Distributed Channel Assignment in Multi-Radio 802.11 Mesh Networks

Columbia University Technical Report

Bong-Jun Ko*, Vishal Misra†, Jitendra Padhye‡ and Dan Rubenstein*
*Dept. of Electrical Engineering, Columbia University, {kobj,danr}@ee.columbia.edu
†Dept. of Computer Science, Columbia University, misra@cs.columbia.edu
‡Microsoft Research, padhye@microsoft.com

***Abstract*— 802.11 wireless mesh networks are an attractive alternative to wired local and metropolitan area networks. To increase the utilization of the 802.11 spectrum in these environments, recent work has explored how to utilize the entire 802.11 spectrum when transmitting data through the mesh network. This paper reports on our design of a distributed, self-stabilizing protocol that assigns channels to mesh nodes in large-scale mesh networks. We demonstrate the efficacy of our protocol on a real-world, 14-node testbed comprised of nodes, each equipped with an 802.11a card and an 802.11g card. We show via extensive measurments on our testbed that our channel assignment algorithm improves the network capacity by 50% in comparison to a homogeneous channel assignment and by 20% in comparison to a random assignment. Furthermore, our protocol provides particular benefit to flows over long paths which would otherwise suffer poor end-to-end performance due to wireless interference.**

I. Introduction

802.11 wireless mesh networks are now an attractive alternative to wired local and metropolitan area networks for connecting wireless clients in a local neighborhood to a wide-area backbone. However, there are limitations. First, mesh networks are a type of ad-hoc network, and it is well-known that an ad-hoc network's capacity drops quickly as the network size grows [8], [13]. Second, despite the existence of multiple channels that can be used within the 802.11 wireless frequency band, the current 802.11 interface does not permit simultaneous operation on multiple channels. Ad hoc networks are therefore often deployed using a single channel, leaving much of the available 802.11 spectrum unused.

Recent work has proposed developing ad-hoc networks that utilize the entire 802.11 spectrum. Some work, such as [3], [16], [26], [24], assumes that interfaces can support multiple channels or that the MAC layer protocols can be modified. Other work has focused on optimizing routing within an ad-hoc or mesh network where the channels are already assigned [10], [20], [9], [5]. Recent approaches have considered architectures where mesh nodes are equipped with multiple wireless interfaces, such that neighboring nodes sharing at least one channel can communicate directly [21], [22], [23], [1], [25].

In this paper, we also consider channel assignment problem for mesh networks consisting of multi-radio 802.11 nodes. The following assumptions distinguish our work from these prior publications:

- The number of nodes forming the network can be large. In particular, for residential mesh networks, the number of nodes could be within the hundreds or thousands. Nodes may also be independently managed.
- Nodes are generally static, and the physical topology is expected to rarely change.
- The traffic matrix is dynamic: the set of flows traversing the network changes rapidly with time, or is difficult to predict.

Given these above assumptions, our goal is to construct *a distributed channel assignment strategy* that utilizes the entire 802.11 spectrum and routes flows intelligently within the network to maximize the capacity of the network. The large size of the network compels our solution to be distributed in nature, and the dynamic nature of traffic suggests that the assignment should depend more on the physical structure of the network than on the current traffic dynamics.

Our channel assignment mechanism is inspired by our previous theoretical work [11] which presents the theoretical design of distributed, self-stabilizing protocol for resource replication problem in emerging networks. The self-stabilizing nature of the mechanism ensures a stable channel assignment that can be used by routing protocols, such as [5], that are designed to find efficient routes in a multi-channel network after the channels have been assigned.

This paper presents our experiences in adapting this theoretical, self-stabilizing, distributed protocol to a real-life wireless mesh environment. A number of practical issues that were overlooked in the theoretical study present themselves here:

- The different channels are not always truly orthogonal such that neighboring nodes' transmissions on different channels may still interfere. However, recent work [15] suggests that utilizing all partially-overlapping channels networks can provide better system performance than using only non-overlapping channels. While that study explores this question in an Access Point setting, we explore it in a ad-hoc setting.
- A node should be able to communicate with some neighbor for every channel it is assigned, and the network communication graph should not be partitioned (a communication path should remain intact between any arbitrarily chosen pair of nodes).
- The hardware configuration may impose restrictions on channel assignment. For example, in our testbed, each node had only two wireless cards installed near one another. Even when the two cards are assigned to orthogonal channels, they would interfere, forcing us to assign channels to them from two different frequency bands. The details are discussed further in Section V.

We describe how we adapt this original theoretical framework to handle the above-mentioned practicalities. In our design, nodes select channels using only locally observed information, and we prove that the distributed channel selection process maintains the self-stabilizing properties of the original theoretical design.

We evaluate the performance of our channel assignment via a set of experiments performed on our 14-node mesh network testbed where each node contains an 802.11a interface and an 802.11g interface. By coupling our channel assignment protocol with the MR-LQSR routing protocol of [5], we provide a complete multi-channel routing solution for our testbed. Using an actual testbed to demonstrate proof-of-concept forces us to address numerous issues that of a real networking environment that are not captured accurately in simulation studies.

While the testbed is admittedly smaller than where we believe our design will have the greatest impact, we still see marked improvement in throughput when running our protocol. In particular, we observe that our channel assignment improves the aggregate throughput of the network on average by 50% compared to the case when all nodes are assigned to the same channels, and by 20% over when channels are assigned at random. Furthermore, our channel assignment benefits particularly the flows over longer paths in the network, which would have typically suffered from poor throughput in multi-hop networks.

After reviewing related work in the following section, we describe the mesh network system architecture and our objectives in greater detail in Section III. Then we present our channel assignment algorithm and protocol in Section IV, and some issues involved in implementing our algorithm is discussed in Section V. In Section VI, we present the performance evaluation results obtained from experiments on our mesh network testbed, and Section VII concludes the paper.

II. Related Work

Much of the recent work in multi-channel 802.11 routing has looked at jointly solving the channel assignment and routing problem. A heuristic solution is looked at in [23], an algorithmic approach that optimizes for throughput is considered in [1], and an approach that preserves network connectivity for QoS is explored in [25]. These are centralized solutions that assume the availability of a global network view (e.g., traffic demand, nodes' status, etc.). In contrast, our modular approach decouples the channel assignment and routing problem separately, with both being solved in a fully distributed manner.

Raniwala et al. [22] propose a distributed channel assignment algorithm for 802.11-based multi-radio mesh network and perform an experimental evaluation. However, the network architecture in [22] is designed for mesh networks specifically used for the wireless Internet access applications, and their channel assignment algorithm works only for routers whose connectivity graph is a tree. In their assignment mechanism, the channel assignment to nodes positioned higher in the tree affects all nodes lower in the tree hierarchy. In contrast, our algorithm can operate on any arbitrary network structure, where every mesh node performs the same assignment task in a fully-distributed manner.

