2010 24th IEEE International Conference on Advanced Information Networking and Applications

A Novel Spatial TDMA Scheduler for Concurrent Transmit/Receive Wireless Mesh Networks

Kwan-Wu Chin
University of Wollongong
Northfields Ave,
Wollongong, NSW, Australia
kwanwu@uow.edu.au

Sieteng Soh
Dept. of Computing,
Curtin University of Technology,
Perth, WA, Australia
S.Soh@curtin.edu.au

Chen Meng
University of Wollongong
Northfields Ave,
Wollongong, NSW, Australia
cm488@uow.edu.au

***Abstract*—The success of wireless mesh networks hinges on their ability to support bandwidth intensive, multi-media applications. A key approach to increasing network capacity is to equip wireless routers with smart antennas. These routers, therefore, are capable of focusing their transmission on specific neighbors whilst causing little interference to other nodes. This, however, assumes there is a link scheduling algorithm that activates links in a way that maximizes network capacity. To this end, we propose a novel link activation algorithm that maximally creates a bipartite graph, which is then used to derive the link activation schedule of each router. We have verified the proposed algorithm on various topologies with increasing node degrees as well as node numbers. From extensive simulation studies, we find that our algorithm outperforms existing algorithms in terms of the number of links activated per slot, superframe length, computation time, route length and end-to-end delay.**

***Index Terms*—Wireless Mesh Networks, Scheduling, Concurrent Transmit/Receive**

I. Introduction

Wireless mesh networks (WMNs) consist of an ad-hoc collection of routers that have the ability to self-organize and self-configure. Specifically, they are able to dynamically establish mesh connectivities to nearby routers to form a wireless backbone capable of carrying traffic to/from clients over multiple hops. Moreover, given their low cost, good reliability and easy maintenance, they have found many applications. For example, in [15], Raman et al. implemented a concurrent transmit/receive WMN using off the shelf IEEE 802.11 hardware. Each access point (AP) is equipped with multiple radios connected to a high gain, directional antenna. The resulting network is then used to interconnect rural villages in India. Other applications of WMNs include traffic control [11], and smart grids [18].

A fundamental problem in WMNs is the lack of capacity. Specifically, Gupta et al. [10] showed that for a given node density n, the total end-to-end capacity is approximately $O(\frac{n}{\sqrt{n}})$, and the available throughput of each node is $O(\frac{1}{\sqrt{n}})$. This means as the number of nodes increases, the throughput of each node decreases to zero. These theoretical results have also been verified via simulation and experimental studies. For example, Das et al. [7] and Li et al. [12] have shown that the throughput achieved by nodes are in the order of kilobits per second, despite having a transceiver capable of transmitting several megabits per second.

To this end, researchers have devised various solutions to increase the capacity of WMNs. One of which is to equip routers with smart or directional antennas [15][17][2][9][13]. As a result, routers have the ability to focus their transmission energy or electromagnetic beam on a given geographical region. In theory, a router equipped with a k-element array is able to provide k spatial channels or null out $k-1$ interfering nodes [6]. The former allows one-to-many connections, whereas the latter enables neighboring devices to transmit simultaneously; as opposed to being blocked when routers use an omni-directional antenna. Moreover, the nulling and beam steering capabilities of such antennas mean that a router has a longer range and causes little interference, thereby, improving gains and lowering bit error rates during packet reception.

Figure 1 illustrates the key advantage of smart antennas. Assume nodes use time division multiple access (TDMA) to access the wireless channel. We first consider the case in which each router is equipped with an omni-directional antenna. The sets of links that do not interfere with one another, and thus can transmit concurrently in a given time slot include $\{L_1, L_{10}\}$, $\{L_4, L_9\}$ and $\{L_4, L_7\}$. On the other hand, with smart antennas, router N_c, N_e and N_f can transmit on links $\{L_1, L_3, L_6\ L_7, L_9\}$ simultaneously in a given time slot. These routers then prepare to receive in the next time slot; i.e., by activating links $\{L_2, L_4, L_5, L_8, L_1\}$. Therefore, a WMN that uses omni-directional and smart antennas is able to concurrently activate two and five links respectively. In other words, smart antennas increased the capacity of the network by 250%.

Henceforth, any proposed scheduler or Medium Access Control (MAC) protocol must maximize the number of links scheduled within each time slot. This problem, however, is NP-hard as link scheduling amounts to computing the chromatic number of a graph. For example, for Figure 1 or a tree topology, only two slots or colors are required to ensure all links receive at least one transmission and reception opportunity. A possible approach to this problem is to ensure all routers self-organize into a tree topology [15]. Unfortunately, this is often impractical as WMN operators need to ensure adequate coverage, and may be constrained by a given environment. Moreover, a tree topology has poor fault tolerance, is not ideal for traffic headed to other parts of a tree, and parent nodes are often bottlenecks. Hence, it is critical that we develop an

1550-445X/10 $26.00 © 2010 IEEE
DOI 10.1109/AINA.2010.13

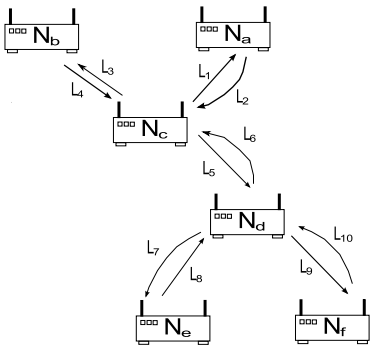


Fig. 1. An example wireless mesh network.

efficient scheduler for arbitrary WMN topologies.

