Allocation of Channels in Wireless Tree Topologies

Dragoș Niculescu ETTI, University Politehnica of Bucharest, Romania
Sudeept Bhatnagar Airtight Networks, Sātāra, India
Samrat Ganguly NEC Corporation of America, USA

***Abstract*— Judiciously assigning channels to a wireless mesh network can substantially enhance capacity of the network. One particular flavor of mesh network is that with a tree topology, which has the property that all traffic passes through one central point. Usually the allocation problem is linked to problems of routing, load, and measurements of interference. In this paper we take advantage of the restrictions on the routing, and on potential load imposed by the topology, to present an allocation algorithm that is tailored for wireless trees. Using the connection characteristics of the topology, we define a conflict graph over which a coloring heuristic can provide better performance by not having to focus on connectivity/routing/interference. We show that the algorithm has a low complexity (quadratic in the number of nodes), and in simulation shows significant gains in performance when compared to simpler solutions.**

I. INTRODUCTION

The capacity of a wireless mesh network is affected by the degree of channel reuse among its wireless nodes. IEEE wireless standards have the provision to use multiple channels in the network and mesh nodes typically have multiple radio interfaces which can be configured to use distinct channels. An efficient channel allocation algorithm can exploit the channel diversity provided by the wireless standards to maximize the capacity of a wireless mesh network by tuning different interfaces to different channels.

A channel allocation algorithm may have the topology knowledge and optionally, the traffic pattern information to aid in its decision making. This work deals with a specific case of wireless mesh networks where the routing structure is a tree. In this topology, all the mesh nodes act as the access points for clients and all the traffic flows through the root node. The root node acts as the gateway for all communications. Each mesh node forwards its traffic towards the root possibly over multiple hops and all the traffic meant for that mesh node follows the reverse path from the root to itself. We leverage these characteristics to design an efficient channel allocation algorithm specifically for tree-structured wireless mesh networks.

II. NETWORK MODEL & PROBLEM STATEMENT

We consider a wireless mesh network of N nodes. Each node has two interface cards to communicate on the mesh. The mesh nodes may also have a third interface card which is used to communicate with the clients. The third interface card uses a different carrier than the two used for communicating with the mesh nodes (for example 11a vs. 11b). The interference set I_k for node k is defined as the set of nodes which can interfere with k if they transmit on the same channel (using either of the interfaces). We consider the Boolean model of interference where a node either interferes or does not interfere with any other node. There is no notion of partial interference or the actual amount of interference (packet loss) that a node causes. The simplicity of this interference model is important to keep the interference information obtainable in a practical setting.

The routing structure for the network is a tree. All nodes act as access points for the clients and route the traffic to/from the root. Each of the other nodes uses its mesh access cards as upcard or downcard (Figure 1). All traffic towards the root is sent on the upcard and all traffic from the root meant for one of the downstream mesh nodes is forwarded on the downcard. All traffic to/from the client uses separate client access cards. The root node uses both its mesh access cards as downcards since it does not have any parent in the mesh. This serves the purpose of increasing the mesh capacity since the root is likely to become the bottleneck in such a scenario. Using two downcards (on different channels) doubles the amount of traffic that the root can source.

A node k has t_k units of traffic to send towards the root. Since all traffic goes to/from the root, the traffic that a mesh node sends to its parent (towards the root) is the sum of traffic from all its children and the traffic generated by the clients. Similarly, all traffic it receives from its parent is the sum of traffic meant for its clients and its children. Thus, the total traffic at the nodes higher up in the tree is always going to be higher than the total traffic at the nodes lower in the tree.