Ramachandran et al. [21] propose a centralized channel assignment algorithm which is performed by a central server that periodically collects dynamically-changing channel interference information. The comparison to this work is of particular interest as their channel selection method takes into account dynamically changing network status (i.e., interference), while our channel assignment is based on more static information (i.e., physical topology). In summary, the performance gain of our mechanism observed in the real-world testbed experiment appears similar to what is shown in their simulation results. This suggests the efficacy of our solution since our distributed mechanism requires only localized interaction between nodes, and does not need to be performed many times once it stabilizes, thus incurring much less overhead in performing the channel assignment than their approach.

In [14], Mishra et al. explore the possibility of utilizing partially-overlapping wireless channels in 802.11 access points, and show that intelligent assignment of non-orthogonal channels increases overall channel utilization

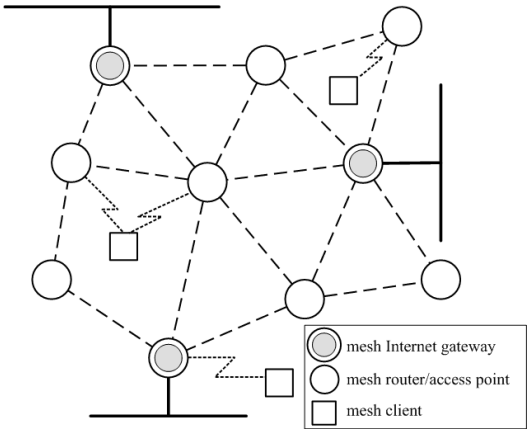


Fig. 1. Wireless mesh network architecture

and the system performance. We integrate their observation into the design of our assignment algorithm, and investigate the efficacy of a simple interference model with our testbed experiment.

Our design of the distributed protocol is motivated by our previous theoretical work on a fully-distributed, self-stabilizing protocol for replica placement [11]. In this work, we propose a distributed replica placement scheme with which identical replicas are placed "far" from one another. We adapt the distributed approach of assigning replicated items to the wireless channel utilization problem, where the channel is the replica. Our work focuses much more on the practicalities of this problem that are ignored in the theoretical work.

III. Architecture and Model

In this section, we present the wireless mesh network architecture considered in this paper, and the overview of the properties of our channel assignment mechanism.

A. Mesh network architecture

Figure 1 depicts a generic wireless mesh network architecture. The mesh network consists of three types of wireless nodes:

- **Mesh clients** are end-user devices, equipped with at least one 802.11 wireless card. They are the users of the mesh network, and each client connects to at least one (typically only one) mesh router to have their packets forwarded from/to mesh gateways or other mesh clients.
- **Mesh routers** are 802.11 wireless nodes (typically stationary) that act as the wireless access points of the mesh clients. These routers form a multi-hop wireless network infrastructure, and forward packets between mesh clients and mesh gateways or between mesh clients using some ad-hoc routing protocol.
- **Mesh gateways** are connected both to the wired network (e.g., Internet) and to 802.11 wireless network, being the relaying points of the traffic between wireless mesh network and external wired network.

Note that a mesh gateway can also serve a dual function and also act as a mesh router, when desired. This dual role may be useful when flow traffic within the mesh network is not only between mesh clients and mesh gateways, but also between pairs of mesh clients. Community surveillance, emergency service, and community resource sharing are potential mesh network applications that would generate a good amount of client-to-client traffic as well as client-to-gateway traffic [2].

Our focus in this paper is to increase the utilization of available spectrum in the wireless multi-hop network. Hence, we focus on how to utilize the 802.11 channels within the wireless network of mesh routers, allowing us to ignore the mesh clients and mesh gateways. We assume each router is equipped with multiple 802.11 interface cards. This multi-radio approach is increasingly accepted as a practical way to improve the capacity of multi-hop network [12], and some commercial multi-radio mesh networks are already in action [6].

B. A Simple Model

Our model of the previously described mesh architecture consists of a set of N nodes, $V = \{1, 2, \cdots, N\}$. There are K wireless channels, $1, \cdots, K$, whose frequency spectrum can possibly overlap. A *channel interference cost function* (simply referred to as *cost function* in the remainder of the paper), $f(a, b)$, provides a measure of the relative interference experienced between channels a and b. The interference cost function is defined in such a way that $f(a, b) \geq 0$ and $f(a, b) = f(b, a)$, where a value of 0 indicates that channels a and b do not interfere with one another. Also, $f(a, b)$ decreases as the gap between channels a and b grows.[1]

A node i belongs to a node j's *interference set* of node j, $i \in S_j$, if there exists a node (either i, j, or possibly a third node k) for which transmissions from i can be corrupted by transmissions from j. In other words, $i \in S_j$ if j's transmissions can corrupt data arriving at or leaving from i.

C. Channel Assignment Objectives

Our goal is to maximize the utilization of the wireless spectrum. Clearly, if one knows the positions and hardware configurations of all nodes in the mesh network and the traffic demands of these nodes, one could perform a centralized allocation of channels to nodes and routes

[1]Our algorithm can use an arbitrary cost function that satisfies the above symmetry. In Section V-C, we discuss about our choice of the cost function used in our testbed implementation.

within the mesh network to maximize the utilization. However, the class of mesh networks we wish to support are not amenable to such a method. In particular, we are interested in mesh networks with the following properties:

- The network contains many nodes, i.e., N is large, such that implementing a centralized algorithm would prove difficult, if not impossible.
- The flow demands are not known a priori, or could change dramatically over short periods of time.
- The network topology is for all intensive purposes static. Nodes or links between nodes may fail or move slowly, and the assignment of channels to nodes and routes within the mesh network should adapt accordingly, but that such changes occur on a timescale much larger than the changes in flow demands. A well-designed distributed solution can handle these small changes gracefully that only affect small portions of the network at a time, with decisions to be made based on local information. Centralized solutions will likely have to reconsider the entire network to update their channel assignment.

We therefore seek a *distributed* mechanism to the channel allocation and routing problems that *stabilizes* to a configuration that maximizes the throughput of an "average" flow within the mesh network. An optimal solution would clearly need to simultaneously consider the channel allocation and routing problems. However, achieving the stability of a good joint solution is extremely difficult, as the selection of channels greatly impacts the desirable set of routes, and the choice of routes affects how one would assign channels within the network to specifically support these routes.

