On Improving Capacity and Delay in Multi Tx/Rx Wireless Mesh Networks with Weighted Links

Hung-Yi Loo, Sieteng Soh
Department of Computing
Curtin University
Perth, Australia
{hung-yi.loo@student;s.soh@}.curtin.edu.au

Kwan-Wu Chin
School of Electrical, Computer and Telecommunications Engineering
University of Wollongong
Wollongong Australia
kwanwu@uow.edu.au

Abstract—**This paper considers the problem of deriving a link schedule for Time Division Multiple Access (TDMA)-based concurrent transmit/receive Wireless Mesh Networks (WMNs) that results in low end-to-end delays as well as high network capacity. We first propose a MAX-CUT heuristic approach, called Algo-2, that maximizes link activations in each slot of a super-frame. Algo-2 is shown to produce better network capacity as compared to existing heuristic approaches and significantly improves the super-frame length of an existing MAX-CUT approach that enforces 2-phase transmit-receive restriction – a node that transmits (receives) in slot $i \geq 1$ is to become a receiver (transmitter) in slot $i + 1$. Then, we propose a heuristic solution, called BDA, as a complement to existing schedulers to reduce transmission delays. Since BDA only reorders slots in the super-frame, it maintains each original schedule's super-frame length, and hence capacity, while reducing delays by up to 70% in 6-node random topology networks.**

Keywords: Wireless Mesh Networks; Transmission Delay; Multiple Transmit/Receive; Scheduler; Weighted Links;

I. INTRODUCTION

Wireless mesh networks (WMNs) are an important advancement in communication technologies [1]. Among others, they have been used as a communication backbone in rural areas and during natural disasters [2]. A key issue in WMNs is network capacity [1] [3]. Recently, researchers have equipped routers with multiple smart or directional antennas that allows each node to communicate with multiple neighboring routers simultaneously, *i.e.*, realizing a multi-transmit-receive (MTR) network. These MTR routers, however, must adhere to the following constraint: a node can either transmit (Tx) or receive (Rx) on a subset of its links, but not Tx and Rx simultaneously. For example, schedules A and B for the WMN in Figure 1 adhere to the constraint (link weights correspond to traffic demands). Note that deriving a schedule is NP-hard, since determining the set of links to be activated in each slot is equivalent to the well-known MAX-CUT problem [1].

Chin *et al.* [4] have recently proposed an efficient algorithm, called Algo-1, that derives a schedule for the said WMNs to increase the number of link activations per timeslot. Their greedy heuristic algorithm iteratively generates a MAX-CUT [1] to maximize the total number of link activations (capacity) in each slot, and hence, minimizing the super-frame length. The algorithm considers a 2-phase transmit-receive restriction imposed by the 2P protocol of [5] that enforces each node that transmits in slot i to become a receiver in slot $i + 1$. For example, Algo-1 generates a MAX-CUT ($P_1 = \{1,3\}, P_2 = \{2,4\}$) for the WMN in Fig. 1 to activate links $\{e_{12}, e_{34}, e_{32}\}$ in Slot 1 and links $\{e_{21}, e_{43}, e_{23}\}$ in Slot 2. Thus, the number of slots must always be even, as every second slot is a mirror of the previous. This rule, however, generates super-frames that are longer than non-2P based methods such as [3]. In particular, Dai *et al.* [3] considered the scheduling problem without the 2P restriction. Their heuristic algorithms first generate a conflict graph. They then generate the Maximum Independent Set (MIS) of the graph to obtain links that can be activated at each time slot. Additionally, they showed that relaxing the 2P restriction allows a significant improvement in WMN capacity.

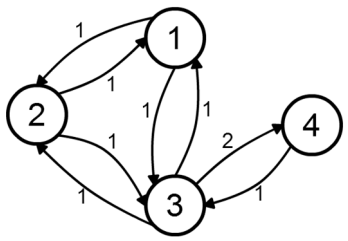

Fig. 1. A single channel MTR WMN. Schedule-A and B are possible outputs from the scheduler presented in [*3*].

Thus far, neither [3] or [4] has considered the issue of end-to-end transmission delay in WMNs [6] [7]. Referring to Fig. 1, assume node 1 is required to make a transmission to node 4; assuming shortest path routing, the demand is routed through links e_{13} and e_{34}. In Schedule A, link e_{13} is activated in slot 3 and e_{34} in slot 1 on the next iteration of the schedule. In Schedule B, however, e_{13} and e_{34} are activated in slot 2 and 3 in the same iteration. This means the transmission will require four timeslots using Schedule A, but only three in Schedule B.