There are a fixed number of channels that we can use. For example, 802.11b has 3 orthogonal channels and 802.11a has 12. The exact number of channels that we can use is an input to our algorithm. The channels are considered orthogonal, i.e., each channel is essentially unaffected by the traffic on any other channel. We do not consider partially overlapping channels like those possible in 802.11b. Our goal is to assign channels to the mesh access cards of the mesh nodes. We do not assign channels to the downcards of the mesh nodes which do not have any children in the tree (essentially these cards are not used). We do not consider the channel allocation between the mesh node and its clients and focus on the mesh portion only. Furthermore, we are not concerned with the routing portion of the problem; in fact in our model where all traffic flows to/from the root, the routing structure is equivalent to

Parts of this work were performed while authors were employed by NEC Laboratories America, Princeton NJ

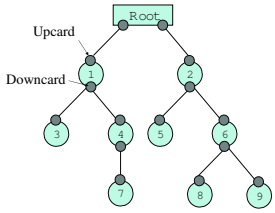


Fig. 1. A wireless mesh network with tree topology

the tree topology. However, we recognize that joint channel allocation and routing is an interesting problem.

III. CHANNEL ALLOCATION

Since all traffic flows through the root, we know that the number of cards that the root node can have is not more than the total number of channels available in the technology the mesh uses. Otherwise, more than one of the cards that the root hosts will be forced to communicate on the same channel thereby splitting the throughput and rendering the extra card useless. In this section, we describe our algorithm to assign channels on the tree topology. The algorithm utilizes the properties of the traffic on the tree to assign channels. The algorithm also uses the information regarding which pairs of mesh nodes are within each others carrier sense range.

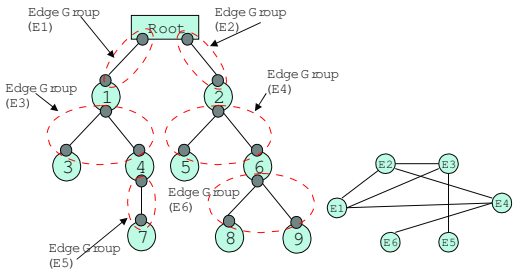
However, one of its limitations is that it does not use any information regarding the interference caused at a node by other nodes that are beyond its carrier sense range. However, our algorithm has been intentionally designed to use this simple information as interference modeling and measurement are not yet fully understood and may require some amounts of network downtime to build. Our algorithm is designed to use only the information which may easily be deduced in real scenarios.

A. Insights on the Tree Topology

We utilize the following information that is valid on a mesh network with a tree topology but not in a generic mesh:

- Since all traffic flows to/from the root, the total traffic at a node higher up in the tree (closer to root) is never less than the traffic at the nodes in its sub tree. In fact, the total traffic between a node and its parent is equal to the sum of the total traffic between itself and its children and the traffic that is sourced by the clients directly attached to it. This because the mesh node merely relays traffic from(to) the children node and its own clients, to(from) the parent node.
- It is better to avoid contention at the links that are higher up in the tree. This because if contention reduces the throughput of a link higher up in the tree, that reduction affects the traffic to/from all the nodes in the sub tree below the link. Hence if channel overlap is unavoidable, it is better to allow interference and higher contention in the lower level links which have less traffic than the higher level links anyway.

We use these insights in designing our algorithm. Before describing the algorithm, we first detail the creation of an auxiliary contention graph from the network topology to ease the task of channel assignment.



(A) Edge groups (B) Contention graph

Fig. 2. (A) Creation of contention graph with the following interface model: Edge groups interfere if they have a common node; nodes 1 and 2 interfere (less than carrier sense distance). (B) Edge groups become nodes in the contention graph

G – set of all edge groups
l_e – load of edge group $e \in G$
h_e – level of edge group $e \in G$
I_e – interference set of edge group $e \in G$
$next(x)$ – element after x in LLPQ
C– set of channels

Fig. 3. Notations used in the algorithm

B. Contention Graph

One key requirement of the channel allocation algorithm is to allocate channels so that the connectivity of the network topology is maintained. In case of a tree topology, a channel allocation algorithm must never allocate separate channels to the up interface of a node and the down interface of its parent (else they would not be able to communicate).

