A Self-Managed Distributed Channel Selection Algorithm for WLANs

D.J. Leith, P. Clifford
Hamilton Institute, National University of Ireland, Maynooth, Ireland

Abstract— **In this paper we consider the problem of a wireless LAN selecting a channel to minimise interference with other WLANs. We focus on interfering infrastructure-mode networks, where each access point (AP) or base station has a wired backhaul link. We introduce a new fully distributed and self-managed channel selection algorithm that does not require direct communication between APs nor explicit estimation of the network interference graph. The sole information required by the algorithm is feedback to each WLAN on the presence of interference on a chosen channel; such feedback is already commonly provided by WLAN protocols such as 802.11. We establish that convergence of the distributed algorithm is guaranteed provided that the channel selection problem is feasible. Extensive simulation results are presented that demonstrate rapid convergence under a wide range of network conditions and topologies. While the scope of the present paper is confined to infrastructure networks with static topology, the utility of the proposed algorithm in situations where the network topology is time-varying is briefly discussed.**

I. Introduction

Increasingly, the trend is towards denser wireless LAN deployments. One factor driving this trend is simply the increasingly widespread uptake of WLAN technology, e.g. see [1]. However, another is the strong correlation between link rate and distance from an access point (AP)[1] With regard to the latter, because the wireless medium is a broadcast one any station operating at a low rate not only impacts that station itself but also reduces the capacity of the entire wireless cell (low rate transmissions occupy more time on the air than high rate transmissions, thus reducing the available transmission opportunities). Denser deployments of APs therefore offer the potential to significantly increase network capacity. However, a key challenge in realising this potential is the effective management of the wireless spectrum to mitigate interference between nearby APs, which tends to increase as deployments become more dense.

We focus on interfering infrastructure-mode networks, where each AP has a wired backhaul link. One approach is to explicitly measure which APs (and associated wireless stations) interfere with each other and to use this information to directly optimise the allocation of wireless channels to APs, e.g. see [2], [3]. This yields a centralised channel allocation scheme. Centralised schemes are, however, simply not feasible in many practical situations where interfering APs do not belong to the same administrative domain; for example in residential and commercial buildings where interfering wireless networks may be operated by different households or businesses. Moreover, even when APs belong to a single administrative domain it is often far from straightforward to determine which APs are interfering with each other as (i) the interference distance is generally larger than the distance at which nodes can communicate (and so an AP cannot rely upon reading packet headers to identify the source of interference) and (ii) interference between APs is often time-varying in nature due to changing environmental conditions.

It is therefore attractive to consider self-managed distributed channel selection schemes that do not depend upon communication of control information. One such self-managed approach is for APs to simply select a channel randomly and rely upon over-provisioning of channels to make it unlikely that nearby APs choose the same channel. However, it is clear that such a scheme is only really viable in situations with few interfering WLANs and many spare channels. A modified version of the 802.11 CSMA/CA scheme might also be employed (e.g. see [4]), but this seems ill-suited to channel allocation as it inevitably involves persistent "collisions" and persistent changes in channel (or transmission slot), even when the network topology is static.

In this paper we propose a new fully distributed algorithm suited to dynamic channel selection by WLANs. In this scheme each AP employs a simple learning rule to adaptively select the channel to transmit on. The algorithm does not require direct communication between APs, hence it is referred to as *self-managed*. The sole information required by the algorithm is feedback to each AP on the presence of interference on a given channel; such feedback is already commonly provided by WLAN protocols such as 802.11. We show that the algorithm is guaranteed to converge to an optimal solution that minimises interference between WLANs provided this is feasible. Moreover, we demonstrate the convergence is, on average, remarkably fast under a wide range of network conditions and topologies.

As an illustration of the potential capacity gain using the proposed algorithm, we briefly consider a simple 802.11 WLAN example where APs are randomly located in a unit square and the WLANs associated with two APs interfere when the APs are located within a radius R of each other. For simplicity, we assume that each WLAN is saturated i.e.

[1]In this paper we use the term AP to denote the co-ordinating station in an infrastructure WLAN that is responsible for channel selection. There is no intention to restrict consideration to a specific WLAN technology and the AP here might equally be the access point in an 802.11 WLAN or the base station in an 802.16 WLAN, etc. Each AP has an associated collection of wireless stations, for which it provides backhaul access, and we refer to the collection of stations plus AP as a WLAN.

0-7803-9550-6/06/$20.00 © 2006 IEEE

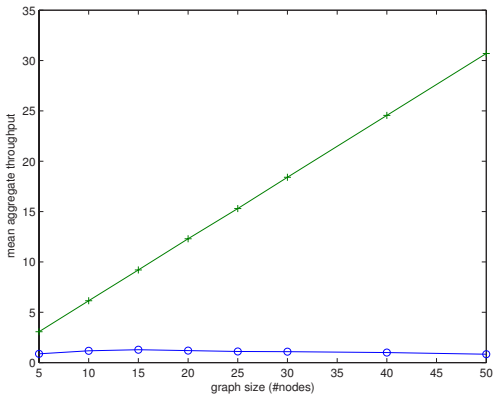

Fig. 1. Mean aggregate network throughput (normalised by channel bandwidth) vs network size with (i) all network nodes using same channel and (ii) with optimal channel allocation (802.11b MAC with 11Mbs PHY and 1500 byte packets, random disk interference graphs with radius R=0.5, each point plotted is the mean throughput taken over 100 random disk graphs, for simplicity we assume that nodes always have a packet to send i.e. are saturated)

always has a packet to send. Straightforward modification of the Bianchi [5] throughput model to include the effect of the interference pattern between WLANs (the standard Bianchi model assumes that every node interferes with every other node) yields the throughput predictions shown in Figure 1. The throughput values shown are the aggregate of all WLANs, averaged over 100 example networks, and thus provide an indication of the mean network capacity. Also shown in Figure 1 are the corresponding mean throughput predictions when channels are optimally allocated to minimise interference between WLANs[2]. Evidently, the potential capacity gain is considerable.