To this end, this paper proposes a novel algorithm that recursively forms a bipartite graph to derive a schedule that maximizes the number of links activated in a given slot for a given topology/graph. Our algorithm has a complexity of $O(|V|^2)$, where V is the set of nodes/routers. From our extensive simulation studies involving topologies with increasing node degree and node numbers, we find that our algorithm is able to generate shorter superframe lengths when compared to existing approaches. As a result, nodes experience lower end-to-end path delays as packets are able to traverse a network quicker. Moreover, unlike the approach presented in [3], our algorithm does not result in bottlenecks or elongated paths, and is several orders of magnitude faster.

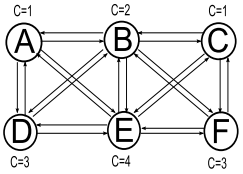
This paper is structured as follows. We first formulate the problem in Section II, followed by a review of existing solutions in Section III. In Section IV, we present our algorithms, and show how they derive the schedule of an example topology. After that, we list some of its key properties in Section V. Section VI outlines the simulation methodology used to obtain the results presented in Section VII. Finally, our conclusions are presented in Section VIII.

II. The Problem

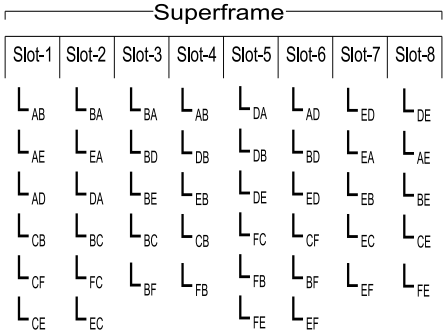
Let G(V,E) be a directed graph that represents a WMN. Let $v_i \in V$ represents node i that is equipped with a smart antenna capable of forming up to $b_i \geq 1$ beams, thereby allowing v_i to simultaneously transmit or receive from its b_i neighbors. Let N_i denote a set containing v_i's neighbors. Let $e_{ij} \in E$ denotes a directional link from node i to j. We assume each v_i has the same in- and out-degree. A node is not allowed to transmit on one or more of its links whilst receiving packets from other links, and vice-versa.

Figure 2(a) shows an example WMN, aka "2boxes", with six nodes, each capable of forming up to $b_i = 5$ links to transmit or receive simultaneously. Notice that each transmission slot is followed by a reception slot. This allows transmissions to be acknowledged promptly, and more importantly, enables WMN operators to employ 2P [15] – a concurrent transmit/receive MAC explained in Section III. The problem is, therefore, to maximize the number of links that are simultaneously transmitting or receiving in each time slot, or alternatively, minimize the superframe length, such that each link is activated at least once. Here, we assume only one packet can be transmitted or received in each time slot.

One solution is to apply a graph coloring algorithm. For the 2boxes topology, the number of colors required or its chromatic number is four; as indicated by the label $C \in \{1, 2, 3, 4\}$. Each color then corresponds to two slots: transmit (TX) and receive (RX). Hence, setting the superframe to 4×2 slots ensures all links are activated at least once in a superframe; see Figure 2(b). For example, in slot-1, node A and C transmit to their respective neighbors synchronously, and in slot-2, they become receiver. As we shall see, the optimal schedule for the 2boxes topology requires only four slots.



(a)



(b)

Fig. 2. (a) "2boxes" topology, (b) Example schedule.

In the aforementioned problem, we assume v_i has b_i neighbors. However, in practice, a node may not have sufficient antenna elements to establish $|N_i|$ links. One example is when some antenna elements are used to null interference. To this end, in Section IV, we show how our algorithm can be modified easily to consider $b_i < |N_i|$.

III. Related Works

To date, very little works have considered the problem of scheduling links in concurrent transmit/receive WMNs. In fact, only [3] and [15] are directly relevant to our work. In [3], Chin presented an algorithm that sacrifices path length for higher network capacity. Specifically, the algorithm disables links in cliques, and causes nodes that were part of a clique to route their packets via an intermediate node. Despite the slight increase in path length, nodes observe lower end-to-end delays.

This is because removing cliques from a topology increases a WMN's network capacity or reduces its chromatic number. Hence, packets are able to traverse a WMN much quicker. Unfortunately, Chin's algorithm creates bottleneck links, and is not ideal for local traffic; i.e., those between nodes that were part of a clique.

As we mentioned earlier, Raman et al. [15] implemented a WMN to interconnect rural villages. Their WMN runs a MAC called 2P, which requires APs to either be in transmit or receive mode – as defined by the problem definition in Section II. This constraint means 2P requires a bipartite topology. To this end, Raman et al. propose an algorithm to construct a tree topology that meets the following criteria: transmission range, angle between neighbors, and hop count or tree depth. In contrast, we present an algorithm that derives bipartite graphs from any topologies, which can then be scheduled using 2P. Thereby, giving network operators more flexibility when deploying a WMN capable of concurrent transmit/receive. In other words, routers need not be organized into a tree.