To eliminate this possibility, we create an auxiliary graph over which our channel allocation algorithm executes. We create *edge groups* (Figure 2) out of interfaces that must communicate on the same channel. Thus, all edges between a node and all its children constitute one edge group. The level of an edge group is defined as the level of the parent node for all its edges. All edge groups corresponding to interfaces that are hosted on the root have a level 0. The level of the edge group e is denoted as h_e (height). The notations used throughout the rest of the paper are summarized in Figure III-B.

In the auxiliary graph, each edge group is treated as a vertex. We create an edge between two vertices A and B if any constituent interface of the edge group corresponding to A is in carrier sense range of any interface in the edge group corresponding to vertex B. How to decide wether interfaces are within carrier sense of each other? In simulation, this is a decision based on distance, but in a testbed pairs of nodes will have to measure wether they can access the medium at the same time. The load on an edge group is then defined as the total traffic of all nodes in that group. Note that the traffic demand t_k at node k is part of the load of the edge group that contains its upcard and not the one that has its downcard. This because all traffic is destined for the root node in our case. The load on an edge group e is denoted as l_e . Furthermore, all edge groups that are neighbors of edge group e in the contention graph (are interferers) form its Interference set I_e. The channel

Algorithm 1 Algorithm to assign channels to edge groups

Step 1: Create Priority Queue(G, l_e, h_e)
Initialize empty *LLPQ*
foreach $e \in G$
  $x \leftarrow \{y \in LLPQ | ((h_y < h_e) \&\& (h_{next(y)} \geq h_e))\}$
  if($x \neq \phi$)&&($h_x == h_e$)
    while $((l_x \geq l_e) \&\& (h_{next(x)} == h_e))$
      $x \leftarrow next(x)$
    endwhile
    insert e after x in *LLPQ*
  endif
endfor

Step 2: Assign_Channels
Virtual_Capacity $\leftarrow max_{e \in G} l_e$
while LLPQ is not empty
  e $\leftarrow$ **extract_first_element**(*LLPQ*)
  if $\exists$ unused channel $c \in C$
    assign channel c to e
    continue
  endif
  if ($\exists c \in C$) || (l_e + **Used_Capacity**(c,e) $\leq$ *Virtual_Capacity*)
    assign channel c to e
    continue
  endif
  chan $\leftarrow$ (-1)
  clevel $\leftarrow \infty$
  cload $\leftarrow \infty$
  foreach $c \in C$
    t $\leftarrow$**Lowest_Level**(c, e)
    u $\leftarrow$**Used_Capacity**(c, e)
    if(t < *clevel)* || *((*t == *clevel) && (cload* > *c))*
      chan $\leftarrow$ c
      cload $\leftarrow$ u
      clevel $\leftarrow$ t
    endif
  endfor
  assign channel chan to e
endwhile

allocation algorithm assigns channels to the vertices in the auxiliary graph. All interfaces corresponding to a vertex will be tuned to that channel. An example of a tree structured mesh and its corresponding contention graph is shown in Figure 2.

C. The Spread Algorithm

The algorithm aims at coloring the previously constructed contention graph with a given number of colors (corresponding to the number of available channels in the network). It has the level and the load information of each vertex at its disposal. We refer to the solution as the *Spread Algorithm* (first described in *[1]*) since it primarily tries to spread the channels far apart depending on the load.

1)Vertex Traversal Order: The algorithm takes a single pass assigning colors to each vertex of the contention graph. Clearly, the order in which the vertices are visited plays an important part in the effectiveness of the channel allocation. If we visit an "important" vertex after assigning orthogonal channels to relatively unimportant vertices earlier, we may end up having to allocate already crowded channel to the important vertex leading to a loss in overall throughput. In our case, the vertex traversal order is determined both by the level and the load of the vertex. We choose to visit a vertex at a lower level (higher up in the tree) early in the algorithm. If there are more than one such vertices, the priority is given to the vertex with a higher load. For efficient traversal, we build a priority queue using this logic. We call this queue the Level Load Priority Queue (LLPQ). The construction of LLPQ is shown at the top of algorithm 1.