II. Related Work

Tassiulas and Ephremides [2] present a centralised algorithm to maximise throughput in multi-hop wireless networks which uses global knowledge of the queue lengths to decide routing and scheduling/channel selection. Raniwala et al [6], [7] also consider channel allocation schemes for multi-radio multi-hop networks. Luo et al [4], [8] study a fair queueing model for multi-hop WLANs which operates on the packet level. They give a distributed implementation which seeks to approximate their ideal centralised algorithm for allocating transmission opportunities although this algorithm relies upon explicit communication of state information between stations via packet headers. All of these multi-hop scheduling problems require that stations at both ends of a hop share a common channel and thus fundamentally differ from the channel allocation problem considered in the present paper.

[2]We assume here that sufficient channels are available to make the optimum allocation feasible. While 802.11b/g is limited to 3 orthogonal channels, other technologies (e.g. 802.11a) support a much greater number. This issue is discussed further in Section V.

Bejerano and Bhatia [3] consider a similar problem to the present paper and give a framework for providing fairness, QoS and high throughput in infrastructure-mode single-hop 802.11 networks where the access points have a wired backhaul connection. Their fair slot assignment approach builds on an approximation algorithm for the disk graph colouring problem but is a centralised scheme. Leung and Kim [9] consider a similar problem and propose a heuristic centralised algorithm. Akella et al [1] study power control and rate adaptation in the same setting, but do not consider algorithms for channel allocation. The closest work to that in the present paper is perhaps that of Kaufmann et al [10] who study a distributed simulated annealing algorithm for channel selection of WLANs. Kaufmann et al model the interference on a certain channel as a sum of received power from other APs on the same channel and the background noise on that channel. Their algorithm implicitly relies on a time parameter to control an AP's channel selection probabilities.

An extensive literature exists relating to channel assignment in cellular phone networks and the reader is referred to the survey paper by Narayanan [11] for details. Since the channel assignment problem is NP-hard, numerous heuristic schemes have been proposed that come with few, if any, guarantees on performance. Notable exceptions include Sparl and Zerovnik [12], Sudeep and Vishwanathan [13], Janssen et al [14] and Narayanan and Shende [15] who study distributed algorithms for frequency assignment in cellular networks and provide competitive performance bounds. However, this work concentrates on the cellular network case where the graph is always a subgraph of the triangular lattice. Moreover, the distributed algorithms considered require explicit communication of channel information between nodes within a k hop neighbourhood, whereas in the present paper we consider schemes that do not require such direct communication.

The channel allocation problem considered here is equivalent to graph colouring. The literature on graph colouring is extensive but has mostly focussed on centralised algorithms or on localised algorithms that use extensive message passing such as Survey Propagation[16]. Finocchi et al [17] and Kubale and Kuszner [18] analyse simple distributed colouring algorithms. Their greedy algorithms simply use extra colours if difficulties occur and so in general may achieve a colouring that uses far from the minimum number of colours.

III. Distributed Algorithm

Let c denote the number of available channels and let each AP maintain a c element state vector p. Let p_i denote the ith element of p with $\sum_i^c p_i = 1$. We consider the following distributed algorithm for updating p.

Algorithm: Distributed Channel Selection

1) Initialise $p = [1/c, 1/c, \ldots, 1/c]^T$
2) Toss a weighted coin to select a channel, with p_i the probability of selecting channel i. Measure the quality of the channel: any interference measure can be used that yields a "success" when interference/channel noise

is within acceptable levels and "failure" otherwise. Thus, we might , for example, use an aggregate measure derived from multiple packet transmissions, direct measurement of the channel SNR or some other approach.

3) On success on channel i, update p as

$$p_i = 1 \quad (1)$$
$$p_j = 0 \forall j \neq i \quad (2)$$

i.e. on success we stay with that channel. This creates a degree of "stickiness" which ensures that any channel allocation that removes interference between WLANs is an absorbing state.

4) On failure on channel i, update p as

$$p_i = (1-b)p_i \quad (3)$$
$$p_j = (1-b)p_j + b/(c-1) \forall j \neq i \quad (4)$$

i.e. on a failed transmission multiplicatively decrease the probability of using that channel, redistributing the probability evenly across the other channels. b is a design parameter, $0 < b < 1$; the selection of the value of b is considered in detail in the next section.

5) Return to 2.

The strategy adopted by this algorithm is a fairly simple one. The station periodically measures the channel quality (the interval between measurements need not be constant and can be selected to respect the cost of switching channel and the time needed to measure channel quality). When the channel quality of acceptable, keep using the same channel. Otherwise, randomly choose a channel with weighted probability based on past experience. Note that on a failure there exists a non-zero probability associated with every channel, albeit with less weight on channels that are prone to failure.