Another approach to building a concurrent transmit/receive WMN is to equip routers with digital adaptive arrays (DAAs)– commonly referred to as Multiple Input and Multiple Output (MIMO) [6][17] or smart antennas. Recall that these routers have the ability to form up to k independent channels in order to obtain spatial multiplexing gain. Here k refers to the number of antenna elements. Hence, in a concurrent transmit/receive WMN, these routers can transmit or receive to/from k neighbors simultaneously. A router can also use a subset of its k antennas to null out interfering transmissions, or use these antennas to transmit the same data to improve the signal-to-noise ratio of its communication.

To date, research into using smart antennas in wireless ad-hoc networks has focused on maximizing the number of packet transmissions whilst minimizing the number of antennas used for nulling. This is a marked departure from works that use omni-directional antenna, as routers are now capable of transmitting or receiving even when there are interfering routers. For example, Sundaresan et al. [17] exploit the nulling capability of smart antennas to maximize the number of concurrent transmissions in a given time. The authors outline an algorithm that schedules transmissions according to the number of cliques in a contention graph a node belongs to, and show how nodes are able to maximize their k antennas for packet transmissions as opposed to using them to null interfering neighbors. The problem considered in existing research, however, is different from ours because they do not consider a node transmitting or receiving from multiple neighboring nodes concurrently. In other words, existing works aim to maximize the number of communicating node pairs whereas we try to maximize the number of point to multi-point, and vice-versa, connections.

Spatial TDMA is a well studied research area. In 1985, Nelson et al. [14] first defined the problem and provided a solution. Since then, many researchers have developed centralized [4] and distributed algorithms [8][5] that maximize spatial reuse. In other words, they sought to increase the number of transmitting links in a given slot; see [1] and references therein for more information. These early works, however, consider nodes with omni-directional antenna. Hence, the resulting WMN is interference limited. To this end, researchers have considered directional antennas. For example, Bao et al. [2] present a distributed algorithm that makes use of two hops neighbor information to activate links according to their priority. When a link is activated for transmission, the algorithm determines whether there are other links in its two hops range that can be activated simultaneously. Another example is [9], where the authors aim to maximize the number of node pairs or non interfering links at a given point in time. Fundamentally, these link scheduling works differ from ours in the following manner. First, they assume nodes are only able to transmit or receive from a given neighbor at any point in time. In other words, each node only has a single link. We, on the other hand, consider a WMN where nodes are capable of transmitting or receiving to/from all of its neighbor synchronously. Secondly, nodes in our WMNs are able to selectively transmit or receive from a given set of neighbors. These fundamental differences, therefore, preclude the use of existing "single-link" scheduling algorithms.

IV. Proposed Algorithms

We first propose an algorithm, called Algo-1, that considers the case when v_i have sufficient beams, i.e., $b_i \geq |N_i|$, to establish a link to all of its neighbors before showing how Algo-1 is revised to address the case when $b_i < |N_i|$.

A. Case: $b_i \geq |N_i|$

Algo-1 recursively partitions a graph V into maximally connected bipartite graphs. Note that, in practice, given that nodes in WMNs are generally static, we expect Algo-1 to be run infrequently. Algo-1 partitions nodes into two disjoint sets, i.e., Set1 and Set2, such that these nodes are maximally connected. The nodes in Set1 are then scheduled to transmit in slot i, and conversely, to receive in slot $i+1$. Algo-1 is then applied to nodes in each set to derive the next schedule; i.e., active links in slot $i+2$ and $i+3$, and so forth.

The function $GSelect()$ plays a crucial role during the construction of Set1 and Set2 to greedily produce bipartite graph with maximal edges. Specifically, GSelect(v_i) is a greedy function that returns true if it can select a node $v_i \in V$ with the largest $(conn2(v_i) - conn1(v_i)) > 0$ value. It returns false otherwise. In other words, $GSelect()$ greedily selects a node with the maximum number of edges that have yet to be considered in the bipartite graph.

We will now show how Algo-1 determines the schedule for the 2boxes topology shown in Figure 2. First, we derive the links to be activated in slot 1 and 2 followed by those in slot 3 and 4.

- *Line 1 – 5*: Initially, $V = \{A, B, C, D, E, F\}$, and $Set1 = Set2 = \{\}$. For ease of exposition, we represent the value of *conn1* and *conn2* for each node v_i as $v_i(conn1, conn2)$. Therefore, we have, A(0, 3), B(0,5),

```
input : G(V, E) represented as an adjacency list
output: Schedule represented as (Set1, Set2)
1  if |V| ≤ 1 return;
   /* The following represents the two
      disjoint sets of a bipartite graph.
      */
2  Set1 = {};
3  Set2 = {};
   /* conn1(v_i) refers to the total
      number of nodes in Set1 that are
      connected to v_i                    */
4  conn1(v_i) = 0;
   /* conn2(v_i) denotes the total number
      of nodes in V that are connected to
      v_i. Recall that id(v_i) denotes the
      in-degree of node v_i in V.         */
5  conn2(v_i) = id(v_i);
6  while (V ≠ {} AND GSelect(v_i ∈ V) == TRUE) do
7      Set1 = Set1 + {v_i};
8      Set2 = Set2 − {v_i};
9      foreach v_j ∈ V AND v_j ≠ v_i do
           /* If v_i and v_j are connected   */
10         if EDGE(v_i, v_j) then
11             Set2 = Set2 + {v_j};
12             conn1(v_j)++;
13             conn2(v_j)--;
14             if conn2(v_j) == 0 then
15                 V = V − {v_j};
16             end
17         end
18     end
19     V = V − {v_i};
20 end
   /* Recursively call Algo-1 again on
      each set.                           */
21 Algo-1(Set1);
22 Algo-1(Set2);
```

Algorithm 1: Proposed algorithm. The output is a schedule where all nodes in Set1 transmit to those in Set2, and vice-versa, in time slot i and $i+1$ respectively.