2) Channel Allocation: The channel allocation procedure gets the next edge group to assign channel by extracting the front element of the LLPQ. The impact of assigning each channel to this edge group is tested and it is assigned the channel whose throughput is expected to be affected the minimum with the new edge group operating on it. The question is how to deduce the impact of the edge group's traffic on the throughput of that channel.

Our rationale in determining the assignment utilizes the tree specific observation: If an edge group higher up in the throughput is affected, then the collective traffic from all nodes underneath that edge group is also affected. Thus, it is better to avoid hurting the higher level edge groups as much as possible. If there are multiple groups at the same level that are within the interference range of the group being assigned a channel, a better design choice is to use the channel of the less loaded group. These two insights serve as guidelines for our algorithm.

An important concept used in our algorithm is that of *virtual capacity*. The virtual capacity of the network is simply the maximum l_e over all edge groups (utility functions are detailed in Algorithm 2). This serves as a guideline for maximum throughout in our tree based mesh. In an ideal scenario, whatever the absolute value of total load be, the edge group carrying the traffic equivalent of the virtual capacity is going to carry the maximum traffic. For example, if all nodes source unit traffic, then one of the two edge groups at the root is going to be the one determining the virtual capacity and also carrying the maximum traffic. Thus if that edge group is assigned a distinct channel, any other channel can be thought of as having a similar capacity (irrespective of the actual value of the traffic). This is because of the implicit relationship between virtual capacity and the maximum possible traffic at any one interface.

The algorithm proceeds by extracting the first element (say E) of the LLPQ. It then checks the assigned channels for the neighbors of E in the contention graph. If there is a channel not yet assigned to any neighbor, E is then assigned that channel.

Algorithm 2 Functions used by Algorithm 1

Used Capacity(c, e)
total_load $\leftarrow$0
foreach $x \in I_e$
  if $assigned\_channel(x) == c$
    $total\_load \leftarrow total\_load + l_x$
  endif
endfor
return $total\_load$

Lowest Level(c, e)
clevel $\leftarrow \infty$
foreach $x \in I_e$
  if (*assigned_channel*(x) == c)&&(h_x < clevel)
    clevel $\leftarrow x$
  endif
endfor
return *clevel*

If not, if there exists a channel whose load is low enough that even after reusing it for E, its total load remains below the virtual capacity, we assign it to E. If no such channel exist, i.e., for all channels there is a neighbor of E which is using it or if its total load will go beyond the virtual capacity, we find the channel for which the highest using edge group (among E's neighbors) has the highest level in the network. For example, if Channel 1 is being used by an edge group at level 2 and channel 2 is being used by an edge group at level 1 (both edge groups being neighbors of E in the contention graph), then we assign channel 1 to E. This rationale ensures that the channel used by E would try to minimize the hurt on the nodes higher up in the tree. If for two channels the highest level edge groups are at the same level, the channel having lower load is preferred for E. The algorithm is shown as algorithm 1. The algorithm's running time is $O(N^2)$ which is the time required to construct the contention graph. The channel assignment operation (assuming fixed number of channels) require less time since it primarily consists of scanning the neighborhood of each node to find the load on a given channel in its vicinity.

IV. Evaluation

We now evaluate our channel allocation algorithm using simulations. For simulations we implemented our channel allocation algorithm along with two other possible schemes:

- Layered Allocation: In this scheme, the two edge groups containing the two interface cards at the root node are allocated two separate channels (say channels 1 and 2). The remaining channels are allocated based on the level of an edge group. In particular, edge group at level 1 is allocated channel 3, the next level channel 4 and so on. When all channels are exhausted and still edge groups remain without channels being assigned, we start using channel 1, 2, and so on.
- Random Allocation: In this scheme, each edge group is assigned a randomly chosen channel.
- Optimal Throughput: In this case, we assign each edge group an orthogonal channel so that there is no cross edgegroup interference. This represents the case where the total throughput for the given topology, carrier sense range, and traffic pattern is the maximum possible. This serves as a guideline as to how close we get to the maximum possible even with a constrained number of channels. This value is reported in the caption of each result figure.