We have the following main result on the convergence of this algorithm. Let G denote the interference graph associated with a wireless network; that is, there exists a vertex in G corresponding to each WLAN and edges exist between each pair of interfering WLANs. A proper channel allocation is one where each WLAN uses a different channel from all of its neighbours in G i.e. an allocation where WLANs do not interfere.

Theorem 1 *Suppose each vertex in a graph G operates the Distributed Channel Selection algorithm. A vertex makes a successful transmission when it chooses a different channel from all of its neighbours, otherwise the transmission may fail. Assume that the number of available channels c is greater than or equal to the chromatic number χ of G (i.e. a proper channel allocation is indeed feasible). Then the Distributed Channel Selection algorithm converges, with probability one, to a proper channel allocation.*

Proof: We will show that in a determined finite amount of steps the system has some minimum positive probability of convergence. We show that starting from any configuration the system can reach some standard state after two steps, and can then potentially reach a state where every vertex experiences a failure simultaneously, allowing convergence without issues of independence. Hence the network always has positive probability of global success and so will almost surely converge.

First we show that if the system has reached a configuration with some colour selection probabilities very small there is a positive probability that it will return in two steps to a standard state with all such probabilites on just collided nodes lower bounded. Thus (with some probability) the initial colour selection probabilities will have no effect on the probability of a given evolution happening. After any step T_0 there was either global success (and convergence) or at least two vertices failed by interfering with each other, referred to in the sequel as a "collision". Starting at time T_0 we allow the system to evolve for 2 more steps and lower bound the probability of the system having no vertex with two consecutive same colour collisions. Consider any vertex; after the first collision by choosing colour i_1, it has probability $pr_1 > \frac{b}{c-1}$ of choosing some specific other colour i_2 and probability $pr_2 > b$ of choosing any colour other than i_1. So the probability of two repeated collisions at a specific vertex is $pr_3 < 1-b$. In the whole system the probability of some vertex having two consecutive same colour collisions is $pr_4 < n(1-b) - \binom{n}{2}(1-b)^2 + \cdots < 1$. Hence with some probability $pr_5 > 1 - pr_4 > 0$ the system has no vertex with consecutive same colour collisions. Thus after these two steps with probability pr_5 all colour selection probabilities of vertices which have just collided are strictly greater than $\frac{b}{c-1}$.

We now describe a specific evolution of the system which concludes with all vertices failing simultaneously, providing some probability of global success after the next step. Using the fact that all colour selection probabilites of colliding vertices are lower bounded, after any collision any colour choice is possible. Hence the probability pr_6 of our specific evolution (while very small) is strictly positive and lower bounded. If we have not converged after the above 2 steps, then two vertices k_1 and k_2, say, have just experienced a collision. By way of notational convenience we say these two vertices were *visited* at step 2. Suppose now that k_1 collides with its first non visited neighbour k_3 (if any) at step 3. Suppose also that k_2 collides with its first non visited neighbour (if any, potentially k_3 also) at step 3 also. We say that such vertices are *visited* at step 3. Inductively suppose now that a vertex once visited collides with all its nonvisited neighbours in consecutive steps. This is possible because a visited vertex having just collided can potentially choose any colour. Note that a vertex being visited simultaneously (along two different equal length paths from k_1 and k_2 say) is also possible.

Suppose that once a vertex has collided with all its non-visited neighbours it then repeatedly chooses colour 1 until step $T_1 = T_0 + 2 + md \times D$. We note that as a vertex k_4 is colliding with its nonvisited neighbours some of them may become visited from other vertices before they collide with k_4; we suppose then that k_4 does not visit such vertices.

We assume without loss of generality that the graph is connected. Thus at time $T_1 - 1$ it is possible for every vertex

to have been visited and to be choosing colour 1. Hence every vertex is colliding and any possible colour configuration is possible. So we can finally suppose that at step T_1 that every vertex selects a colour so that no collisions occur.

Using finiteness and assuming that all colour selection probabilities of failing vertices are lower bounded after the first two steps, there is some probability $pr_7 > 0$ such that $pr_6 > pr_7$ irrespective of the initial colour selection probabilities and which vertices collided initially. Defining $pr_8 = pr_7 pr_5$ to ensure no repeated collisions in the first two steps we can say that every $2 + md \times D$ steps the system will converge with probability at least pr_8. Hence after $j(2 + md \times D)$ steps we have converged with probability at least $1 - (1 - pr_8)^j$ which converges to 1 as $j \to \infty$. ■

Comments

1) *Clock Synchronisation*. Although, for simplicity, we assume slotted time in the foregoing analysis, Theorem 1 carries over without change to the situation where clocks are not synchronised. All that is required is that a proper channel allocation results in no transmission failures (we neglect the impact of non-interference related channel noise), while use of the same channel by neighbouring WLANs (i.e. a non-proper allocation) induces transmission failures with positive probability.
2) *Hidden nodes*. The presence of hidden nodes corresponds to directed links in the network interference graph. That is, hidden nodes induce transmission failures in neighbouring WLANs that share the same channel but do not themselves experience failures. Provided that the number of channels is sufficiently large (generally larger than the chromatic number of the associated undirected graph) so that all absorbing states of the distributed channel selection algorithm are feasible, the algorithm also converges to a proper channel allocation on directed graphs.
3) *CSMA/CA*. Although both are stochastic algorithms, the proposed distributed channel selection algorithm differs from CSMA/CA type algorithms in many fundamental respects. For example, for a given network of WLANs the channel selection algorithm converges to a static allocation with no packet collisions, whereas the CSMA/CA algorithm incurs a persistent packet collision overhead.
4) *Learning Automata*. The proposed channel selection algorithm is closely related to learning automata, see for example [19]. However, previous work has largely focussed on individual automata rather than the interconnection of a large number of automata, with few results known about the properties of interconnected learning automata. To our knowledge, Theorem 1 is one of the first convergence results for interconnected learning automata.