Our preliminary approach therefore is to decouple these two problems. Recent work by [5] has investigated the routing problem, finding "good" routes under the assumption that the channel assignment is given a priori, and developed a protocol, called MR-LQSR (Multi-Radio Link Quality Routing), that can explore channel-diverse paths. Our goal here is to provide a distributed channel selection mechanism that will stabilize to a desirable channel allocation upon which good routes can then be selected.

IV. Distributed Channel Assignment

In this section, we describe our distributed channel assignment mechanism, beginning with the channel selection algorithm in the context of networks with ideal wireless transceivers that can receive data on all channels regardless of the channel selected to transmit. We then proceed to how to adapt this baseline algorithm into a more practical situation where nodes in the network have a limited number of radio cards, each of which can transmit and receive on a single channel at a time.

A. Baseline channel selection algorithm

We begin by making a temporary assumption, which we will relax in the following section, that a node is able to transmit on a single channel that can be selected from any of the K available channels, but can listen to all K channels simultaneously. This assumption simplifies our problem because it allows a node to choose its sending channel in a way that reduces interference with other senders, without worrying about whether the intended receiver is listening on the appropriate channel. Here we would like to assign each node a channel through which it transmits data.[2]

Intuitively, a node would like to choose a channel upon which its transmissions are least likely to suffer interference from other senders' transmissions (on interfering channels). To do this, each node continually seeks to greedily improve its current choice of channel via the following algorithm:

Algorithm 1: **ChannalSelection**(node i)
Input
S_i : Set of nodes in i's interference range.
c_j : The channel of each node $j \in S_i$
c_i : i's current channel
begin procedure
for all $k = 1, \cdots, K$,
$F(k) \leftarrow \sum_{j \in S_i} f(k, c_j)$.
if $F(c_i) > F(k)$ for any $k = 1, \cdots, K$, **then**
$c_i \leftarrow k_{min}$ where $k_{min} = k : F(k) \leq F(k') \quad \forall k' = 1, \cdots, K$
end if
end procedure

In other words, a node i selects a channel that *minimizes the sum of interference cost* from the set of nodes, S_i within its interference range. When there are multiple channels that minimize the interference cost, the node can select one of them arbitrarily. If its prior choice minimized the sum of interference costs, then the node makes no change.

We make two important observations:

- Each node i's choice of channel depends only on information that is available within its local domain, i.e., how many nodes will experience collisions when node i attempts to transmit to them. Hence, the algorithm is truly distributed, using only information available within its local region.
- The efficacy of the node's choice depends on how well this sum of interference costs actually maps to

[2] A more idealistic case of nodes being able to send and receive through all channels can be treated in a similar fashion with channels being assigned to links between nodes.

the interference levels the node experiences. A more ideal solution would be to test every channel and see which one interferes the least with neighboring transmissions. However, it would be unrealistic in a distributed setting to assume that every node could simultaneously perform such checks, varying its own channel selection across the spectrum while its neighbors stay fixed.

It is not intuitively obvious that this distributed channel selection process is self-stabilizing, i.e., that nodes continually looking to improve on their local interference cost will eventually converge to a stable channel allocation; one node's channel change can increase some other node's interference level, and cause the other node to change its channel, and so forth. However, we will next show that indeed this process does stabilize. To prove stabilization, we make some simplifying assumptions about the network environment. Namely:

- Every node i has the correct channel information of all other nodes in its interference range, S_i.
- No other node in S_i changes its channel simultaneously with node i.

In the next subsection, we will present additional detail of our protocol that ensures that these two properties hold. For now, these assumptions permit us to prove the following Theorem.

Theorem 1: If every node selects its channel following Algorithm 1, within a finite number of channel changes by nodes, the channel assignment reaches a stable state where nodes cease changing channels.

Proof: Consider a node $i \in V$ executing Algorithm 1 that, at some time t, begins to change its channel from a channel c_i to another channel c_i' and completes the change at time $t' > t$. For any other node j, let c_j and c_j' be j's channel at time t and t' respectively. By the assumption that no other node changes its channel simultaneously, $c_j = c_j'$ for all nodes $j \in S_j$.

For each node $j \in V$ (including node i), let $F_j = \sum_{h \in S_j} f(c_j, c_h)$, and$F_j' = \sum_{h \in S_j} f(c_j', c_h')$.

Now let $F = \sum_{j \in V} F_j$ and $F' = \sum_{j \in V} F_j'$, i.e., the sum of the interference levels for all nodes before and after node i's channel change, respectively. We will show that $F' < F$.

In Algorithm 1, each node changes its channel only when it can *decrease* the interference level. Therefore, for the changing node i, $F_i' < F_i$. For each node j in S_i, $F_j' = F_j + f(c_j, c_i') - f(c_j, c_i)$ because i is the only node that changes the channel between time t and t'. For all other nodes $j \in V - S(i) - \{i\}$, $F_j' = F_j$ since i's channel change does not affect j's interference level. Therefore, we have the following:

$$\begin{aligned} F' &= F_i' + \sum_{j \in S_i} F_j' + \sum_{j \in V - S_i - \{i\}} F_j' \\ &= F_i' + \sum_{j \in S_i} (F_j + f(c_j, c_i') - f(c_j, c_i)) \\ &\quad + \sum_{j \in V - S_i - \{i\}} F_j. \end{aligned}$$

Since $\sum_{j \in S_i} f(c_j, c_i') = F_i'$ and $\sum_{j \in S_i} f(c_j, c_i) = F_i$,

$$\begin{aligned} F' &= F_i' + \sum_{j \in V - \{i\}} F_j + F_i' - F_i \\ &= F_i' + \sum_{j \in V} F_j + F_i' - F_i - F_i \\ &= F + 2(F_i' - F_i) < F, \end{aligned}$$

where the last inequality holds because $F_i' < F_i$.

The above inequality means F decreases monotonically whenever a node changes its channel. Since F must be greater than or equal to 0 and can take only a finite number of distinct values, F cannot decrease indefinitely and must stop decreasing after finite number of steps of nodes' channel changes, hence Algorithm 1 stabilizes. ■

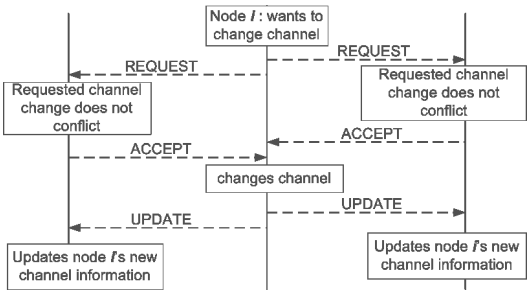
The details of the above proof reveal that each node's distributed decision of changing channel based on its local optimization goal causes the total interference level of all nodes, $F = \sum_{i \in V} \sum_{j \in S_i} f(c_i, c_j)$, to decrease. This is an interesting artifact of the process because it means that each node's greedy choice eventually leads to a channel assignment in which all nodes are satisfied with their channel choice.