Henceforth, this paper makes the following contributions. First, we propose a novel MAX-CUT based heuristic algorithm for scheduling links in MTR WMN that improves upon Dai *et al.* [3]'s heuristics. A key feature of our algorithm is that it does not require the generation of a conflict graph – a key advantage, as the size of the conflict graph increases rapidly

with network size. Note that a node in a conflict graph represents a link in regular WMN topology, and therefore the conflict graph of a $|V|$ nodes in a fully connected WMN will have $O(|V|^2)$ nodes and $O(|V|^4)$ links. Second, we propose a novel heuristic algorithm, called Bucket Draining Algorithm (BDA), to minimize the average end-to-end delay of an MTR WMN. Advantageously, BDA can be used to complement any TDMA link scheduler developed for an MTR WMN, *e.g.*, those in [3] and [4], in order to retroactively minimize their transmission delay without affecting super-frame length and capacity.

The rest of the paper is organized as follows. In Section II, we formally define the end-to-end delay problem model and domain. In Section III, we propose a two-step solution to produce a TDMA schedule from a weighted MTR WMN with good capacity and end-to-end delay. We evaluate the efficacy of our solution via simulation where we compare it against existing schedulers in Section IV. We conclude the paper in Section V.

II. The Problem

We model an MTR WMN with weighted links as a multigraph $G(V, E)$, where V and E correspond to the set of nodes/routers and directed links respectively. The number of edges connecting node i and j is denoted by its weight w_{ij}. Specifically, E is a multi-set where $w_{ij} \geq 1$ corresponds to w_{ij} copies of directed edge e_{ij} in E. We assume static nodes and links – no new nodes or links are added or removed at any stage, and all link weights remain the same. A solution to the MTR WMN link scheduling problem is to produce a super-frame $S = (S_1, S_2, \ldots, S_z)$, where each slot $S_i \in S$ contains a set of directed, transmitting links. A key constraint is that all edge e_{ij} must be activated *at least* $w_{ij} \geq 1$ times [3] [4]. Note, we say "at least" because opportunistic links (*i.e.*, those that were scheduled in prior slots) may be added into a slot subject to the constraint outlined in Section I.

Let d_{ab} be a demand from node a to b, and P_{ab} is the shortest path to route demand d_{ab} for each node pair $a, b \in V$. Thus, P_{ab} is a sequence of nodes starting from node a and ending at node b, where each pair of consecutive nodes form a link in E, and the length of the path is $|P_{ab}| - 1$. We define δ_{ab} as the time delay to transmit a packet for demand d_{ab}through P_{ab} w.r.t. S. Formally, given P_{ab} for a demand d_{ab}, and a super-frame S, the transmission delay of d_{ab} through P_{ab} is

$$\delta_{ab} = \sum_{i \in P_{ab}} W_i(a, b) \qquad (1)$$

where $W_i(a, b)$ is the waiting time for a node i to transmit after the transmission of its predecessor node $i - 1$, for each pair of sequenced nodes in P_{ab}. Note that $W_a(a, b)$ is the waiting time required for node a to transmit w.r.t. S and $W_b(a, b) = 0$, as node b is at the end of the path and does not need to transmit any further. For example, for d_{14} in Fig. 1 with Schedule A, $P_{14} = \{1,3,4\}$, and $\delta_{14} = W_1 + W_3 + W_4 = 3 + 1 + 0 = 4$.

The average transmission delay δ_{ave} in $G(V, E)$ w.r.t. a super-frame is calculated by taking the average of δ_{ab} over all demands d_{ab}. In this paper, we consider all possible (a, b) pairs, for $a, b \in V$, and thus,

$$\delta_{ave} = \frac{\sum_{a,b \in V; a \neq b} \delta_{ab}}{|V|(|V|-1)} \qquad (2)$$

Our end-to-end delay problem is to find a schedule S for $G(V, E)$ such that the average end-to-end delay in δ_{ave} is minimal while also maximizing its network capacity as defined in [3] and [4]. For example, a solution to the problem in Fig. 1 is $S = (\{e_{12}, e_{32}, e_{34}\}, \{e_{13}, e_{23}, e_{43}\}, \{e_{31}, e_{21}, e_{34}\})$ where $\delta_{12} = 1$, $\delta_{13} = 2$, $\delta_{14} = 3$, $\delta_{23} = 2$, $\delta_{24} = 3$, $\delta_{34} = 1$, $\delta_{41} = 3$, $\delta_{42} = 4$, $\delta_{43} = 2$, $\delta_{31} = 3$, $\delta_{32} = 1$, $\delta_{21} = 3$, and so $\delta_{ave} \cong 2.33$.