IV. Convergence Rate

Theorem 1 guarantees that the Distributed Channel Selection algorithm converges to a solution that minimises interference, provided we have enough channels for the allocation problem to be feasible, but says nothing about the *rate* of convergence i.e. the number of iterations of the algorithm needed before convergence to a proper channel selection. Since graph colouring is known to be an NP-hard problem in general, tight bounds on convergence rate are non-trivial to obtain. In this section we use simulation studies to explore the convergence rate of the algorithm under a range of network conditions and topologies.

A. Impact of Learning

We begin by investigating the impact on convergence rate of the learning elements of the Distributed Channel Selection algorithm, namely Steps 3 and 4. We can remove these steps to yield a crude algorithm which assigns a constant probability to each channel and thus evolves as a uniform random walk over every possible combination of channel allocations; while this is guaranteed to converge eventually since the random process is ergodic, it is clear that the convergence rate of this algorithm will generally be extremely slow. More interesting is a modification of this crude algorithm to add the "stickiness" step 3 whereby an AP settles on a successful channel, but which upon failure still assigns uniform probability to every channel (i.e. in the Distributed Channel Selection algorithm step 4 is replaced by "On failure update p to $[1/c, 1/c, ..., 1/c]^T$"). Figure 2 plots the mean number of iterations to converge versus the number of wireless nodes for this strategy and for the full Distributed Channel Selection algorithm[3]. In this example the network interference graph is modeled as a random disk graph; that is, APs are uniformly randomly located in a unit square and the WLANs associated with two APs interfere when the APs are located within a radius R of each other. A "failure" or "collision" occurs when neighbouring nodes select the same channel at a given iteration of the channel allocation algorithm, and a "success" when a node selects a different channel from all of its neighbours. Each of the convergence time values plotted are the average over 1000 randomly chosen disk graphs. The impact of the learning step 4 is evident: e.g. for a 30 node graph the learning step yields an improvement of four orders of magnitude in mean convergence time.

We can gain further insight into convergence behaviour as follows. Let $F_i(k)$ denote the probability that AP i experiences a failure at iteration k, and let $F(k)$ denote the probability that any AP experiences a failure at iteration k. A proper channel allocation induces a failure probability F of zero. The Distributed Channel Selection algorithm is a stochastic learning algorithm and it is therefore of interest to consider the expected failure probability $E[F(k)]$, where the expectation is taken over an ensemble of runs of the algorithm each starting from the same initial probabilities specified in Step 1. Figure 3 plots $E[F(k)]$ versus iteration number k for a randomly

[3]Note that in Figure 2 for each graph the number of channels is set equal to the chromatic number χ (calculated using the DSATUR algorithm); that is, we use the minimum possible number of channels for a feasible solution. The impact on the convergence rate of using larger numbers of channel is discussed in detail later.

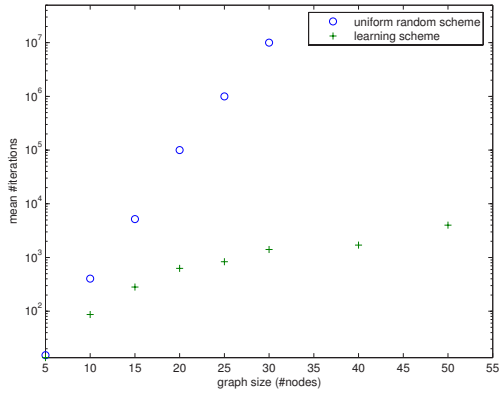

Fig. 2. Mean number of iterations to converge to a proper channel allocation vs number of nodes in interference graph for uniform random strategy and stochastic learning strategy (random disk graphs with radius R=0.5, mean is taken over 1000 graphs, channel number $c = \chi$, learning scheme parameters: $b = 0.1$)

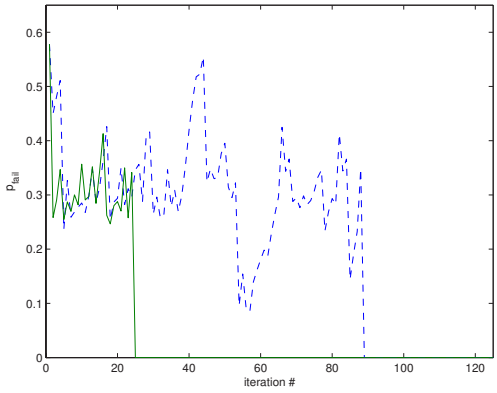

Fig. 4. Example realisations of failure probability time history (randomly selected 10 node graph with chromatic number χ of 6, channel number $c = \chi$, $b = 0.1$)

selected 10 node disk graph. The expectation is taken over a 1000 run ensemble. The data on a log scale and it can be seen that the rate of convergence appears to be exponential.