The implementations take as input the topology along with the traffic from each node to/from root. The algorithms then generate channels for each edge group. To test the quality of an assignment, we extended the ns-2 network simulator to handle multi interface multichannel topology. In the evaluation topology, each node has two interfaces, one each to communicate with its parent and its children. The root node uses both of its interfaces to communicate with the children (the left and the right sub tree). The leaf nodes have a single interface to communicate with their respective parent nodes.

The simulations are done on two topologies shown in Figure 4. The distance of a parent child link is set to 20m in each case, and the carrier sense range is set at 40m. This is consistent with indoor measurements from popular Atheros based 802.11a cards operated indoors [2]. The "snowflake" topology represents a dense case where the depth of the tree is less but the number of mesh nodes is high to provide coverage. The "fish" topology represents a case where two corridors of a rectangular building are covered. In this case, the depth of the tree is high for a smaller number of nodes. The base case is the string topology where the root only utilizes one of its interfaces.

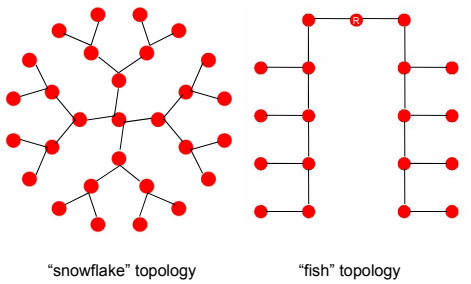


Fig. 4. Topologies used for evaluation: "snowflake" topology can be used as an access network for a circular area; "fish" network for a rectangular area.

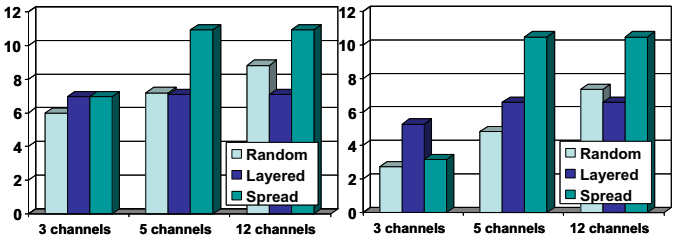


Fig. 5. Performance [Mbps] for "snowflake" topology: each card at the root can handle 6Mpbs (750Kbps per leaf). Optimal throughput=10.94Mbps. left: unidirectional right:bidirectional

We simulate two traffic patterns for these topologies: 1) Unidirectional traffic where all traffic flows from the root to the other nodes. The traffic rate is set to 6Mbps per card from the root and is divided evenly between all the nodes. 2) Bidirectional traffic where half the traffic from each node goes to the root and half from the root to the node. Each flow is a CBR flow with the maximum packet size of - 1460 bytes. The metric of interest is the total end to end throughput over all flows (root to node and node root) that a given channel allocation attains.

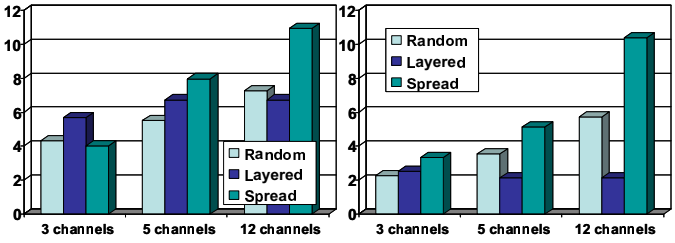


Fig. 6. Performance [Mbps] for "fish" topology. each card at the root can handle 6Mpbs (1.5Mbps per leaf). left: unidirectional, optimal=10.95Mbps right:bidirectional, optimal=10.38Mbps

