: Simulation results over 12 real and representative hotspot topologies in a 14.7 km by 1 km urban area. The number of APs and the maximum connected component size (CC size) is listed above each topology's map. The bars, from left to right, correspond to LCCS, MAXchop**, and RaC respectively.**

such that there were no interference between any AP or client that belonged to two different topologies. Hence, for efficiency reasons we ran simulations separately for each of these topologies. It is easy to that there exists great variability in the density and AP deployment pattern of these topologies.

6.2 Simulation Methodology

We performed two sets of simulations. In the first set, we study the performance of the three channel assignment algorithms over a single representative topology (call it the sample topology, described later) drawn from the *Wigle* hotspot database. This allows us to compare the performance of the three algorithms (LCCS, MAXchop, and RaC) under similar topological conditions. In the second set, we study statistical properties of the channel assignment algorithms by simulating them over the 12 topologies that we sampled for the city of San Francisco.

All simulations were performed using the NS-2 simulator. When implementing channel hopping, the slot durations of the APs were loosely synchronized (Section 5.3). Also, to be fairly conservative, a channel switch latency of 20ms was incorporated for every switch operation. Practical costs for channel switching are discussed in Section 5. For each of the algorithms, we study the following two metrics: (i) aggregate network throughput: This allows us to study the impact of channel assignment algorithms on the overall network capacity. (ii) fairness in per-AP throughput: This is the central metric that channel hopping algorithms aim to improve in

uncoordinated wireless deployments. We measure the fairness in throughput by calculating its *Jain's* fairness index, which is given as $\frac{\sum {x_i}^2}{n \sum {x_i}^2}$, where x_i per-AP for n APs in the system. For some cases, we also plot the actual throughput vectors to visually show the unfairness in the allocations.

We used the following settings in our simulations. APs used a transmit power of 15 dBm, as commonly used by commodity APs. For LCCS and MAXchop, client to AP associations were fixed, as are expected in uncoordinated deployments. However, to faithfully implement the RaC algorithm as in [4], this restriction was relaxed in order to allow RaC to perform centralized load balancing. Also each AP had five clients associated to it on average, as was reported in prior study on urban hotspots by Akella et. al. [1]. RTS/CTS was turned off. This is the default setting in most commercial APs. The physical client-AP separations were chosen to be representative of typical separations in coffee shops. The radio propagation model used was the standard two-ray path loss for large scale paths and the Friis equation at small distances.

6.3 Results

We discuss results in two parts. In the first part we closely examine a single sample topology out of the 12. We examine performance on this topology using both non-overlapping as well partially-overlapping channels. Subsequently, we present results for the 12 topologies. Since the trend in the remaining 11 topologies with respect to non-overlapping and partially-overlapping channels is similar to our sample topology, to save space, we present an evaluation of only the partially-overlapping channels based simulations on these remaining topologies (and skip the results using non-overlapping channels).

Sample Topology: From the 12 topologies studied in this section, we selected one such topology that is representative of the variability in deployment densities. This topology shown in Figure 3 (Topology 2 in Figure 5) has 27 APs with uneven density of distribution. Eight of these APs suffer considerable interference as they are closely located, while the others had significantly less interference (average interference with 2 other APs). We performed simulations using LCCS based channel assignment and compared the throughput results to that obtained from channel hopping. Other than the 8 dense APs, the rest had similar throughputs. We thus, focus on the unfairness suffered by these APs. Figure 4 shows the individual UDP and TCP throughputs for these APs. From these bar plots, it is clear that static assignment based on LCCS causes severe unfairness in per-AP throughput. For example, APs 4 and 6 suffer considerably compared to APs 1 and 8. On the other hand, with MAXchop all the APs get similar throughput. This happens as no single AP suffers considerably for long due to an unfavorable assignment of channels. Channel hopping cycles through different global assignments providing each AP with a 'close-to' fair share of the total system bandwidth.

We next use this sample topology to present a comparison between impact of these algorithms using non-overlapping and partially-overlapped channels. Table 2 summarizes our results using 3,4 and 5 non-overlapping channels and a corresponding number of 11, 16 and 21 partially overlapped channels[1]. In each column in Table 2, the entry in bold indicates the best performance with respect to a specific metric. It follows from the table that MAXchop improves fariness over LCCS when using non-overlapping channels, but is additionally effective, when partially-overlapped channels are available. Note that in most cases (Table 2), MAXchop performs the best out of the three algorithms. In other cases, the performance of MAXchop is better than LCCS and only marginally lower than RaC. The performance gap between MAXchop and RaC remains to within 10%. This is an encouraging observation as RaC is a centralized algorithm that benefits from explicit coordination among APs. Thus, its performance acts as a benchmark to compare against.