C(0,3), D(0,3), E(0,5), F(0,3). For example, node A currently has three links connected to it in V.

- *Line 6 – 9*: The function GSelect() then selects a node with the biggest (conn2 - conn1) value. In this case, there are two nodes: B and E. In this case, Algo-1 simply selects node B over E arbitrarily. Node B is then added to Set1.
- *Line 10 – 17*: Algo-1 then visits each neighbor (child) of node B and adds them to Set2. Moreover, it updates *conn1* to reflect an additional connection from each child to node B, who is now in Set1. Also, *conn2* is decremented by one for a given child as it now has one less connection in V.
- *Line 19*: Node B is then removed from V.

At this stage, $Set1 = \{B\}$ and $Set2 = \{A, C, D, E, F\}$, and the value of *conn1* and *conn2* for each node is A(1,2), C(1,2), D(1,2), E(1,4) and F(1,2). Lastly, $V = \{A, C, D, E, F\}$. Algo-1 then repeats lines 6 – 9 as follows:

- *Line 6 – 9*: GSelect() picks node E because $(conn2(E) - conn1(E))$ has the largest value; i.e., 3. Node E is then added to Set1 and removed from Set2 as per line 7 and 8 respectively.
- *Line 10 – 17*: Algo-1 then visits nodes in V that are adjacent to node E, and adjusts their *conn1* and *conn2* accordingly.
- *Line 19*: Algo-1 then removes node E from V.

At this point, $Set1 = \{B, E\}, Set2 = \{A, C, D, F\}$, and $conn1$ and $conn2$ have the following value for each node: A(2,1), C(2,1), D(2,1), F(2,1). Lastly, $V = \{A, C, D, F\}$. Notice that the $(conn2(v_i) - conn1(v_i))$ value for all remaining nodes v_i in V is -1. Hence, the while loop encompassing line 6 – 20 finishes. As a result, we have a bipartite graph represented by $Set1 = \{B, E\}, Set2 = \{A, C, D, F\}$, and also the links to be scheduled in slot 1 and 2; see Figure 3.

To determine the links for slot 3 and 4, we apply Algo-1 on $Set1 = \{B, E\}$ and $Set2 = \{A, C, D, F\}$; line 21 and 22. Notice that both sets are independent. Thus, the links chosen for both sets can be activated simultaneously.

We first show the steps carried out by Algo-1 on $Set1 = \{B, E\}$.

- *Line 1 – 5*: $V = \{B, E\}, Set1 = Set2 = \{\}$ and B(0,1), E(0,1).
- *Line 6 – 9*: GSelect() selects $v_i = B$ arbitrarily given that both node B and E have the same $(conn2 - conn1)$ value.
- *Line 10 – 16*: Algo-1 adds node E to Set2, and updates *conn1(E)* and *conn2(E)* accordingly.

At this stage, $Set1 = \{B\}, Set2 = \{E\}$ and $V = \{\}$. Therefore, the first link to be activated in slot 3 is e_{BE}. Algo-1 is then recursively applied on $Set1 = \{B\}$ and $Set2 = \{E\}$ to discover links to be activated in slot 5 and 6. In both cases, Algo-1 returns immediately as $|V| \leq 1$. Hence, there are no links to be activated in slot 5 and 6.

Next, Algo-1 operates on $Set2 = \{A, C, D, F\}$, where it will look for links to be activated in slot 3 and 4. Repeating lines 1 to 16 on this set yield the links e_{AD}, e_{CF}. As before, Algo-1 is applied recursively to the sets $Set1 = \{A\}$, $Set2 = \{D\}$, $Set1 = \{C\}$, and $Set2 = \{F\}$. All of which do not produce any additional links. Hence, there are no links to be activated in slot 5 and 6. Therefore, the 2boxes topology only requires four slots. Figure 3 shows the links that are activated in slot 1 and 4.

B. Case: $b_i < |N_i|$

To address this case, we remove line 21 and 22 from Algo-1. In other words, it is no longer applied recursively to Set1 and Set2. From hereon, we denote the revised algorithm as Algo-1b.

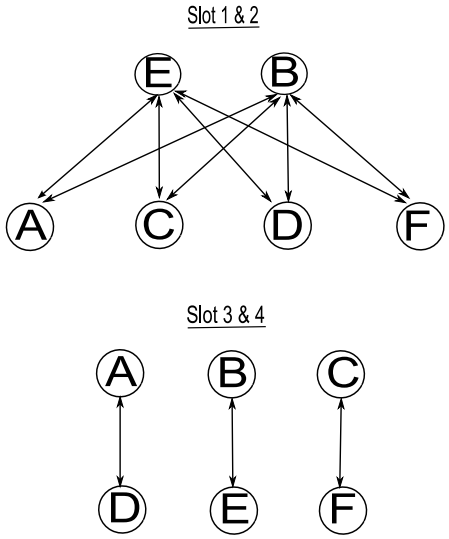


Fig. 3. Schedule for the 2boxes topology.