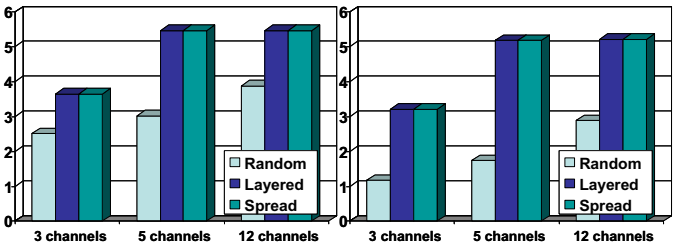


Fig. 7. Performance [Mbps] for string topology. Left: unidirectional Right:bidirectional. 4 channels are necessary for all hops to operate at maximum capacity.

bias fraction	unidir. / bidir.	channels used	optimal [Mbps]	difference [%]
0.2	B	3	6.37	37
0.2	B	5	6.37	12
0.2	U	5	6.59	16
0.5	B	3	8.07	32
0.5	B	5	8.07	30
0.5	U	5	8.25	18
0.8	B	3	9.72	10
0.8	B	5	9.72	37
0.8	U	5	9.93	29

TABLE I

Gain of the Spread method over the layered method when traffic offered is biased in the snowflake topology. Bias fraction is the ratio of traffic between the left subtree and the right subtree of the root. Difference is calculated as a percentage of the optimum attainable throughput in the given setting.

We use the two ray ground propagation model. In the ns2 simulator, a node interferes with a receiving node only if they are within carrier sense range of each other. Further, the capture threshold is set to 10db, implying that a signal received from an interfering node has to be within 10db of the receiving signal strength if the interfering node has to cause collision. Otherwise the interfering signal is considered too weak relatively and would be captured. The results for the "snowflake" topology are shown in Figure 5 and for the "fish" topology in Figure 6. We see that our algorithm is able to attain the optimal throughput in most cases while using only 5 channels. However, with 3 channels there are a couple of cases where the layered algorithm can outperform our scheme. The reason is that the layered algorithm will put the nodes at the same level (and 2 levels apart) in each others CS range. Thus, instead of being hurt by packets from an interfering node which we are not aware of, they properly share bandwidth. Our algorithm is hurt in this scenario because there are very few channels to spread and it assigns same channels to interfering nodes which are outside CS range, and this slightly degrades the performance. We note that we generated the Integer Programs corresponding to coloring the contention graph to figure out the number of colors required to color each contention graph, and the optimal numbers were 4 or 5 in all cases. Having optimal throughput with fewer channels is not a violation of this reasoning because the ILP does not have any notion of traffic or interference. It merely serves as an indicator of the number of channels required given the unweighted contention graph with only CS information.

The string topology is an important usecase since multihop wireless is often considered as an alternative transport network. Here the optimality is easily achieved by the layered method by simply using all channels in a round robin fashion. In Figure 7, we see that the Spread algorithm achieves this optimal performance by the virtue of its clause for reusing channels as wide apart as possible.

Another set of experiments shows the result of our algorithm when the traffic pattern is biased. The goal of the experiment is to see how well does our load awareness perform when compared to other schemes which are load unaware. In this setting, we reduced the total traffic demand of each node on that is in the subtree under one of the interfaces at the root (say the "left" card in both fish and the snowflake topology). We set the traffic demand of left sub tree to be 0.2, 0.5, and 0.8 times the demand of that of nodes in the right subtree. The gain of our method vis-a vis the layered algorithm is shown for a subset of values in Table I. Note that we are not showing any results for 12 channels since layered algorithm is unable to take advantage of 12 channels. These results are for snowflake topology and similar results hold for the fish topology as well. In general, we see that our algorithm has significant gains over other algorithms irrespective of the traffic bias, topology, or other different settings.