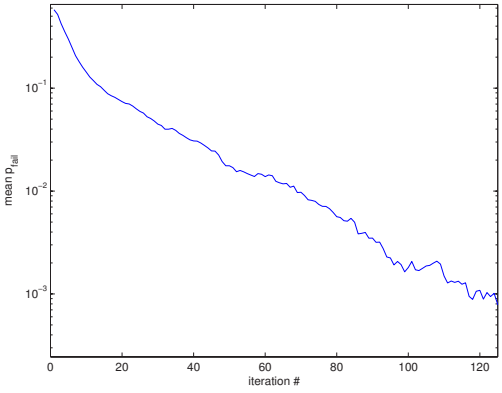

Fig. 3. Mean failure probability vs number of iterations (randomly selected 10 node disk graph with chromatic number χ of 6, channel number $c = \chi$, mean is over 1000 runs, $b = 0.1$)

This is initially somewhat surprising as the channel allocation problem is equivalent to graph colouring, which is known to be NP-hard, and thus exponentially fast convergence is unexpected. To understand this behaviour, we note that the plot is of the *expected* failure probability. Figure 4 shows two example realisations of the failure probability F (rather than $E[F]$) for the same graph as in Figure 3. It can be seen that the failure probability evolves in a complex manner before decaying rapidly to zero at some threshold time, τ, after which F remains identically equal to zero. The exponential decay of $E[F]$ can therefore be interpreted as $prob[\tau > k]$ decaying exponentially with k. That is, although it is possible for some realisations of the stochastic learning process to be slow to converge, as might be expected for an NP-hard problem, convergence appears nevertheless to be rather rapid on average.

B. Choice of Learning Parameter b

We next consider the choice of the the parameter b in the distributed channel selection algorithm. This parameter determines how quickly an AP discounts previous successes on a channel (or failures on other channels) on experiencing transmission failures on that channel. When $b = 0$, we have that no action is taken on failures. That is, when $b = 0$ an AP simply settles forever on the first channel on which it experiences a successful transmission. It is easy to see that this greedy strategy will not, in general, converge to a proper channel allocation. We therefore require $b > 0$. For $b > 0$ we have that the algorithm reduces the probability of choosing a channel, uniformly increasing the probability of choosing the remaining channels. Hence, failures can lead to selection of a new channel, regardless of previous successes. As b is made larger, failures are penalised more greatly and the "inertia" or "stickiness" of the system decreases. Small inertia allows the system to escape from poor choices of channel allocation but if the inertia is too small then convergence is slowed. Figure 5 plots the mean number of iterations to converge to a proper channel allocation versus the choice of learning parameter b used. It can be seen that as b approaches 0 the convergence time rapidly increases, as expected (recall that we know the algorithm can fail to converge when $b = 0$). As b approaches 1 the convergence time also rises as a consequence of the small inertia in the system. We can see that values of b in the range 0.1-0.3 yield the fastest convergence times for a range of interference levels (results are shown for interference radius R=0.25, 0.5, 0.75), with the convergence rate largely insensitive to the value used within this range.

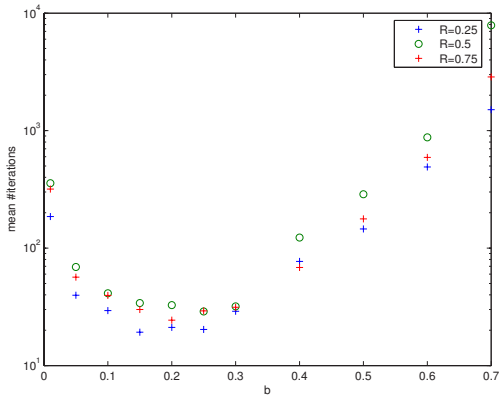

Fig. 5. Mean number of iterations to converge to a proper channel allocation vs learning parameter b (random disk graphs with radius R=0.5, channel number $c = 1.25\chi$, mean is taken over 1000 graphs)

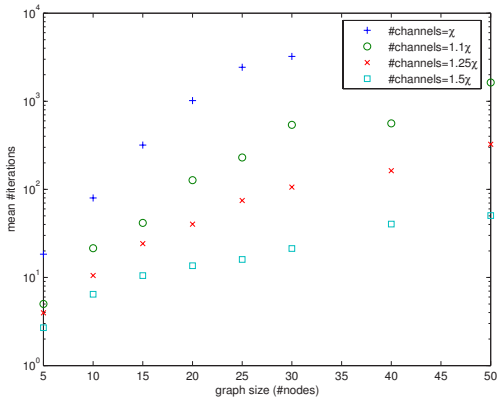

Fig. 6. Mean number of iterations to converge to a proper channel allocation vs number of nodes in interference graph and channel provisioning relative to chromatic number χ (random disk graphs with radius R=0.5, mean is taken over 1000 graphs, $b = 0.1$)

C. Impact of Channel Over-provisioning

In addition to considering situations where the number of channels is selected to be equal to the chromatic number χ of the network interference graph (i.e. we use the minimum possible number of channels for a feasible solution), it is also interesting to study the impact on convergence rate of over-provisioning the number of channels. Intuitively, as the number of available channels is increased we expect that the channel allocation problem becomes easier. Figure 6 plots the mean number of iterations for the Distributed Channel Selection algorithm to converge versus the channel provisioning (as a percentage of the chromatic number). As expected, we can see that the convergence time decreases as the level of over-provisioning is increased. However, what is perhaps more interesting is that it can also be seen that the impact of even a relatively small amount of over-provisioning can be very considerable. For example, 25% additional channels over and above the minimum required for a feasible solution yields more than an order of magnitude reduction in convergence time, while 50% yields nearly two orders of magnitude reduction. This level of reduction is largely insensitive to the interference graph parameters, see for example Figure 7.