B. Distributed protocol

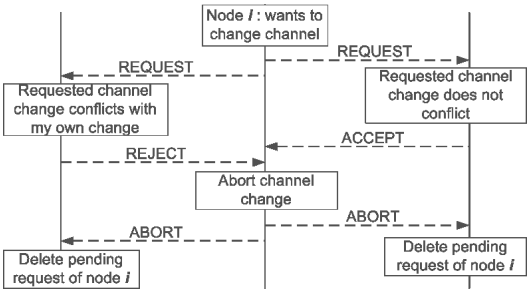
In order to guarantee the stabilization of the algorithm in a fully-distributed manner, we need a mechanism that can satisfy the two assumptions made for the above theorem, i.e., the correctness of channel information and the asynchronous channel changes of multiple nodes within the interference range. For this, we use a distributed protocol, similar to the one used in [11]. We provide here a brief sketch of this protocol component.

Each node exchanges protocol messages with the nodes in its interference range to inform the current channel information and to verify and ensure the information is up-to-date for the correct operation of the channel selection algorithm. To do this, each nodes use a three-way handshake of five types of messages (REQUEST–ACCEPT or REJECT–UPDATE/ABORT).[3]

[3]We assume the reliable, in-order delivery of these messages, and we achieve this reliability using hot-by-hop TCP connection in our implementation.



(a) Node i initiates the channel change request, gets accepted by all other nodes, and changes channel successfully.



(b) Node i initiates the channel change request, gets rejected by some other node, and aborts the change.

Fig. 2. Typical scenario of protocol message flows

In a high level, whenever a node changes its channel, it informs other nodes of the change with an UPDATE message, from which other nodes learn the up-to-date channel information of the changing node. The three way handshake mechanism guarantees that, when a node changes its channel, no other node in its interference range changes the channel simultaneously. The procedure of three-way handshake is as follows (See Figure 2(a) and Figure 2(b) for the message exchange sequences).

First, when a node decides to change its channel according to Algorithm 1, it seeks the 'approvals' from other nodes in its interference set by sending REQUEST messages before actually committing the change. This REQUEST from a node serves two roles: to verify the correctness of channel information used on the channel change decision, and to request the other nodes to remain in the current channel until the node completes the change. This means if the REQUEST is 'approved' by all nodes, the node's decision to change channel is valid in that the change results in the node's interference level to decrease.

Therefore, when a node j receives a REQUEST from some other node i, it decides to approve it or not based on two criteria: if the requesting node has incorrect information about j's current channel, and if j's own channel change 'conflicts' with i's channel change (see the below Algorithm 2 for detailed description of 'conflict'). If none of these two conditions is true, then j approves the request with an ACCEPT message. Otherwise, j disapproves it with a REJECT message. Then the requesting node i commits the change if its request is approved from all nodes (and sends UPDATE), but aborts it if any node disapproves the request (and sends ABORT).

Whether or not a request from some other node "conflicts" is determined by the following algorithm.

Algorithm 2: **IsRequestConflicting**(node j)
Input
i : node that sent the REQUEST.
c_i : i's current channel number.
c_i^{new} : channel number i intends to change to.
c_j : j's current channel number.
c_j^{new} : channel number j intends to change.
begin procedure
if $f(c_i, c_j) > 0$ **then** return **true**
else if $f(c_i, c_j^{new}) > 0$ **then** return **true**
else if $f(c_i^{new}, c_j) > 0$ **then** return **true**
else if $f(c_i, c_j^{new}) > 0$ **then** return **true**
else return **false**
end if
end procedure

In short, a node's channel change conflicts with some other's change if either one's channel change may invalidate the correctness of the other's decision of channel selection based on Algorithm 1.

When a node accepted some other node's request, it is prevented from changing its channel until it receives the subsequent decision (UPDATE or ABORT) from the requesting node.

To prevent a deadlock where all requests keep getting rejected by other nodes, we introduce a pre-defined ordering of nodes (represented by $i > j$ to indicate node i has higher order than node j), which can be easily defined by, for example, nodes' identifiers. If $i > j$ for two nodes i and j and their requests conflicts with each other, i's request is accepted by j while i rejects j's request, hence ensuring at least one node in the network gets accepted by other nodes.

Note that, according to Algorithm 2, it is possible that multiple nodes change their channels at the same time as long as their simultaneous channel changes do not result in changes in the interference level of one another. Since the simultaneous channel changes under this condition does not break the monotonicity argument in the proof of Theorem 1, this can only speed up the whole channel selection process as it provides a higher level of parallelism.

In order to suppress the conflicts (and subsequent rejection) of multiple nodes' channel changes and further increase the "acceptance ratio" of the requests, we intro-

duce a exponentially distributed random delay between the instance of each node's deciding to change its channel and that of its issuing the request to change. More specifically, when a node decides to change its channel, instead of sending REQUEST messages immediately, it waits for a duration chosen exponentially at random. By doing this, each node effectively breaks the potential synchronization with other nodes, and can learn other nodes' channel change (or intention to change), which can often causes the node to stay on the current channel, rather than change it.

C. Limited Channel Reception

So far in this section, we have described our algorithm and protocol under the assumption that nodes can receive across all channels simultaneously. However, current commodity 802.11 devices must be assigned to a single channel that covers both transmission and reception of data. If each wireless card is assigned to a channel so as to minimize interference, it may be assigned to a channel that no nearby neighbor is sending to or receiving on, effectively disconnecting the device on that channel from the rest of the network. This is particularly true when the network is not dense and nodes have only small number of 802.11 interfaces.

1) A common "local path" channel: In order to guarantee that the network graph is not partitioned, and that every node can communicate with at least one other node, we identify a particular channel (e.g., channel 0) as the *default channel*. Every node has one interface card assigned to the default channel. Those devices with additional interfaces can then be allocated to alternate channels.

2) Preventing assignment to a "useless" channel: An ideal assignment of the channels to the remaining interfaces would select channels such that there is little interference on the channel, but that the channel is in fact used. Hence, a node should choose a channel for its interface that is not selected by many nodes in its interference range, but is selected by *at least* one node within the communication range. The assignments to the remaining interfaces are performed using the same algorithm (Algorithm 1) presented in Section IV-A, with one exception: *the channel selected must be from one of those already assigned to some neighbors within communication range*. The channel selection strategy and the distributed protocol remain the same: choose the channel from this restricted subset of choices that minimizes the interference level, and use the three-way handshake.