The links activated in each slot s_i for a graph G(V,E) are then computed according to the following steps:

1) s_i = {}.
2) (Set1, Set2) = Algo-1b(G(V,E)).
3) Select up to b_i links from Set1 and Set2, and add them to s_i. The selection of node in Set2 is such that the remaining degree of the node is as maximal as possible.
4) Remove all b_i links in s_i from G(V,E).
5) Exit if $|E|$ is zero. Otherwise, go back to Step-1 for links in slot s_{i+2}.

We now illustrate how the above steps determine the links in each slot for the 2boxes topology when $b_i = 2$.

1) *Slot 0*. Running Algo-1 yields $Set1 = \{B, E\}$ and $Set2 = \{A, C, D, F\}$. For each node in Set1, select up to b_i links to nodes in Set2. We arbitrary select nodes in Set2 given that all nodes in Set2 have the same degree. Without loss of generality, assume that the links to be activated in s_0 are $\{e_{BA}, e_{BC}, e_{ED}, e_{EF}\}$. Hence, in s_1 we have $\{e_{AB}, e_{CB}, e_{DE}, e_{FE}\}$. These links are then removed from G(V,E).
2) *Slot 2*. Applying Algo-1 on G(V,E) yields $Set1 = \{B, E\}$ and $Set2 = \{A, C, D, F\}$ again. As before, we seek to establish up to b_i links from node B and E respectively to nodes in Set2. So we have $s_2 = \{e_{BD}, e_{BF}, e_{EA}, e_{EC}\}$, and $s_3 = \{e_{DB}, e_{FB}, e_{AE}, e_{CE}\}$. As before, these links are then removed from G(V,E).
3) *Slot 4*. At this point, only the following links remain in G(V,E): e_{AD}, e_{BE}, and e_{CF}. Given that G(V,E) is bipartite and the number of links to be activated is less than b_i, we have $s_4 = \{e_{AD}, e_{BE}, e_{CF}\}$, and $s_5 = \{e_{DA}, e_{EB}, e_{FC}\}$.

Therefore, six slots are sufficient when $b_i = 2$. We can repeat the example above to find $b_i = 3$ and $b_i = 4$ only require four slots.

V. Analysis

We will start by comparing the schedule generated by Algo-1 and graph coloring (GC), see Figure, 2(a), for the 2boxes topology. We see that GC requires eight slots, and Algo-1 only needs four slots. Moreover, from Figure 2(b), we see that GC schedules 44 links in eight slots, i.e., an average of 5.5 links per slot. On the other hand, in Figure 3, Algo-1 only manages to activate 22 links in four slots. Notice, however, in slot 3, node A can also activate its link to node E. Similarly, node B and C could activate their link to node D and E respectively. Then in slot 4, the following links are activated: e_{EA}, e_{DB}, e_{EC}, and e_{FB}. Thus, in slots 3 and 4, Algo-1 could schedule eight additional links, for a total of $6 + 8 = 14$ links. In other words, Algo-1 can schedule up to 30 links in four slots, or on average, 7.5 links per slot, which is much better than GC.

We now outline several properties of Algo-1.

Theorem V.1. *Algo-1 produces a schedule with time frame $\lceil log_2 n \rceil$ for a fully connected graph with n nodes.*

Proof: Without loss of generality, consider n is a power of 2. At time frame 1, Algo-1 produces $|Set1| = |Set2| = \frac{n}{2}$, each containing $\frac{n}{2}$ nodes. Notice that Set1 and Set2 are each a fully connected graph with $\frac{n}{2}$ nodes. The algorithm will repeatedly produce Set1 and Set2 for the subsequent time frame, until $|Set1| = |Set2| = 1$. It is obvious that the algorithm repeats the step $log_2 n$ times. ■

Corollary V.2. *Algo-1 produces a schedule with time frame at most $log_2 n$ for a graph with n nodes.*

Proof: We first consider the graph as a fully connected graph. From Theorem V.1, Algo-1 will produce a schedule with time frame $log_2 n$. ■

We compute the time complexity of Algo-1 for the two possible constraints: $b_i \geq |N_i|$ and $b_i < |N_i|$. Each line in Algo-1 takes O(1), except for lines 6 and 9. The $Gselect()$ function takes $O(|V|)$ times, and line 9 considers $|V|$ possible nodes. Therefore, for constraint $b_i \geq |N_i|$, the time complexity of Algo-1 can be expressed by a recurrent function $T(|V|) = T(|Set1|) + T(|V| - |Set1|) + O(|V|)$. In the worst case, $|Set1| = 1$, which gives $T(|V|) = T(1) + T(|V| - 1) + O(|V|)$. Thus, Algo-1 has time complexity $O(|V|^2)$ in the worst case for constraint $b_i \geq |Ni|$. For constraint $b_i < |Ni|$, function Algo-1b has time complexity $O(|V|)$. In the worst case, the function is repeated $|E|$ times, and thus Algo-1 has time complexity $O(|E||V|)$ for this constraint.