We next focus on all 12 topologies and present results for partially-overlapped channels only, due to space constraints.

Urban Topologies: Figure 5 summarizes the results for the 12 urban topologies discussed earlier. The first panel in the figure shows the AP placements. This gives a pictorial view of the high densities along with high variations in the densities. For each topology, the plots show the fairness and throughputs for UDP and TCP flows using the three channel assignment algorithms, namely, LCCS, RaC and our MAXchop. These algorithms used the 11 partially overlapped channels based on the notion of I-factor, discussed earlier in Section 2. We draw the following observations from the results presented in Figure 5 and Table 2:

1. *Fairness gains.* Channel hopping provides good fairness in user throughput. This is evident from Figure 4, Table 2 and the Jain's fairness index plots for the urban topologies shown in Figure 5. Specifically, we note that channel hopping improves fairness over LCCS by an average of 42 %. Also, the performance of the RaC algorithm which does joint centralized channel assignment and load balancing, acts as a benchmark to evaluate against. We observe that the fairness achieved by channel hopping is very close to that of RaC. This also suggests possible applicability of channel hopping techniques to centralized deployments.

2. *Throughput gains.*Channel hopping is able to utilize partially overlapped channels without any interaction between APs. This is evident from the results in Figure 5 and Table 2 which used the 11 partially overlapped channels available in the 2.4 GHz band. When compared to LCCS, channel hopping gave an average performance improvement of 30 %. Also, its performance was comparable to RaC ($+/-5\%$).

3. *Impact on TCP.* The TCP throughputs presented in Figure 5 show that the impact of channel hopping on TCP performance gets outweighed by its gains. The fact that channel hopping yields good performance improvements despite the channel switch overheads supports this observation.

4. *Other Results.* We summarize some of the other simulations results that we obtained. We implemented the carrier-sense optimizations presented in the *Echos* system [5]. We observed that when combined with Echos, MAXchop gave an additional throughput and fairness improvement of 12% over the sample topology of 27 APs discussed earlier. Similar results were obtained using the power control techniques proposed by Akella et. al. in [1]. These results show that our channel hopping method is complementary to the optimizations presented in prior work on uncoordinated deployments that use power control to reduce interference.

7. IMPLEMENTATION

In this section, we quantify the benefits of using our proposed channel hopping technique through implementation based experiments over a testbed hotspot topology consisting of five indepen-

[1]The number of partially overlapped channels is given as $M = 5N - 4$, while keeping the same amount of spectrum bandwidth. Refer [8].

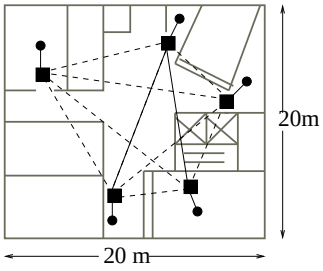


Figure 6: Design of the hotspot testbed. Solid rectangle represents an AP. Dashed edges represent interference.

dent hotspot networks. This testbed was designed to be representative of typical hotspot topologies found in urban environments. It was built from commodity 802.11 hardware commonly found in hotspots [1]. We have implemented channel hopping over this testbed using GNU/Linux and instrumented open-source drivers. All our experiment results take into account the channel switching overhead incurred at the hardware level 5 and reflect its impact on the network layers.

Through our experiments, we find that channel hopping improves fairness significantly (40 – 60 % improvement in the Jain's fairness index) among hotspots over static assignments. We also find that channel hopping is able to achieve good performance gains using partially overlapped channels in a fully distributed manner. The gains achieved by channel hopping using partially overlapped channels match the gains obtained by using centrally optimized channel assignments that use partially overlapped channels based on the concepts presented in [8]. This shows the MAXchop is capable of taking full advantage of partially overlapped channels without the need for centralized control and coordination. Apart from providing good fairness, this ability of using partially overlapped channels without explicit interaction between interfering APs is one of the key strengths of our channel hopping technique. Finally, we observe that the impact of the overheads of periodic channel switching is negligible. We find both UDP and TCP throughputs are impacted by less than 0.6 % due to channel switching. We first describe the testbed topology, the platform used for the implementation and the experiment setup. Next, we present the experiment results obtained over this testbed and discuss their implications.