3) A "grouping" inefficiency: We have identified a potential inefficiency in the channel assignment that can arise when two nodes have competing change requests and the node allowed to change is chosen arbitrarily. Consider the situation where there are four nodes, A, B, C, and D, each with two interface cards that are within close range of one another so that the communication graph is a clique. Suppose they are turned on sequentially (in the order of A, B, C, and then D), each initially choosing its channel at random, with sufficient time between joins to the network. When node B has joined the network after node A, according to our algorithm, one would change to the other's channel to avoid having a "useless" channel. When node C joins to some random channel, eventually C would join the same channel as A and B, or one of A or B might change over to C's channel. The latter event would leave either A or B with a "useless" channel and, following the algorithm, that node too would change over. In any event, after some amount of time, all three nodes will be tuned to the same channel.

When node D joins, one of the following two situations will happen: 1) one of nodes A, B, and C would change to the channel of D, or 2) node D would join other nodes on their channel and all of them would be on the same channel. Which one occurs depends on the sequence of channel change determined by random delay in sending the REQUEST messages. Clearly, the first case is preferred since it makes more efficient utilization of the available spectrum.

4) Preventing grouping: To prevent the above grouping inefficiency, we implement an additional priority ordering among nodes whose change requests conflicts, such that the node with higher interference cost gets precedence. This additional priority relationship precedes the one based on the global ordering explained in Section IV-B when nodes resolve conflicts.

In the context of the above example, consider the scenario where nodes A,B,C,D all reside on the same channel. Whether it has sent its own REQUEST message to the others or not, if it receives a REQUEST message from node A, B, or C, it would accept the request since the requesting node would have more interfering nodes (2 nodes in this case) than it would (0 by initial selection after bootstrapping duration). On the other hand, D's request would be rejected by all three others, and thus one of nodes A, B, and C would eventually change to D's channel (exactly which node would change is determined by the random delay.)

5) Final Thoughts: Given the limited number of channels, this channel assignment strategy described in this subsection is desirable. Connectivity is ensured through the shared channel, giving a type of "local" connectivity . At the same time the devices allowed to choose from the varied channels extend the use of the available spectrum, effectively introducing "express" links between nearby nodes in the network.

Since nodes choose their channels taking into account

both the connectivity and the interference level of the selected channels, we expect the channels to be assigned in such a way that nodes share the same channel with some of their immediate neighbor nodes, while other non-neighboring nodes in the interference range are likely to be on different channels.

V. Discussion

In this section, we consider some issues in implementing our channel assignment mechanism using off-the-shelf 802.11 radio cards. We also discuss the impact of the choice of the interference cost function on the channel assignment.

A. Which Band?

Our protocol is designed to work with arbitrary number of interface cards per node. Our testbed hardware, however, supports only two wireless cards for each node.

The testbed hardware also restricts the way we assign channels to the two cards. We use wireless cards with a PCMCIA form factor. On each machine, the separation between the slots for the cards is less than 2 inches. Due to the physical proximity, the cards interfere with each other if they are assigned channels from the same frequency band. The interference persists even if the channels are non-overlapping (e.g. channels 1 and 11 in 802.11b/g band). This problem has been reported by other researchers [5], [23] as well. One way to overcome the problem is to assign channels from two different frequency bands to the two cards. In other words, we allocate a channel in IEEE 802.11b/g frequency band (2.4GHz band) to one card and a channel in 802.11a band (5GHz band) to the other card. This separation technique has also been used in [5].

As mentioned in the previous section, our design requires that one card on each node be tuned to a common channel. After some initial experiments, we decided to use channel 36 in the 802.11a band as the common channel. Our experiments showed that in our testbed, we could achieve higher overall throughput when the 802.11a band provided the common channel and a varied channel assignment was implemented within the 802.11g band.

B. Interference range

Our algorithm needs information about which nodes are within interference range of each other. The problem of accurately determining the interference patterns within a network is a difficult one [19]. In our experiments, we use the heuristic from [8] that assumes that all other nodes within three hops of X are in X's interference set. This heuristic provides a symmetric definition of interference between a pair of nodes: if node A interferes with node B, then node B interferes with node A. The symmetry is essential to guarantee the stability of our channel allocation algorithm.

C. Impact of interference cost function

How we model the interference between partially overlapping channels in 802.11g radio will affect how channels are allocated by our algorithm. We use a relatively straightforward cost function with a tunable parameter δ:

$$f(a,b) = \max(0, \delta - |a-b|), \tag{1}$$

where channel indices a and b also denote the their center frequencies. We use the difference in channel number of 802.11g channels as the determining factor of the interference cost function since the channel spacing between adjacent 802.11g channels is identically 5 kHz.

Our cost function in Equation (1) implies that the interference decreases linearly with the spectrum distance between channels until reaching 0. As δ increased, the maximum spectrum distance that contains overlapping channels also increases: a large δ translates to a more heavily overlapping channel space, and small δ translates to a more orthogonal channel space. There are two special cases: when $\delta = 1$, all channels are orthogonal from one another, such that interference only occurs between two competing transmissions on the same channel. When $\delta = 0$, no channel is assigned any weight, and $F(k)$ will always equal 0. This case enables us to test randomly assigned channels: when $\delta = 0$ and nodes are initially assigned their channel at random, there is no incentive to change, so the allocation remains in its initial (random) state.

In Section VI, we evaluate the impact of the choice of δ on the channel assignment and the overall system performance, with nodes selecting a channel from all (partially overlapping) 802.11g channels.

VI. Performance Evaluation

In this section, we report the performance evaluation results, obtained via experiments on our 14-node mesh network testbed. We start by describing our mesh network testbed environment.

A. Mesh network testbed

Our mesh network testbed consists of 14 nodes, placed throughout three floors of a multi-story building (see Figure 3 for the locations of the nodes in one of the floors). Each mesh node is an Intel Pentium 4 desktop PC, equipped with two IEEE 802.11a/b/g wireless interface cards: a NetGear WAG311, D-Link DWL-AG530, Linksys Wireless A+G, and an Orinoco 802.11abg ComboCard Gold card. The cards are configured in ad-hoc mode. All nodes run the Microsoft Windows XP (SP2)

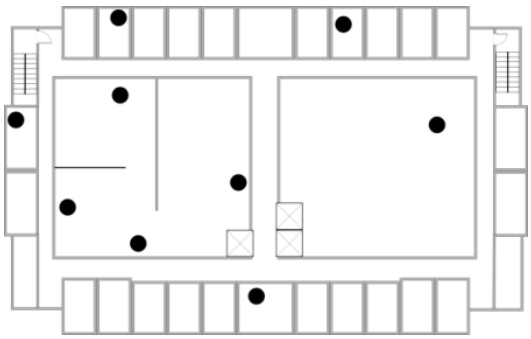


Fig. 3. Wireless mesh network testbed : node locations within a floor of a multi-story building

operating system and they have statistically assigned IPv4 addresses in a private IP address domain.