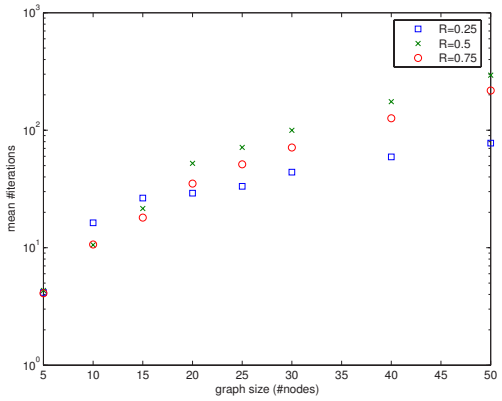

Fig. 7. Mean number of iterations to converge to a proper channel allocation vs number of nodes in interference graph and disk radius R of graph. (#channels is 1.25χ (25% over-provisioning)), random disk graphs, mean is taken over 1000 graphs, $b = 0.1$)

V. Performance Gain from Interference Management

In this section we briefly consider in more detail the potential gain in network throughput that can be achieved by proper channel allocation. A key issue in assessing the potential gain is the impact of interference on the MAC layer performance. We consider in particular two contrasting examples: (i) a naive centralised MAC scheduler that schedules a transmission in every available slot and (ii) an 802.11 CSMA/CA MAC scheduler. Figures 8 and 9 plot the mean per node throughput as a function of the offered load in a network of interfering WLANs. As before, we assume that the WLANs are uniformly randomly distributed in a unit square and that transmissions by WLANs within a distance R interfere (so that simultaneously transmissions on the same channel result in a failed transmission). For simplicity, we also assume that time is slotted and every WLAN always has a packet to send in each transmission slot. Packet arrivals are exponentially distributed and the impact of queueing dynamics is ignored (small queues are used).

It can be seen from Figure 8 that a naive centralised MAC scheduler yields eventually zero throughput as the network offered load is increased. This is because as the offered load at each WLAN increases, the local MAC scheduler eventually schedules a transmission in every available transmission slot. Hence, interfering WLANs sharing the same channel generate packet collisions in every slot and the achieved throughput is zero. Note that in practice link rate adaptation and a more sophisticated scheduler would yield better performance. Also shown in Figure 8 is the corresponding network throughput when the proposed distributed channel allocation is used.

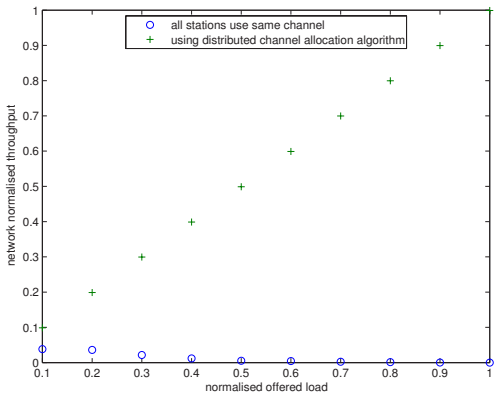

Fig. 8. Mean per node throughput (normalised by maximum physical throughput) versus normalised mean offered load for naive centralised MAC scheduler. Results are shown when (i) all nodes use the same channel for transmissions and when (ii) our distributed channel allocation algorithm is used. (20 node random disk graphs with radius R=0.5, channel number c=$/chi$, results are the average over 100 graphs, $b = 0.1$).

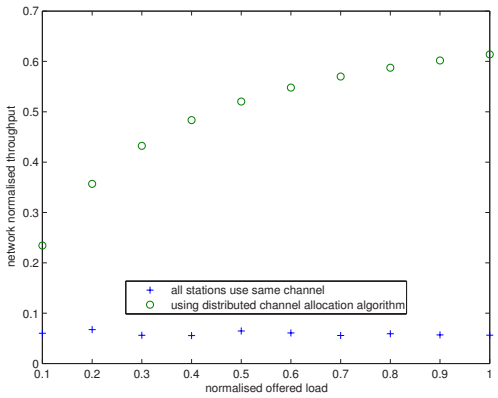

Fig. 9. Mean per node network throughput (normalised by maximum physical throughput) versus normalised offered load for 802.11b CSMA/CA MAC scheduler. Results are shown when all nodes use the same channel for transmissions and when our distributed channel allocation algorithm is used. (11Mbs PHY and 1500 byte packets, 20 node random disk graph with radius R=0.5, results are the average over 100 graphs, $b = 0.1$).