7.1 Testbed Design and Implementation

7.1.1 Platform

We used the following platforms for our APs and clients. APs were built out of an embedded PC104 board, called *Stargate*, which houses a compact flash slot for the wireless card. We used a Netgear 802.11b wireless card which uses the Prism 2.5 chipset and provides a channel switch latency of 20 ms (see Section 5). The clients were IBM Thinkpad Laptops supported with PCMCIA ZoomAir cards which also use the Prism 2.5 chipset (20 ms latency). Both APs and clients were running GNU/Linux (2.6 kernel) and used the *hostap* driver which was suitably instrumented to perform channel switching at the driver level.

7.1.2 Implementing MAXchop

A practical implementation of MAXchop would trigger channel switching using dedicated beacon messages. This design choice was made since such messages are used for client-AP synchronization and are thus received by all clients (even in the power-save mode) using the lowest supported bit-rate. This also minimizes packet losses which are possible while switching channels. Since the firmware for wireless cards is proprietary, we trigger channel switching using a UDP broadcast message sent by the AP which implements our MAXchop algorithm as a user-level daemon process. A corresponding daemon process executing at the client triggers this channel switch upon receipt of this message. The AP and client daemons also implement client-driven estimation of interference by requesting periodic scans of all channels. We note that our user-level implementation of channel hopping with instrumented drivers will suffer from additional overheads such as context switches, interrupt delays and data packet losses (as opposed to management frames) when compared to implementing this at the firmware level. Thus, we believe that the results obtained here will act as a *lower bound*, that is, network throughputs in a firmware-level implementation would be better as they would suffer less packet-loss and overhead.

7.1.3 Topology and Experiment Design

We designed a testbed network topology consisting of five APs. This is the most common number of APs found interfering with each other in average urban environments [1]. These APs were placed in an in-building environment in the lobby areas which closely mimic shops and cafe positions in typical shopping malls. Figure 6 shows the design of our hotspot testbed. The black rectangle represents the APs. There was one client associated to each AP. The client-AP distance was chosen to mimic the dimensions of a typical coffee shop. With such positioning, we found that all APs and clients interfere with each other much like in a typical dense hotspot area.

The design choice of using one client per AP was made in order to study the effect of interference *among* APs and to prevent diluting the results with intra-AP contention due to multiple clients. Network behavior due to intra-AP contention has been studied previously [5] and is not the focus of this paper. Also in our full evaluation we have studied the combined effect of inter-AP and intra-AP interference using multiple clients per-AP. The results closely resemble the ones presented here, and we do not discuss them due to space limitations.

We performed experiments using four methods of channel assignment: First was LCCS which used the three non-overlapping channels, represented as *NOV-LCCS*. This is the most common method used by APs in hotspots. Second was MAXchop using the same three non-overlapping channels, represented as *NOV-MAXchop*. Third, was MAXchop using the 11 partially overlapped channels based on the concept of I-factor [8], represented as *POV-MAXchop*. Finally, we manually built the best possible static assignment of channels to the five APs using partially overlapped channels and explicitly measuring interference at the APs. We note that this assignment can be computed in an algorithmic fashion, however, this would require considerable interaction among the APs [8]. We use this as a benchmark to evaluate whether channel hopping is capable of taking full advantage of partially overlapped channels in a fully distributed fashion. We represent this as *POV-static*. For each of these algorithms, we measured TCP and UDP throughputs at the application layer. FTP file transfers were used to trigger TCP flows, while high but constant bit-rate transfers were used as the UDP application.