III. A Solution

We propose to solve the problem in two steps. First, we use Algo-2, an extension of Algo-1 [4], to heuristically generate super-frame *S* with maximum capacity. Second, we reorder the slots in *S* to produce a super-frame *R* that minimizes δ_{ave}. For the second step, one may use a brute force approach to generate all possible permutations of slots in *S* and select one that produces minimal δ_{ave}. However, this is computationally infeasible for large $|S|$ and $|V|$. Thus, in this paper we propose a heuristic algorithm called Bucket Draining Algorithm (BDA) to re-order slots. It is important to note that since *R* contains the same slots as *S*, the schedule has the same capacity and super-frame length as compared to *S* but with lower end-to-end delay between node pairs.

A. Algo-2 – A Maxcut-based Algorithm

Algo-2 iteratively finds the MAX-CUT [8] of the network, *i.e.*, partition the network into a bipartite graph such that the weighted links between the two partitions are maximized. Algo-2 uses the MAX-CUT to maximize the number of link activations per timeslot, as each MAX-CUT directly translates to a single timeslot in the generated super-frame [4]. However, since the problem to generate MAX-CUT is NP-Complete, this paper uses the greedy heuristic proposed in [4]. For each generated MAX-CUT, and thus link activations in a new slot, Algo-2 updates link weights, and generates another MAX-CUT, which in turn is used to obtain the link activation in the next slot. More specifically, Algo-2 runs the following steps:

1. Set $P_1 = V$, $P_2 = \phi$, and super-frame $S = \phi$.
2. For each node *n* in P_1, calculate its $\Delta_n \geq 0$ as
$$\Delta_n = \sum_{m \in P_1, m \neq n} w_{nm} - \sum_{m \in P_2} w_{mn}$$
3. Find node n in P_1 with the maximum Δ_n, If $\Delta_n > 0$, move n to P_2. If $\Delta_n = 0$, move n to P_2 if $|P_2| < |P_1|$.
4. Repeat steps 2 and 3 until all calculated Δ values become negative. The links connecting nodes in P_2 to nodes in P_1 form an approximate MAX-CUT.
5. Add the timeslot $\{e_{ij} | i \in P_2 \text{ and } j \in P_1\}$ to S and decrement the weights $\{w_{ij} | i \in P_2 \text{ and } j \in P_1\}$ by 1. If any $w_{ij} = 0$, delete e_{ij} from E.

6. Reset $P_1 = V$ and $P_2 = \phi$, then repeat steps 2 to 5 until there are no more links in E, meaning the weight requirements have been satisfied by the schedule.

7. Return S.

After the initialization in Step 1, Steps 2 to 4 greedily generate a MAX-CUT. In Step 2, the first term is the total number of outgoing link from node n to all nodes in P_1, while the second term is the incoming link from nodes in P_2 to node n. The algorithm aims to get node n that has the maximum difference between the values of the two terms, i.e., maximum Δ_n, for each node in P_1 and put it in P_2 so that it maximizes the number of links connecting nodes in P_2 to every node in P_1, i.e., a MAX-CUT. Note that when each Δ_n is negative, moving a node from P_1 to P_2 does not increase the size of the MAX-CUT, and thus Step 5 returns the MAX-CUT. Step 6 creates a new time slot in S that contains all links connecting nodes in P_2 to P_1, and reduces all the links' weight by one; a link with $w_{ij} = 0$ has been activated as many times as required, and therefore is deleted from the network. Step 6 reinitializes P_1 and P_2 and repeat Steps 2 to 5 to generate the next MAX-CUT, and hence link activations in the next time slot in S. Step 7 returns the super-frame S after all links in E have been activated at least as many times as required by their weights. Note that the only difference between Algo-1 [4] and Algo-2 is in Step 5. Since Algo-1 assumes 2P protocol, for each MAX-CUT, it generates 2 consecutive slots, the first contains all links from nodes in P_2 to all nodes in P_1, and the second contains all links from nodes in P_1 to all nodes in P_2. Therefore, the time complexity of Algo-2 can be calculated similar to that for Algo-1, *i.e.* $O(|V|^2)$.

To illustrate Algo-2, we show how it generates Schedule A for the graph in Figure 1; initially, $P_1 = \{1,2,3,4\}$, $P_2 = \phi$, and $S = \phi$.