Figure 9 shows the corresponding results when an 802.11 CSMA/CA MAc scheduler is employed. These results are obtained using the finite load 802.11b model developed in [20], modified to include a general interference graph. The CSMA/CA scheme is elastic, increasing the average interval between transmission attempts as the channel becomes more heavily loaded and/or as the level of interference increases. As a result, the throughput does not fall to zero as the offered load increases, even when all WLANs share the same channel. It can nevertheless be seen that the throughput gain achieved by allocation of channels to minimise interference remains considerable.

These performance gains are, of course, contingent on the availability of sufficient channels to allow interference between WLANs to be minimised. Figure 10 illustrates the dependence of packet loss due to colliding transmissions on the number of available channels when the proposed distributed channel allocation algorithm is used. In this example, each interfering WLAN is saturated i.e. always has a packet to send, and a naive centralised MAC scheduler is used. For small numbers of channels, almost all packet transmissions collide and the achieved throughput is very low (in line with the previous discussion). As the number of channels is increased, it can be seen that the loss rate decreases, falling below 10%, on average, when 9 channels are available. Figure 11 shows the corresponding mean per node throughput values. As might be expected, when only a very small number of channels are available, no channel allocation is capable of yielding good performance and it is necessary to extend consideration to, for example, joint channel allocation and power control in order to mitgate interference. This is, however, beyond the scope of the present paper.

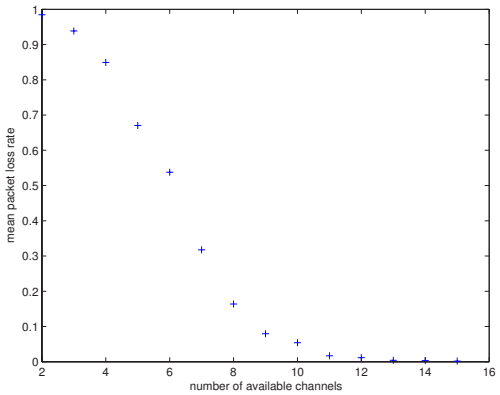

Fig. 10. Mean packet loss rate (ratio of losses to transmission attempts) versus number of available channels for naive centralised MAC scheduler and saturated WLANs. (20 node random disk graphs with radius R=0.5, results are the average over 100 graphs, $b = 0.1$).

VI. Non-stationary Topologies

Although the focus of the present paper is primarily on collections of infrastructure mode WLANs where the interference graph is static, we briefly consider the impact of time-variations in the interference graph on the performance of our distributed channel selection scheme. Time-variations might arise from factors including changes in traffic load as certain WLANs become idle, from AP mobility and/or from changes in environmental conditions.

Suppose that the interfering WLANs have converged on an optimal channel allocation that minimises interference. Changes in the interference graph will then typically require adjustment of these channel allocations. During the adjustment period, the channel allocations are sub-optimal and AP transmissions may interfere, resulting in packet loss. We can therefore measure the cost of changes in the interference graph via the number of packet losses induced. Intuitively, we expect

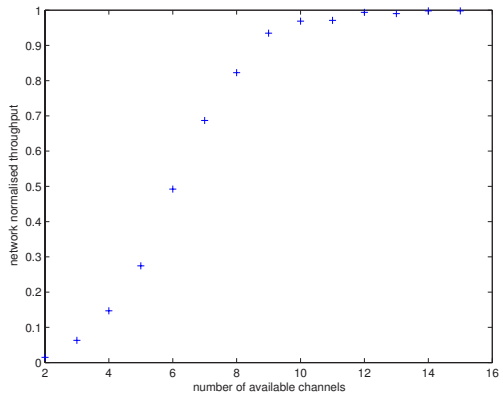

Fig. 11. Mean per node network throughput (normalised by maximum physical throughput) versus number of available channels for naive centralised MAC scheduler and saturated WLANs. (20 node random disk graphs with radius R=0.5, results are the average over 100 graphs, $b = 0.1$).

that provided changes in the interference graph occur slowly, compared with the convergence time of the channel selection scheme, the level of packet losses will be small.

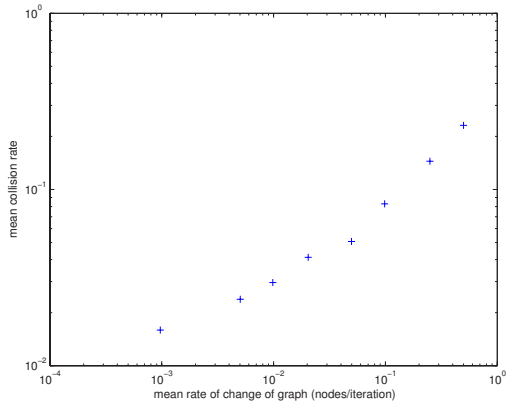

Fig. 12. Packet loss overhead (ratio of losses to transmission attempts) induced by topology changes vs rate of interference graph change. (Mean number of nodes 20, nodes are added/deleted at exponentially distributed intervals with mean rate given by the x-axis to random disk graph with radius R=0.25, a maximum of 5 channels are available, results are the average over 100 graphs, $b = 0.1$).