7.2 Results

Figure 7 shows the results of our experiments. Shown on the Y axis are the individual throughputs obtained at each hotspot AP for TCP (Figure 7(a)) and UDP flows (Figure 7(b)) using each of the four algorithms. From a careful study of these plots, the following

salient points are evident:

7.2.1 Fairness Gains

Our results show that channel hopping improves fairness in throughput among hotspots. Consider the UDP throughputs obtained using three non-overlapping channels for *NOV-LCCS* versus *NOV-MAXchop* shown in Figure 7(a). With a static assignment performed using LCCS, AP5 gets a good channel, while APs 1,2 and 3,4 share a single channel each. Thus, AP5 gets a good throughput at the expense of the other APs. This unfairness would grow as the number of interfering APs increased. However, with channel hopping all five APs get roughly the same throughput. One can also observe that with partially overlapped channels, channel hopping improves fairness over the best static assignment using such channels, represented by *POV-static*. This is clear from both TCP throughputs in Figure 7(a) and UDP throughputs in Figure 7(b).

7.2.2 Throughput Gains

Our proposed channel hopping algorithm is capable of providing throughput gains by using partially overlapped channels. For example for UDP throughputs, *POV-MAXchop* improves total system throughput by 10% (Figure 7(a)). This matches the throughput gains of 8.9 % obtained by *POV-static* which was manually computed to best utilize partially overlapped channels. Similar gains were achieved for the TCP throughputs: 15.13 % by *POV-MAXchop* over *NOV-chop* and 15.05 % by *POV-static* over *NOV-LCCS*. Note that our channel hopping algorithm achieves these gains in a fully distributed fashion without the need for any interaction between APs.

7.2.3 Impact of channel switching

All experiments that used channel hopping incurred the costs associated with implementing the channel switch operation. Additionally, the TCP results also incorporate effects on TCP's congestion window due to occasional packet losses while switching channels. Our results show that the effect of this overhead on UDP/TCP throughputs is negligible (less than 0.6 %). A detailed discussion of this tradeoff is presented in Section 5. From the results shown in Figure 7, with non-overlapping channels, the channel hopping algorithm namely, *NOV-MAXchop* incurred a 0.57% reduction in total system throughput from 17.5 to 17.4 Mbps. With TCP throughputs, the *NOV-MAXchop* incurred a 0.54 % reduction in total system throughput. Interestingly, with partially overlapped channels, *POV-MAXchop* provided an improvement of 2.6% over using static partially overlapped channels. For TCP throughputs, *POV-MAXchop* incurred a reduction of 0.46 % in total system throughput. We note that these numbers show an over-estimate of the actual costs that would be incurred since we implement channel hopping at the driver/user-level within the host operating system as opposed to implementing them in the wireless card's firmware.

8. RELATED WORK

Prior work by Akella et. al. in [1], the *Echos* system [5] and the SSCH protocol by Bahl et. al. [6] have been discussed before in Section 1. Hence we do not discuss them further in this section.

There has been much recent work in improving client throughputs in the context of centrally managed wireless LANs. Work by Bejerano et. al [3] presented a centralized linear program (LP) based formulation of balancing client load among APs using association control. Work by Balachandran et. al. [2] discuss a network-driven method for providing hints to users. These hints are used to improve the quality of service experienced by a user by, for example, requesting them to roam to a better provisioned AP.

The CFAssign methods in [4] address the joint problem of channel assignment and load balancing in centrally managed WLANs. The paper shows that client-driven mechanisms are important in capturing interference accurately. By comparing against prior vertex coloring based approaches [14] the paper also shows that vertex coloring approaches tend to be inefficient for centrally managed WLANs. The scope and nature of their problem is much different than that of uncoordinated wireless environments. We discussed this in detail in Section 3.

Apart from wireless LANs, there has been much work on channel assignment algorithms in the context of cellular networks [15, 16] and wireless mesh networks. The interested reader is referred to [17, 18, 19, 20] and the references therein.

9. CONCLUSION

In this paper, we have addressed the problem of distributed channel assignment in uncoordinated wireless environments. where fairness assumes significance over throughput maximization. We propose channel hopping as a simple and efficient method of distributed channel assignment that provides good fairness properties and is also able to take full advantage of partially overlapped channels for the much needed throughput gains in such dense networks. We have evaluated our approach extensively through simulations over real hotspot topologies and a full-fledged testbed based implementation.

The key contribution of this paper has been to bring into focus the fairness properties of our channel hopping technique as a simple, efficient and distributed method of dividing a system's resources among participating users. We hope that the research insights gained from this work would lead to a practical deployment of these methods in hotspots and further research into applying them to other domains where a fair division of a system's resources is an important consideration.