VI. Simulation

We used MatGraph [16], a Matlab toolkit for working with simple graphs, to verify the performance of Algo-1. The toolkit comes with five graph coloring algorithms. We, however, only employ the "optimal" algorithm as it can reliably determine the chromatic number or schedule of a graph. We compare Algo-1 with two existing schemes: the clique removal algorithm proposed by Chin [3], hereafter referred to as "Link Relaxation", and graph coloring (see Figure 2).

Our experiments involve graphs with varying node degree and node density. In the former, we vary the degree of 20 nodes from two to nine, whereas in the latter, the number of nodes ranges from 10 to 100. For each graph, we record the resulting chromatic number before and after applying Algo-1. Note that our implementation also employs the opportunistic link activation extension described in Section V. This means in each iteration, Algo-1 maximally activates links that may or may not have been activated in earlier iterations, provided that the resulting graph remains bipartite.

For each experiment, after deriving the schedule of a given topology, we computed the number of links activated, the superframe length and the end-to-end packet delay. To compute the packet delay, we set each node to transmit one packet to another random node and record the delay incurred by the packet. Hence, in a 100 nodes network, there will be 100 packets. Note, the speed in which packets traverse the shortest path for a given node pair is determined by the superframe length, and the time in which packets have to wait at each node before their outgoing link becomes active. We also calculate the shortest end-to-end path length for a given source and destination node. Finally, we measure the computation time of each algorithm using Matlab's stopwatch timer.

VII. Results

We first investigate the impact of node degree before focusing our attention on node density.

A. Node Degree

1) Average Links Activated: Figure 4 shows the average number of links activated in each slot for Algo-1 and graph coloring with increasing node degree. We can see that Algo-1 is particularly advantageous when nodes have a high degree. For example, when nodes' degree exceeds seven, Algo-1 manages to activate 28% to 30% more links than graph coloring. This is because with increasing links, Algo-1 has more opportunities to activate previously scheduled links. Note, we did not consider Link Relaxation because links are intentionally disabled, meaning its performance with respect to the number of links activated is inevitably much poorer than graph coloring and Algo-1.

2) Superframe Length: Figure 5 shows the superframe length for each algorithm. We see that Algo-1 yields shorter superframes as compared to all other algorithms, especially when nodes have a high degree. In particular, Algo-1 has shorter superframe lengths than Link-Relaxation. This means network operators do not have to intentionally disable links to increase capacity. In other words, it is not necessary to trade-off path length for capacity. Moreover, as shown in Figure 6, packets do not need to traverse an artificially elongated path. Note that Algo-1 has the same path length as graph coloring because the same set of links is used to compute the shortest path between nodes.

3) Average Path Delay: Figure 7 shows the average path delay for each algorithm. The key reasons for the lower delay when Algo-1 is used are shorter superframe and path length. Although Algo-1 and graph coloring result in similar path lengths, they produce different superframe lengths, which is critical in ensuring packets traverse their respective path quickly. This is the key reason that causes graph coloring to have a high end-to-end delays, see Figure 7, as packets have to wait longer at each node before their respective out-going link becomes active.

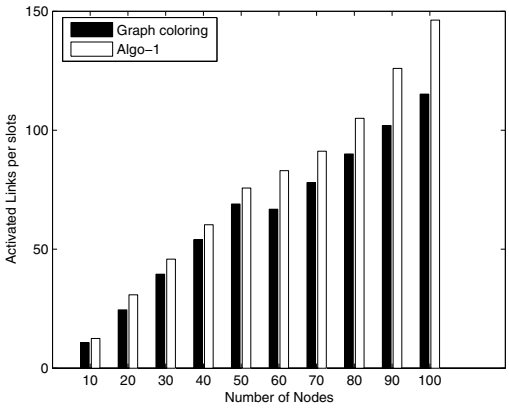


Fig. 4. The average number of links activated in each slot.

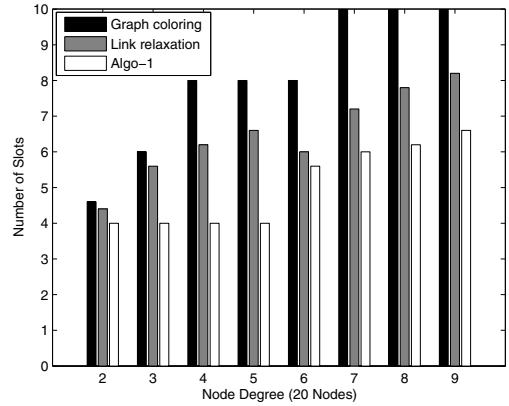


Fig. 5. Superframe length.

B. Node Density

1) Average Links Activated: Figure 8 shows the number of links activated with increasing node numbers. Again, Algo-1 outperforms graph coloring because it maximally activates links in each time slot. Hence, with increasing node counts, the chances of finding links to schedule becomes higher. This is evident from Figure 8. For example, when there are 90 to 100 nodes, Algo-1 managed to find 30% more links to activate.

2) Superframe Length: Figure 9 shows that in general Algo-1 and Link-Relaxation have comparable superframe length. This is because Link-Relaxation is a clique removal algorithm. Hence, it is particularly beneficial when a WMN has many cliques – as we demonstrated in Section VII-A.