Our channel assignment module resides on top of TCP/IP with a WinSock2 API interface. It also interfaces with a few other lower layer modules in order to query wireless link-layer information and routing information (MCL [18]), to set and query 802.11 device configurations (NDIS [17]), and to reconfigure the interface card (devcon [4]).

To forward packets across the multi-hop network, we use LQSR protocol with the WCETT metric [5]. The WCETT routing metric is designed to select channel-diverse paths in multi-radio environments.

B. Methodology and parameters

To evaluate the capacity of the network, we measure the aggregate throughput achieved by multiple simultaneous TCP flows. Each node in the network acts as a source of a TCP flow. Thus, we have 14 TCP flows in our network. The destination of each TCP flow is chosen at random from remaining 13 nodes, while ensuring that there are no single-hop TCP flows in the network (i.e., between nodes within transmission range of one another). Previous studies [7] have shown that single-hop TCP flows will significantly dominate multi-hop flows in a mesh environment. By using only multi-hop flows in our traffic pattern, we avoid this bias in our results.

All flows start simultaneously. Each flow sends data as fast as TCP permits for 120 seconds. We do not claim that this traffic pattern is realistic; we have deliberately chosen it to create heavy load on our testbed.

One network card of each node is assigned to a single, common channel (channel number 36 at 5180 kHz, the lowest channel in the 802.11a radio spectrum). The other network card is assigned one of 11 channels in 802.11g radio in the following manner.

We first consider three baseline channel assignment strategies:

- **samech** (same channel assignment): all nodes are assigned the same 11g channel.
- **11-rand** (random assignment with all 11 channels): each nodes is assigned one of 11 802.11g channels uniformly at random. This assignment corresponds to the case of $\delta = 0$ in the context of our cost function.
- **3-rand** (random assignment with 3 orthogonal 802.11g channels): each nodes is assigned one of three orthogonal 802.11g channels (i.e., channel 1, 6, and 11) uniformly at random.

Then, we consider channel assignments generated by our distributed protocol using three different values for $\delta = 1$,3, and 5 in the interference cost function, and refer the respective assignments as ***δ*=1**, ***δ*=3**, and ***δ*=5**.

Note that δ=5 implies that only channels 1, 6 and 11 are truly orthogonal. Smaller values of δ allow our algorithm to aggressively assign partially-overlapping channels.

For each of the six channel assignment strategies described above (3 baselines + 3 for different δ values), we generate 5 different channel allocations. The different allocations are generated by using different random seed, and for the samech assignment, we used channel 1, 3, 6, 9, and 11 in 802.11g radio for those 5 allocations. Thus we have 30 total channel allocations. For each channel allocation, we run the 4 traffic sets denoted by **fset1**, **fset2**, **fset3**, and **fset4**, each of which consists of 14 TCP flows as described earlier, . The difference between the four traffic sets is that the flows have different destinations.

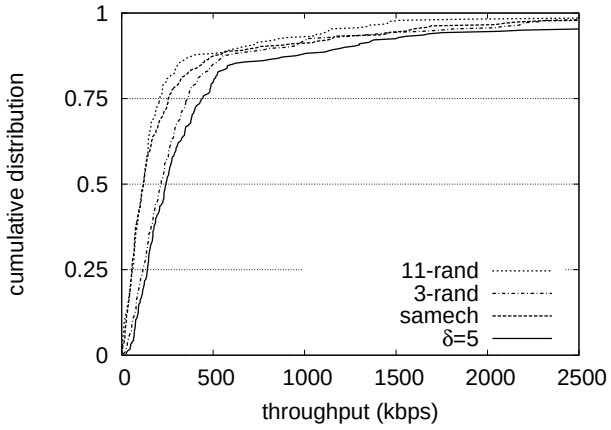
We measure the TCP throughput of the flows under each channel allocation for each flow set.

During the experiment, the RTS-CTS handshake and the auto-rate control of 802.11 wireless cards are turned on, and we set the parameter β of MR-LQSR to 0.5, where β is the parameter that controls the weight given to channel-diverse paths in MR-LQSR's path selection. All other TCP/IP configuration parameters are set to their default values.

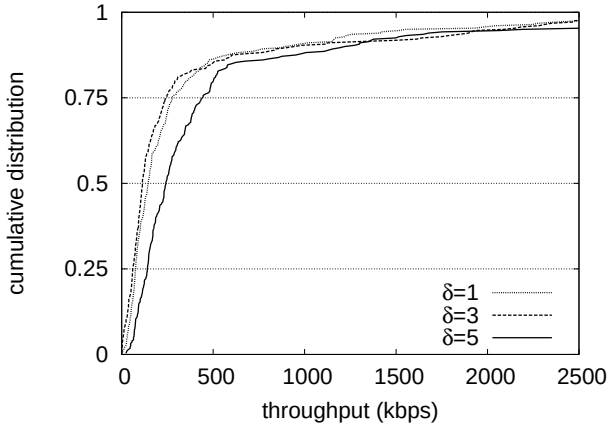
C. Throughput

In Figure 4(a), we compare the CDF of throughput of δ=5 channel assignment strategy to the baseline channel assignment strategies. Recall that for each strategy, we consider 5 channel allocations, and for each allocation, we consider 4 traffic sets of 14 flows each. Thus, each CDF is based on of $5 * 4 * 14 = 280$ throughput measurements.

The x-axis in the figure represents the throughput in kbps, and the y-axis indicates the ratio of the number of flows that achieve the end-to-end throughput up to the value in x-axis. Table I shows 25^{th}, 50^{th}, and 75^{th} percentile values.



(a) Comparison to baseline allocation strategies



(b) Comparison between different choice of δ

Fig. 4. Cumulative distribution : throughput of individual TCP flows

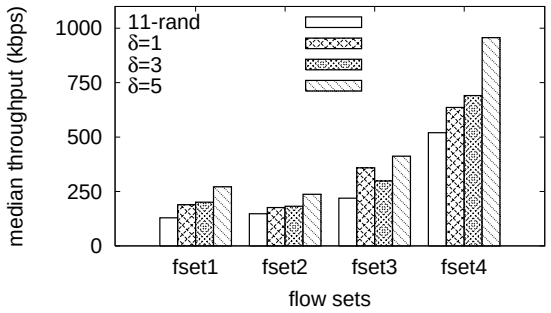
The performance gain by our mechanism is clear. For instance, at the 75-percentile mark, the throughput by our algorithm with δ=5 reaches 440 kbps, while the best of the other assignments (3-rand) achieve no more than 350 kbps. Also the benefit of δ=5 assignment is the most noticeable in the region that the per-flow throughput is relatively small (up to 75% percentile mark, or below 500 kbps). These lower-rate flows typically traverse more hops, and hence are more adversely affected by high interference conditions. Hence, we see these flows benefit significantly from a well-designed channel selection protocol.