10. REFERENCES

[1] A. Akella, G. Judd, S. Seshan, and P. Steenkiste, "Self-management in chaotic wireless deployments," in *ACM Mobicom*, 2005.

[2] A. Balachandran, G. Voelker, and P. Bahl, "Wireless hotspots: Current challenges and future directions," in *ACM WMASH*, 2003.

[3] Y. Bejerano, S. Han, and L. Li, "Fairness and load balancing in wireless lans using association control," in *ACM Mobicom*, 2004.

[4] A. Mishra, V. Brik, S. Banerjee, A. Srinivasan, and W. Arbaugh, "A client-driven approach for channel management in wireless lans," in *IEEE Infocom*, 2006.

[5] A. Vasan, R. Ramjee, and T. Woo, "Echos: Enhanced capacity 802.11 hotspots," in *IEEE Infocom*, 2005.

[6] P. Bahl, R. Chandra, and J. Dunagan, "Ssch: Slotted seeded channel hopping for capacity improvement in ieee 802.11 ad hoc wireless networks," in *ACM Mobicom*, 2004.

[7] "Intel pro/wireless network connection for mobile," http://www.intel.com/network/connectivity/products.

[8] A. Mishra, V. Shrivastava, S. Banerjee, and W. Arbaugh, "Partially overlapped channels not considered harmful," in *ACM Sigmetrics*, 2006.

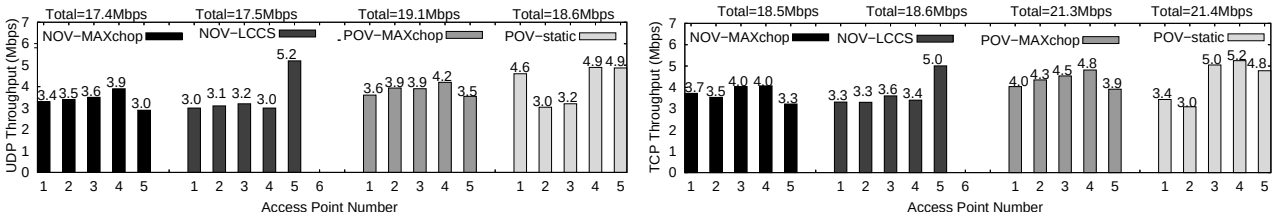
[9] J. Geier, "Assigning 802.11b access point channels," *Wi-Fi Planet*, 2004.

[10] C. McDiarmid and B. Reed, "Channel assignment and weighted coloring," *Wiley Networks*, 2000.

[11] G. B.Sabbatel, A. Duda, M. Heusse, and F. Rousseau, "Short-term fairness of 802.11 networks with several hosts," in *IEEE MWCN*, 2004.

[12] A. Blum, "New approximation algorithms for graph coloring," *JACM*, 1994.

[13] "Wigle: Wireless geographic logging engine," http://www.wigle.net/.



(a) UDP Throughputs. (b) TCP Throughputs.

Figure 7: Experiment Results.

[14] A. Mishra, S. Banerjee, and W. Arbaugh, "Weighted coloring based channel assignment for wlans," *Mobile Computer Communications Review (MC2R)*, vol. 9, no. 3, 2005.

[15] T. Rappaport, *Wireless Communications: Principle and Practice*, Prentice Hall, 1996.

[16] B. Krishnamachari, S. Wicker, R. Bejar, and C. Fernandez, "On the complexity of distributed self-configuration in wireless networks," *Telecommunication Systems*, 2003.

[17] M. Alicherry, R. Bhatia, and L. Li, "Joint channel assignment and routing for throughput optimization in multi-radio wireless mesh networks," in *ACM MobiCom*, 2005.

[18] M. Kodialam and T. Nandagopal, "Characterizing the capacity region in multi-radio, multi-channel wireless mesh networks," in *ACM MobiCom*, 2005.

[19] Ashish Raniwala, Kartik Gopalan, and Tzi cker Chiueh, "Centralized channel assignment and routing algorithms for multi-channel wireless mesh networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, 2004.

[20] J. So and N.H. Vaidya, "Routing and channel assignment in multi-channel multi-hop wireless networks with single network interface," in *QShine*, 2005.

Appendix

In this appendix, we sketch a proof of convergence of the MAXchop algorithm discussed in Section 4.