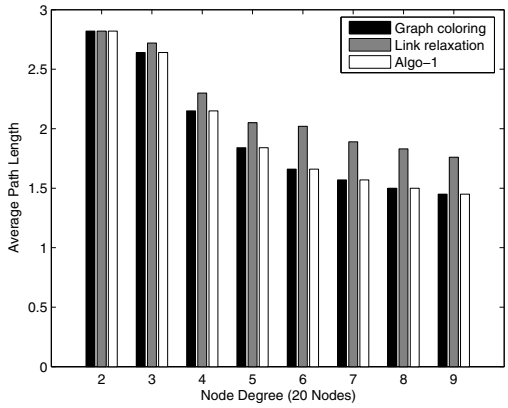


Fig. 6. End-to-end path length.

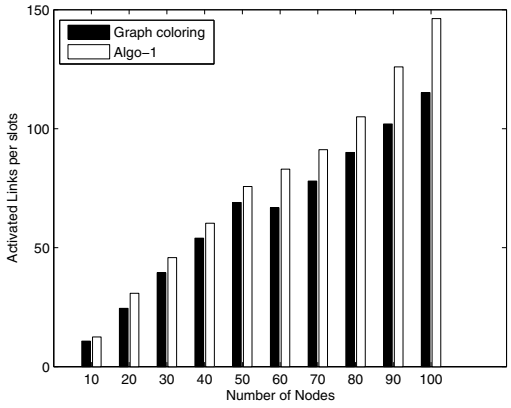


Fig. 8. The average number of links activated in each slot.

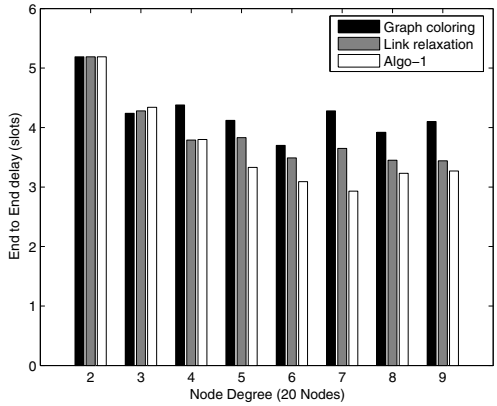


Fig. 7. Average end-to-end delay.

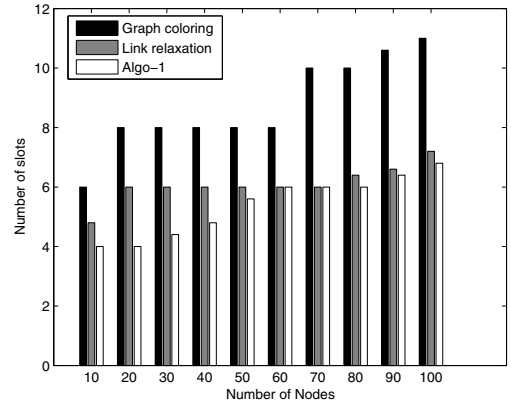


Fig. 9. Superframe length.

However, in this experiment, there are not many cliques. As result, Algo-1 has no significant advantage over Link-Relaxation in terms of superframe length. Nevertheless, Algo-1 manages to yield similar or shorter superframe lengths for most tested topologies. This is especially true if a WMN has many odd cycles. For example, an odd cycle with five nodes requires six slots if we use Link-Relaxation but requires only four slots using Algo-1.

3) Average Path Delay: Figure 10 shows the end-to-end delay incurred by packets for all three algorithms. Both Algo-1 and Link-Relaxation perform better than graph coloring as they have much shorter superframe lengths; see Figure 9. As a result, packets are able to traverse their respective path quicker.

C. Computation Time

Lastly, we measure the computation time taken by Algo-1 and Link-Relaxation on a laptop with a 2.2 gigahertz Intel processor, and one gigabyte of memory. Note, we do not measure the time it takes to color a graph. This allows us to only compare the key parts of Algo-1 and Link-Relaxation as opposed to Matgraph's graph coloring algorithm. From Figure 12, we see that Algo-1 has orders of magnitude faster computation time as compared to Link-Relaxation. This is because Link-Relaxation needs to visit each node and its children to ascertain whether they form a clique. Unfortunately, this operation becomes computationally very expensive with increasing node numbers.

VIII. Conclusion

We have outlined a simple link scheduling algorithm that outperforms existing schedulers. Unlike prior approaches, our algorithm does not need to trade-off path length in order to increase capacity. In addition, we do not require the topology to be bipartite. This is advantageous to network operators as it provides much needed flexibility when deploying wireless routers.

A key future work is to develop a joint routing and link scheduling approach to take advantage of links that are activated more often than others. Apart from that, we intend to develop a scheduler that is cognizant of traffic load. In other words, activate links according to their load. Hence, some

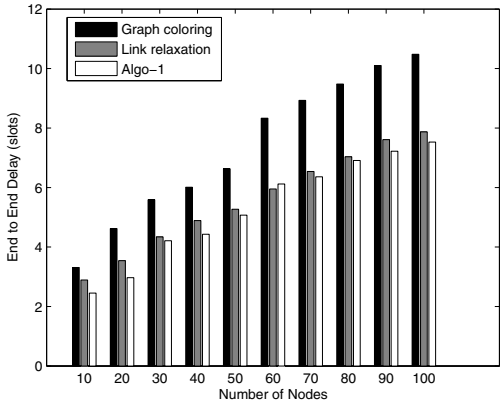


Fig. 10. Average end-to-end delay.