The results clearly show that with 5 or more channels our algorithm is always better than the other algorithms. With only 3 channels there are instances where our algorithm can underperform as layered algorithm tends to put competing nodes in CS range. With fewer channels when it is not possible to assign orthogonal channels to all or most of the competing nodes, having them in CS range serves better. However, due to its inherent nature, layered allocation does not take advantage of extra channels. This is evident as the throughput obtained using 5 or 12 channels is almost identical (since topology depth is at most 5) even though it is far away from optimal. This suggests that if large number of channels are available (as would be the case with 802.11a networks, or the newer whiteband networks), using layered allocation would under utilize the resources unless the topology is very deep. Secondly, with fewer channels, layered allocation may serve as a simpler alternative. However, its viability is dependent on the traffic pattern and the nature of the nodes (CS range and interference sensitivity) and needs to be understood for specific hardware.

V. RELATED WORK

Researchers have studied the benefits of using multiple channels and multiple radios in a wireless mesh network: authors in [3] proposed a new MAC layer to support multiple channels. Although a new MAC could utilize multiple channels more efficiently, but it would require modification to the existing 802.11 MAC, whereas our work is meant for use in the existing 802.11 MAC. Authors in [4], [3], [5] provide mechanisms for using multiple channels using a single interface. The protocol proposed in [4] does not require synchronization and can work with existing 802.11 MAC. Each node switches channels according to a pseudo random sequence, and it is guaranteed that the channels of any two nodes overlap periodically. However, switching might also

introduce delays. In this work, we consider multiple interfaces. [3] proposed the use of single interface to switch channels for load balancing. [5] proposed a protocol where each node with a packet to transmit has to switch the channel of the receiver before transmitting data. This would be costly even on present hardware, since channel switching introduces high per packet delay, and interference relations are highly complex. As shown in [6], [7], the channel allocation is NP-Complete, therefore an optimal solution is presently difficult to achieve. The authors in [6] also employ methods to control topology, which may not be achievable in our case, since the tree is already restrictive. Malone et al. [8] proposed a completely distributed scheme that is based on live load, and experimentally showed that it achieves considerable improvements. Our method is somewhat similar in spirit, since we also consider the actual load on each link.

References [9], [10], [11], [12] model the capacity of multiple channels and interfaces, understanding the benefits. In [10], authors analyze theoretically the scalability of capacity with respect to the number of number of channels, and number of available interfaces. [9] presents a capacity model for multiple channels, employing the feasibility of a rate matrix can be verified. Authors in [11] provide an ILP formulation for throughput optimization in mesh network. They study the impact of interfaces and channels on the overall throughput.

Contributions [13], [14] provide solutions to the joint channel assignment and routing problem. Authors of [15], proposed a method for centralized channel assignment to maximize the throughput. In [16], authors proposed an adaptive channel assignment based on the congestion on a given link. [17] propose a link layer solution for striping data over multiple interfaces. [18] proposed a metric WCETT, which is suitable for mesh network with multiple channels. The network considers the channel interference and bandwidth apart from ETX measure. [19] considers the problem of creating a survival topology in a multichannel scenario and proposes a bandwidth aware routing algorithm.

VI. CONCLUSIONS

We studied the problem of channel allocation for wireless mesh networks with a tree topology. Specifically, in our scenario, each mesh node has two interface cards – the upcard to communicate with the parent and the downcard to communicate with the children. All traffic is routed through the root node of the system. We designed a novel algorithm that utilizes the knowledge of the topology and the knowledge of the traffic pattern if available, to assign channels to all cards.

Our experiments show that the algorithm outperforms the alternate algorithms significantly. In some cases, the throughput gains are more than two fold. We also find that in several cases the algorithm attains the maximum throughput when using the required minimum number of channels which we calculated using graph coloring ILP for the contention graph. However, there are further problems to study in this area even in the specific case of tree topology. Foremost, getting an absolute interference model and incorporating its impact is an interesting problem. Furthermore, our work considers the case where the routes are given to us and we just assign channels. Given the traffic demands of the nodes, a possible direction would be to jointly assign channels and routes. This would provide additional load balancing capability and another knob to control the interference caused by channel sharing.