Proof: Consider the Hop routine shown in Algorithm MAXchop. Assume that the current AP x was using color $\lambda_i(x)$ in slot i prior to executing Hop. The amount of interference as captured by the η function for slot i was $\eta(\lambda_i(x))$. After executing Steps 2-5 of the function Hop, the new interference value equals η_{min} Let $\tilde{\lambda}_i(x)$ denote the new color obtained in Step 7.

Thus, since $\tilde{\lambda}_i(x) \in C$: $\eta(\tilde{\lambda}_i(x)) \leq \eta(\lambda_i(x))$**—(1)**.

Note that the above inequality holds even if Step 7 were to choose *any* random color out of the set C. Consider $L_{sum} = \sum_{\forall(u,v)\in E} L(u,v)$, the sum of all *L-values*. Now,

$$L_{sum} = \sum_{\forall(u,v)\in E} L(u,v) = \frac{1}{2}\sum_{v\in V}\sum_{u\in N(v)} L(u,v)$$
$$= \frac{1}{2}\sum_{v\in V}\sum_{u\in N(v)}\sum_{i=1\ldots N_s} \rho(\lambda_i(u),\lambda_i(v))$$

Rearranging the innermost two summations,

$$L_{sum} = \frac{1}{2}\sum_{v\in V}\sum_{i=1\ldots N_s}\sum_{u\in N(v)} \rho(\lambda_i(u),\lambda_i(v))$$
$$= \frac{1}{2}\sum_{v\in V}\sum_{i=1\ldots N_s} \eta^v(\lambda_i(v))$$

Let $\tilde{L}_{sum}$ denote the value of L_{sum} after node x executed the Hop function. Based on the above steps,

$$\tilde{L}_{sum} = \frac{1}{2}\sum_{v\in V}\sum_{i=1\ldots N_s} \eta^v(\tilde{\lambda}_i(v))$$

Now,

$$\tilde{L}_{sum} - L_{sum} = \frac{1}{2}\sum_{i=1\ldots N_s} [\eta^v(\tilde{\lambda}_i(v)) - \eta^v(\lambda_i(v))]$$

Based on Equation 1,

$$\tilde{L}_{sum} - L_{sum} \leq 0 \quad \text{as} \quad \eta^v(\tilde{\lambda}_i(v)) \leq \eta^v(\lambda_i(v))$$

Thus, its clear that $\tilde{L}_{sum} \leq L_{sum}$. Since L-value is the number of interfering edges in the network as a whole. This is *integral* and has a minimum. Also $L_{sum} \leq |E|$. This proves convergence in $O(\delta)$ rounds, where $\delta = |E|/|V|$.

Below, we present a reference algorithm for the ComputeMinMax routine discussed in Section 4.

Procedure ComputeMinMax(C, i)

1: **if** i = 1 **then**
2: return a color from C uniformly at random
3: **end if**
4: Compute: $\forall u \in N(x), L(u) = \sum_{j=1\ldots i-1} \rho(\lambda_j(u),\lambda_j(x))$
5: Compute: $\forall c \in C, L_c(u) = L(u) + \rho(c,\lambda_i(u))$
6: Define: $\forall c \in C, \hat{L}_c = \langle L_c(u)\rangle$ over all $u \in N(x)$ sorted in non-increasing order of $L_c(u)$ value
7: Let $\hat{L}_{min} = \hat{L}_\alpha, C_{min} = \alpha$ for a random $\alpha \in C$
8: **for** $c \in C$ **do**
9: **if** $\hat{L}_{min} >_{lex} \hat{L}_c$ **then**
10: $\hat{L}_{min} = \hat{L}_c$ and $C_{min} = c$
11: **end if**
12: **end for**
13: return C_{min}

We briefly explain the working of the ComputeMinMax routine described above. For the first slot, the routine returns a color randomly from the set C (Step 1-3). Otherwise, for the current AP x, Step 4 computes the amount of interference $L(u)$ with each AP u that interferes with x (i.e., $u \in N(x)$). Step 5, calculates for each color $c \in C$, the interference array $L_c(u)$ for every $u \in N(x)$ assuming the AP x were to choose color c. Out of all such colors, the routine selects the color c which provides the best min-max values for the interference array $L_c(u)$ (Steps 8-12). This is done by sorting these arrays in non-increasing order and finding the one that has the least lexicographic value. The corresponding color/channel is then returned.