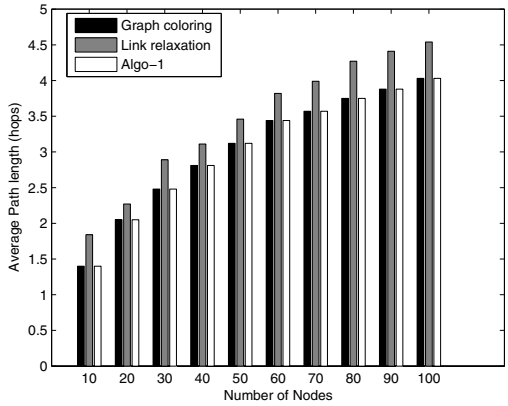


Fig. 11. End-to-end path length.

links will be given multiple opportunities to transmit in each superframe.

REFERENCES

[1] L. Bao and J. Garcia-Luna-Acceves. *Algorithms and Protocols for Wireless and Mobile Networks*, chapter 4: Distributed Channel Access Scheduling for Ad Hoc Networks. Chapman and Hall, CRC, 2004.

[2] L. Bao and J. Garcia-Luna-Aceves. Transmission scheduling in ad hoc networks with directional antennas. In *Proceedings of the 7th ACM/IEEE Internation Conference on Mobile Computing and Networking (MOBICOM'2001)*, Atlanta, USA, Sept. 2002.

[3] K.-W. Chin. A new link scheduling algorithm for concurrent Tx/Rx wireless mesh networks. In *IEEE ICC 2008*, Beijing, China, May 2008.

[4] I. Chlamtac and A. Farago. Making transmission schedules immune to topology changes in multi-hop packet radio networks. *IEEE/ACM Transactions on Networking*, 2(1):23–33, Feb. 1994.

[5] I. Cidon and M. Sidi. Distributed assignment algorithms for multihop packet radio networks. *IEEE/ACM Transactions on Networking*, 38(10):1353–1361, Oct. 1989.

[6] M. Cooper and M. Goldburg. Intelligent antennas: Spatial division multiple access. Annual Review of Communication, 1996.

[7] S. R. Das, C. Perkins, and E. M. Royer. Performance comparison of two on-demand routing protocols for ad-hoc networks. In *Proceedings of IEEE INFOCOM'2000*, Tel-Aviv, Israel, 2000.

[8] A. Ephremedis and T. Truong. Distributed algorithm for efficient and interfere-free broadcasting radio networks. In *IEEE INFOCOM'1988*, San Francisco, CA, USA, 1988.

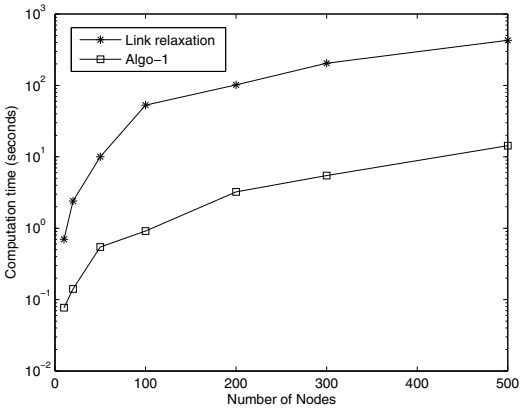


Fig. 12. Computation.

[9] J. Groönkvist. Assigning strategies for spatial reuse TDMA. Master Thesis, Mar. 2002. Sweden Royal Institute of Technology, TRITA-S3-RST-0202, ISSN 1400-9137.

[10] P. Gupta and P. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, Mar. 2000.

[11] K.-C. Lan, Z. Wang, M. Hassan, T. Moors, R. Berriman, and L. Libman. Experiences in deploying a wireless mesh network testbed for traffic control. *ACM SIGCOMM Computer Communications Review*, 37(5), Oct. 2007.

[12] J. Li, C. Blake, D. S. J. D. Couto, H. I. Lee, and R. Morris. Capacity of ad hoc wireless networks. In *ACM MOBICOM 2001*, Rome, Italy, July 2001.

[13] X. Liu, A. Sheth, M. Kaminsky, K. Papagiannaki, S. Seshan, and P. Steenkiste. DIRC: Increasing indoor wireless capacity using directional antennas. In *ACM SIGCOMM'09*, Barcelona, Spain, Aug. 2009.

[14] R. Nelson and L. Kleinrock. Spatial TDMA: A collision-free multihop channel access protocol. *IEEE Transactions on Communications*, COM-33(9):934–945, Sept. 1985.

[15] B. Raman and K. Chebrolu. Design and evaluation of a new MAC for long-distance 802.11 mesh networks. In *The 11th Intl. Conference on Mobile Computing and Networking (MOBICOM)*, Cologne, Germany, Aug. 2005.

[16] E. R. Scheinerman. Matgraph 1.7. http://www.ams.jhu.edu/ ers/matgraph.

[17] K. Sundaresan, R. Sivakumar, M. A. Ingram, and T.-Y. Chang. A fair MAC protocol for ad-hoc networks with MIMO links. In *IEEE INFOCOM*, Hong Kong, China, 2004.

[18] Tropos Inc. Tropos - helping communities build america's smart grid. White Paper, Jan. 2009. http://www.tropos.com/pdf/solutions/Constructing-Smart-Grid-021909.pdf.