Acknowledgment: This work was supported in part by the CNCSIS grant PN2 Resurse Umane 11/01.07.2009 and by POSDRU/89/1.5/S/62557.

REFERENCES

[1] Dragoș Niculescu, Sudeept Bhatnagar, and Samrat Ganguly. Channel assignment for wireless meshes with tree topology. In *Proc. of International Conference on Communications (COMM 2010)*, pages 383–386, Bucharest, Romania, June 2010.

[2] Dragoș Niculescu. Interference map for 802.11 networks. In *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 339–350, New York, NY, USA, 2007. ACM.

[3] J. So and N. Vaidya. Multi channel MAC for ad hoc networks: handling multi channel hidden terminals using a single transceiver. In *ACM Mobihoc*, 2004.

[4] Paramvir Bahl, Ranveer Chandra, and John Dunagan. SSCH: slotted seeded channel hopping for capacity improvement in IEEE 802.11 ad-hoc wireless networks. In *MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking*, pages 216–230, New York, NY, USA, 2004. ACM.

[5] N. Shacham and P. King. Architectures and performance of multichannel multihop packet radio networks. In *IEEE Journal on Selected Area in Communications*, 1987.

[6] Mahesh K. Marina, Samir R. Das, and Anand Prabhu Subramanian. A topology control approach for utilizing multiple channels in multi-radio wireless mesh networks. *Computer Networks*, 54(2):241 – 256, 2010.

[7] Anand Prabhu Subramanian, Himanshu Gupta, Samir R. Das, and Jing Cao. Minimum interference channel assignment in multiradio wireless mesh networks. *IEEE Transactions on Mobile Computing*, 7:1459–1473, 2008.

[8] D. Malone, P. Clifford, D. Reid, and D.J. Leith. Experimental implementation of optimal WLAN channel selection without communication. In *2nd IEEE International Symposium, DySPAN 2007*, pages 316 –319, apr. 2007.

[9] M. Kodialam and T. Nandagopal. Characterizing the capacity region in multi radio multi channel wireless mesh networks. In *ACM Mobicom*, 2005.

[10] P. Kyasanur and N. Vaidya. Capacity of multi channel wireless networks: Impact of number of channels and interfaces. In *ACM Mobicom*, 2005.

[11] A.K. Das, H.M.K. Alazemi, R. Vijayakumar, and S. Roy. Optimization models for fixed channel assignment in wireless mesh networks with multiple radios. In *Sensor and Ad Hoc Communications and Networks, 2005. IEEE SECON 2005. 2005 Second Annual IEEE Communications Society Conference on*, pages 463–474, Sept., 2005.

[12] W. Wang and X. Liu. A framework for maximum capacity in multi channel multi radio wireless networks. In *Proceedings of CCNC*, 2006.

[13] 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.

[14] P. Kyasanur and N. Vaidya. Routing and interface assignment in multi channel multi interface wireless networks. In *WCNC*, 2005.

[15] Ashish Raniwala, Kartik Gopalan, and Tzi-cker Chiueh. Centralized channel assignment and routing algorithms for multi-channel wireless mesh networks. In *ACM Mobile Computing and Communications Review(MC2R)*, volume 8, April 2004.

[16] A. Raniwala and T. Chiueh. Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network. In *IEEE Infocom*, 2005.

[17] A. Adya, P. Bahl, J. Padhye, A. Wolman, and L. Zhou. A multi radio unification protocol for IEEE 802.11 wireless networks. In *Microsoft Research TR*, volume 44, 2003.

[18] Richard Draves, Jitendra Padhye, and Brian Zill. Routing in multi-hop multi-radio wireless mesh networks. In *ACM MobiCom*, Philadelphia, PA, September 2004.

[19] J. Tang, G. Xue, and W. Zhang. Interference aware topology control and QoS routing in multichannel wireless mesh net works. In *ACM Mobihoc*, 2005.