In Figure 4(b), we compare the impact of the choice of δ for our channel assignment. It can be clearly seen that δ=5 results in the best per-flow throughput among different cost function parameters. This result is an anticipated one since the cost function with $\delta = 5$ reflects the fact that channels separated by at least 5 channel(e.g., channel 1 and 6) are in fact orthogonal to one another.

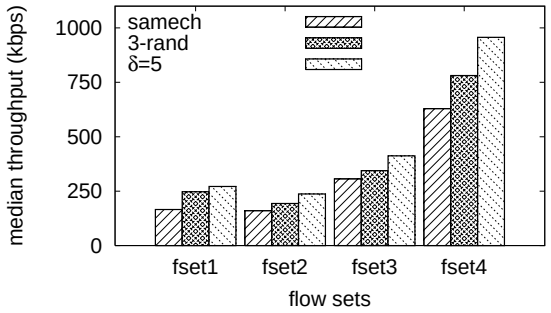
Assignment	25%	50%	75%
samech	55.1	119.4	254.5
11-rand	60.4	118.4	208.2
3-rand	115.0	214.4	357.0
δ=1	76.1	144.5	275.6
δ=3	60.7	112	239.9
δ=5	140.1	241.6	440.9

TABLE I

PER-FLOW THROUGHPUT (IN KBPS) THAT 25%, 50%, AND 75% OF FLOWS ACHIEVE. FIRST THREE ASSIGNMENTS ARE BASELINES, LAST THREE ARE FROM OUR ALGORITHM.



(a) Median throughput of 11-rand, δ=1, δ=3, and δ=5



(b) Median throughput of samech, 3-rand, δ=5

Fig. 5. Comparison of Median throughput in four traffic sets

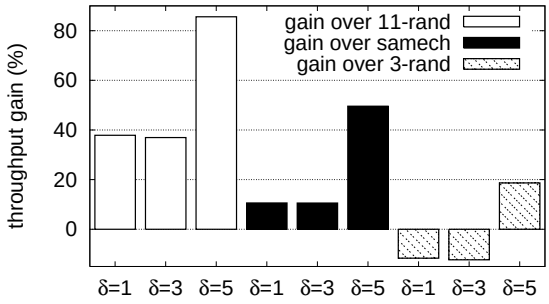


Fig. 6. Gain(in %) in median throughput of δ=1, δ=3, and δ=5 over 11-rand, samech, and 3-rand

We now look at the results in more detail. In Figure 5, we consider median throughput for each traffic set,

for different channel assignments. Note that each traffic set has 14 flows, and for each channel assignments, we generate 5 channel allocations. So each bar represents median of $14 * 5 = 70$ flows.

In Figure 5(a), we compare performance of our channel assignment strategy to the 11-rand strategy. Our channel assignment outperforms 11-rand assignment for all three values of δ, for all four traffic sets. As noted earlier, δ=5 assignment provides the best performance. This is an impressive result because nodes running our algorithm can also select any channel from all 11 partially-overlapping channels with the cost function simply acting as a 'guideline' for the selection.

In Figure 5(b), we compare the best case of our channel assignment (i.e. δ=5) to the two other baselines (i.e. samech and 3-rand). We see that δ=5 outperforms the best case among the other assignments (3-rand) by 20%.

Figure 6 shows the percentage improvement in median throughput achieved by our channel assignments (for three values of δ) over the three baseline assignments(11-rand, samech, and 3-rand).

Our assignments significantly outperform 11-rand assignment by 40 to 80%, and samech assignment by 10 to 50%. The comparison to samech assignment is particularly interesting since it indicates that utilizing even partially overlapping channels exhibits better performance than tuning the interface cards of nodes to the same channel.

The performance gain of our assignments is lower when compared to the 3-rand assignment. While the δ=5 assignment outperforms 3-rand by 20%, lower values of δ perform worse than 3-rand.

We suspect that the relatively good performance of 3-rand assignment is an artifact of our testbed setup, in which many nodes are densely clustered together around the center of the network. When only 3 channels are randomly assigned, nodes are likely to have some neighbor assigned on the same channel in this dense area, and the probability that the assigned channel is isolated becomes very low. Considering a good portion of traffic would be routed through the middle of the network, it follows that this random assignment would provide a relatively good end-to-end throughput in our testbed. Note however, that by assigning the three channels intelligently (i.e. by using δ=5), our algorithm still provides a 20% gain. We believe that in a larger network this gain would be even higher.

Overall, we can see that our mechanism strikes a good balance between two conflicting requirements of maintaining connectivity between nodes of same channel and minimizing the interference from nodes of same channel.

samech	11-rand	3-rand	δ=1	δ=3	δ=5
10.1	3.0	15.1	16.3	11.6	22.7

TABLE II

CHANNEL UTILIZATION(IN %): PERCENTAGE OF TCP THROUGHPUT CARRIED ON 802.11G CHANNELS

	δ=1	δ=3	δ=5
1) messages	51.2	79.5	90.0
2) bytes	1205	1855	2080
3) time (sec)	23.4	40.6	32.4
4) changes	0.14	0.15	0.22
5) requests	0.24	0.34	0.70

TABLE III

AVERAGE PROTOCOL DYNAMICS OF NODES: 1),2) NUMBER OF MESSAGES/BYTES TRANSMITTED, 3) TIME ELAPSED (INCLUDING CHANNEL SWITCHING TIME) UNTIL STABILIZATION, 5) NUMBER OF CHANNEL CHANGES, 6) NUMBER OF CHANNEL CHANGE REQUESTS

D. Channel utilization

Now we investigate how the channels are utilized in the experiments. Table II shows the utilization of the 802.11g band, where utilization measures the percentage of end-to-end traffic carried on channels on the 802.11g band. For instance, under our channel assignment with δ=5, the wireless links supported by 802.11g channels collectively carried 22.7% of end-to-end traffic, while links of the common 802.11a channel contributed for the other 77.3%.

The utilization of 802.11g channels is in large part lower due to the active usage of 802.11b/g infrastructure network around our testbed. Nevertheless, we see that our channel assignment (especially δ=5) makes better use of 802.11g channels, and thus reduces the congestion on the common 802.11a channel.