Figure 12 presents simulation results showing the mean packet loss (as a proportion of total packet transmissions) as a function of the rate of change of the interference graph. The results are the average over 100 random disk graphs each with a mean of 20 nodes. Nodes are randomly added/deleted at time intervals which are exponentially distributed, with the rate of change of the interference graph given by the reciprocal of the mean number of iterations between node addition/deletion. A maximum of only 5 channels are assumed available (so that at some instants the channel allocation problem may in fact not be feasible). It can be seen that, as expected, the rate of packet loss increases with the rate of change of the network interference graph. Observe, however, that the absolute rate of packet loss remains low even in rapidly changing conditions; for example, the packet loss rate is only 10% of transmissions even when a node is added/deleted from the network on average every 5 iterations. Recall that an iteration of the channel selection algorithm corresponds to the duration of a single packet transmission and so might be on the order of 1ms or less. Although owing to space limitations we do not explore performance under changing conditions further in the present paper, we do comment that these results indicate the potential utility of the proposed distributed algorithm.

VII. Conclusions

In this paper we consider the problem of a wireless LAN selecting a channe l to minimise interference with other WLANs. We introduce a new fully distributed channel selection algorithm that does not require direct communication between APs; that is, the algorithm is *self-managing*. The sole information required by the algorithm is feedback to each AP on the presence of interference on a chosen channel; such feedback is already commonly provided by WLAN protocols such as 802.11. We establish that convergence of the distributed algorithm is guaranteed provided that the channel allocation problem is feasible. While we do not as yet have full analytic results relating to the *rate* of convergence of the distributed algorithm, extensive simulation results are presented that demonstrate rapid convergence under a wide range of network conditions and topologies. While the scope of the present paper is confined to infrastructure networks with static topology, the utility of the proposed algorithm in situations where the network topology is time-varying is briefly discussed.

VIII. Acknowledgements

This work was supported by Science Foundation Ireland grant IN3/03/I346. The authors would like to thank Ken Duffy, David Malone and Rick Middleton for many helpful discussions.

References

[1] A. Akella, G. Judd, P. Steenkiste, and S. Seshan. “Self management in chaotic wireless deployments”. In *MobiCom 05: Proceedings of the 11th annual international conference on Mobile computing and networking*, 2005

[2] L. Tassiulas, A. Ephremides, “Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks”, *IEEE Transactions on Automatic Control*, 37 (12), 1992.

[3] Y. Bejerano, S.-J. Han and L. (Erran) Li, “Fairness and load balancing in wireless LANs using association control”, *MobiCom ’04*, 2004.

[4] H. Luo, P. Medvedev, J. Cheng, S. Lu, “A self coordinating approach to distributed fair queuing in ad hoc wireless networks”, *Proc. of IEEE INFOCOM ’01*, 2001.

[5] G. Bianchi, “Performance analysis of IEEE 802.11 distributed coordination function”, *IEEE Journal on Selected Areas in Communications*, 18(3):535-547, March 2000.

[6] A. Raniwala, K. Gopalan, T. Chiueh, “Centralized Algorithms for Multi-channel Wireless Mesh Networks”. ACM SIGMOBILE Mobile Computing and Communications Review (MC2R), 2004.

[7] A. Raniwala, T. Chiueh. “Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network”. In *Proceedings of IEEE International Conference on Computer Communications*, 2005.

[8] H. Luo, S. Lu, "A topology independent fair queueing in ad hoc wireless networks", *Proc. of the 2000 International conference on network protocols*, 2000.
[9] B.J.Leung, K.K.Kim, "Frequency assignment for IEEE 802.11 wireless networks". *Proc. 58th IEEE Vehicular Technology Conference*, 2003.
[10] B. Kauffmann, F. Baccelli, A. Chaintreau, K. Papagiannaki, C. Diot, "Self Organization of Interfering 802.11 Wireless Access Networks,", *INRIA Technical Report*, August 2005.
[11] L. Narayanan, "Channel assignment and graph multicoloring," *Handbook of wireless networks and mobile computing*, Wiley series on parallel and distributed computing, 2002.
[12] P. Sparl, J. Zerovnik, "2-local 5/4-competitive algorithm for multicoloring triangle-free hexagonal graphs", *Information Processing Letters* 90, 2004.
[13] K. S. Sudeep, S. Vishwanathan, "A technique for multicoloring triangle-free hexagonal graphs",' *Discrete Mathematics* 2002.
[14] J. Janssen, D. Krizanc, L. Narayanan, S. M. Shende, "Distributed online frequency assignment in cellular networks," *Proc. of the 15th annual symposium on theoretical aspects of computer science*, Lecture Notes in Computer Science Vol. 1373, 1998.
[15] L. Narayanan, S. M. Shende, "Static frequency assignment in cellular networks", *Algorithmica* 29 (2001).
[16] A. Braunstein, M. Mezard, R. Zecchina, "Survey propagation: an algorithm for satisfiability", Random Structures and Algorithms, 2005
[17] I. Finocchi, A. Panconesi, R. Silvestri, "Experimental analysis of a simple, distributed vertex coloring algorithms", *Algorithmica* 41 (2005).
[18] M. Kubale, L. Kuszner, "A better practical algorithm for distributed graph coloring," *Proc. of IEEE PARELEC'02*, 2002.
[19] K. Narendra, M. A. L. Thathachar, "Learning Automata: an introduction", Prentice Hall, 1989.
[20] Duffy, K., Malone, D., Leith,D.J., Modelling the 802.11 DCF under heterogenous finite load. *Proc. Workshop of Resource Allocation in Wireless Networks, Trento, Italy*, 2005.