E. Protocol dynamics

Table III shows statistics regarding the operation of our distributed protocol until it stabilizes. 'Messages' and 'bytes' mean the average number of messages (and amount of data) a node transmits to other nodes during the channel selection process. Also, 'time' is the *convergence time* of our protocol, meaning the average time that has elapsed until each node ceases to transmitting protocol messages beginning with randomly assigned channel. Finally, 'changes' and 'requests' are the average number of channel changes and change requests that each node makes.

We can see from this data that nodes exchange only a small amount of data for the protocol operation. δ=5 generates the most protocol overhead shown in terms of the number of messages and change requests issued. This is because the wide interference range implied in large value of δ effectively narrows each node's choice of channel. This in turn triggers many change requests

that conflict with other nodes' channel selection. Nevertheless, the overall usage of network resource shows our protocol is very light-weight, and the convergence time of the protocol is quite small. Since each node only needs to interact with those in the interference range, we believe our distributed protocol is suitable for the channel assignment task in large-scale wireless mesh networks.

VII. Conclusion

We presented a fully-distributed mechanism that assigns 802.11 channels to multi-radio nodes in wireless mesh networks. We showed that our assignment algorithm stabilizes to a desirable channel configuration that routing protocols can exploit to provide better end-to-end system performance. Our design takes into account several constraints present in current 802.11 devices, and its distributed nature ensures it is sufficiently lightweight to be executed on large-scale mesh networks.

The modular design that decouples channel selection from data forwarding makes our solution readily available for operation, providing a complete solution to wireless mesh networks in combination with existing routing protocols.

We ran experiments on our small wireless mesh testbed and showed that our channel assignment can increase the capacity of wireless mesh network between 20% and 50% over other conventional channel selection mechanisms. In the future, we plan to expand the size of the testbed. We anticipate even greater improvement in comparison to conventional methods in larger settings.

References

[1] Mansoor Alicherry, Randeep Bhatia, and Li (Erran) Li. Joint channel assignment and routing for throughput optimization in multi-radio wireless mesh networks. In *MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking*, pages 58–72, New York, NY, USA, 2005. ACM Press.

[2] P. Bahl. Opportunities and challenges of community mesh networking, keynote @ mics workshop eth zurich, july 6, 2004.

[3] Alan A. Bertossi and Maurizio A. Bonuccelli. Code assignment for hidden terminal interference avoidance in multihop packet radio networks. *IEEE/ACM Transactions on Networking*, 3(4):441–449, 1995.

[4] The DevCon command-line utility. *http://support.microsoft.com/*.

[5] R. Draves, J. Padhye, and B. Zill. Routing in multi-radio, multi-hop wireless mesh networks. In *MobiCom'04*, 2004.

[6] Mesh Dynamics. *http://www.meshdynamics.com/*.

[7] M. Gerla, R. Bagrodia, L. Zhang, K. Tang, and L. Wang. Tcp over wireless multihop protocols: Simulation and experiments. In *Proceedings of IEEE ICC' 99*.

[8] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE/ACM Transactions on Information Theory*, 46(2), March 2000.

[9] Z. Haas. A new routing protocol for the reconfigurable wireless networks. In *Proc. IEEE Int. Conf. on Universal Personal Communications*, 1997.

[10] David B Johnson and David A Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[11] Bong-Jun Ko and Dan Rubenstein. Distributed self-stabilizing placement of replicated resources in emerging networks. *IEEE/ACM Transactions on Networking*, 13(3):476–487, 2005.

[12] Pradeep Kyasanur and Nitin H. Vaidya. Capacity of multi-channel wireless networks: impact of number of channels and interfaces. In *MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking*, pages 43–57, New York, NY, USA, 2005. ACM Press.

[13] Jinyang Li, Charles Blake, Douglas S. J. De Couto, Hu Imm Lee, and Robert Morris. Capacity of ad hoc wireless networks. In *Proceedings of the 7th ACM International Conference on Mobile Computing and Networking*, pages 61–69, Rome, Italy, July 2001.

[14] A. Mishra, E. Rozner, S. Banerjee, and W. Arbaugh. Exploiting partially overlapping channels in wireless networks: Turning a peril into an advantage. In *IMC '05*.

[15] Arunesh Mishra, Suman Banerjee, and William Arbaugh. Weighted coloring based channel assignment for wlans. *SIGMOBILE Mobile Computing and Communication Review*, 9(3):19–31, 2005.

[16] A. Nasipuri, J. Zhuang, and S. Das. A multichannel csma mac protocol for multihop wireless networks. In *Proc. of IEEE Wireless Communications and Networking Conference (WCNC'99)*, September 1999.

[17] NDIS. *http://www.microsoft.com/whdc/*.

[18] Self-Organizing Neighborhood Wireless Mesh Networks. *http://research.microsoft.com/mesh/*.

[19] J. Padhye, S. Agarwal, V. Padmanabhan, L. Qiu, A. Rao, and B. Zill. Estimation of link interference in static multi-hop wireless networks. In *Short paper, IMC 2005*.

[20] C. Perkins. Ad hoc on demand distance vector (aodv) routing. In *IETF, Internet Draft, draft-ietf-manet-aodv-00.txt*, November 1997.

[21] Krishna N. Ramachandran, Elizabeth M. Belding-Royer, Kevin C. Almeroth, and Milind M. Buddhikot. Interference-aware channel assignment in multi-radio wireless mesh networks. In *to appear in IEEE Infocom'06*.

[22] Ashish Raniwala and Tzi cker Chiueh. Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network. In *IEEE Infocom'05*.

[23] Ashish Raniwala, Kartik Gopalan, and Tzi cker Chiueh. Centralized channel assignment and routing algorithms for multi-channel wireless mesh networks. *Mobile Computing and Communication Review*, 8(2):50–65, 2004.

[24] Jungmin So and Nitin H. Vaidya. Multi-channel mac for ad hoc networks: handling multi-channel hidden terminals using a single transceiver. In *MobiHoc '04: Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, pages 222–233, New York, NY, USA, 2004. ACM Press.

[25] Jian Tang, Guoliang Xue, and Weiyi Zhang. Interference-aware topology control and qos routing in multi-channel wireless mesh networks. In *MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, pages 68–77, New York, NY, USA, 2005. ACM Press.

[26] S.-L. Wu, C.-Y Lin, Y.-C. Tseng, and J.-P. Sheu. A new multi-channel mac protocol with on-demand channel assignment for multi-hop mobile ad hoc networks. In *ISPAN '00: Proceedings of the 2000 International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN '00)*, page 232, Washington, DC, USA, 2000. IEEE Computer Society.