1. To generate the first timeslot, Algo-2 calculates the Δ values $\{\Delta_1, \Delta_2, \Delta_3, \Delta_4\} = \{2,2,4,1\}$ in Step 2, and since $\Delta_3 > 0$ is the maximum, Step 3 moves node 3 from P_1 to P_2, and thus $P_1 = \{1,2,4\}$ and $P_2 = \{3\}$. Repeating Steps 2 and 3, Algo-2 calculates $\{\Delta_1, \Delta_2, \Delta_4\} = \{0,0,-2\}$ where the maximum is either $\Delta_1 = 0$ or $\Delta_2 = 0$. Since $|P_2| < |P_1|$, Step 3 randomly chooses Δ_2 and moves node 2 to get $P_1 = \{1,4\}$ and $P_2 = \{2,3\}$. Calculating the Δ values again, it gets $\{\Delta_1, \Delta_4\} = \{-2,-2\}$, thus the algorithm obtains a MAX-CUT $P_1 = \{1,4\}$, $P_2 = \{2,3\}$, and Step 5 adds links $\{e_{21}, e_{31}, e_{34}\}$ to the first timeslot A_1 in Schedule A and decrements the weights $\{w_{21}, w_{31}, w_{34}\}$ by one.

2. To find the second timeslot, Algo-2 simply repeats this procedure with the updated weights. First, it calculates $\{\Delta_1, \Delta_2, \Delta_3, \Delta_4\} = \{2,1,2,1\}$. Node 3 has the maximum Δ so Algo-2 moves node 3 and gets $P_1 = \{1,2,4\}$ and $P_2 = \{3\}$. Then, it calculates $\{\Delta_1, \Delta_2, \Delta_4\} = \{1,-1,-1\}$, moves node 1, and obtains $P_1 = \{2,4\}$ and $P_2 = \{1,3\}$. Finally, it calculates $\{\Delta_2, \Delta_4\} = \{-2,-1\}$, sets $A_2 = \{e_{12}, e_{32}, e_{34}\}$, and decrements the corresponding weights $\{w_{12}, w_{32}, w_{34}\}$ by one.

3. To find the third timeslot, Algo-2 calculates $\{\Delta_1, \Delta_2, \Delta_3, \Delta_4\} = \{1,1,0,1\}$ and moves node 4. Then it calculates $\{\Delta_1, \Delta_2, \Delta_3\} = \{1,1,-1\}$ and move node 2. Finally, it calculates $\{\Delta_1, \Delta_3\} = \{1,-2\}$, move node 1, then stops after obtaining a MAX-CUT $P_1 = \{3\}$, $P_2 = \{1,2,4\}$, and so Step 5 sets $A_3 = \{e_{13}, e_{23}, e_{43}\}$ and decrements the links' corresponding weights.

4. At this point, all links have been deleted from E, *i.e.* all links have weight = 0, and thus Step 6 stops and Step 7 returns the super-frame A as shown in Fig. 1.

B. Bucket Draining Algorithm

Our proposed BDA reorders the slots in S to produce a super-frame R that minimizes δ_{ave}. BDA is based on the reasoning that each link should be activated 'fairly' and proportional to its weight, with its activation spread evenly across the super-frame. Fair and even activation of all links in a network will avoid some links being neglected ('starved') for a disproportionate amount of time due to other links being inadvertently prioritized. It is possible for a link to be temporarily starved within a series of slots in the super-frame. If there is a large contiguous series of slots in the super-frame in which a particular link is not activated, it may increase the delay for any demand routed through that link since the demand transmission will most likely have to wait longer than is necessary for that particular link to be activated. On the other hand, if a link is only activated within a consecutive slots, *e.g.*, slots at the beginning of the super-frame, it may increase delay for demands routed through that links that require its activation on the other parts of the super-frame, *e.g.*, at the end. Fair and even activation of links will thus allow more demand routes to be consecutively activated to completion within a reasonable time.

To fairly activate links, BDA creates a bucket $b_{ab} = (w_{ab}, \text{flag}_{ab})$ for each link e_{ab}. Here, $\text{flag}_{ab} = true$ signifies that the bucket has been *drained*, *i.e.*, the link in the bucket has been activated once within a round. The flag ensures that no bucket, hence link, is activated more than once while links in other buckets, *i.e.*, those with flags set to *false*, have yet to be activated within this round. More specifically, in order to generate timeslots in R given a super-frame S, BDA runs the following steps:

1. Create a bucket $b_{ab} = (w_{ab}, \text{flag}_{ab})$ for each link $e_{ab} \in E$.

2. Find the heaviest non-empty bucket $b_{\alpha\beta}$ with $\text{flag}_{\alpha\beta} = false$, and find $S_k \in S$ such that $e_{\alpha\beta} \in S_k$. If there are two or more buckets with the heaviest weight, arbitrarily pick any one of them.

3. For each $e_{ab} \in S_k$, drain b_{ab} by decrementing w_{ab} by 1 and set $\text{flag}_{ab} = true$. Append S_k to the new schedule sequence R, and remove it from S.

4. If S contains only one slot, append this slot to R, and terminate BDA with R as its output.
5. If the buckets are all empty, append all of the remaining slots in S to R, and terminate BDA with R as its output.
6. If $\text{flag}_{ab} = true$ for all edge e_{ab}, reset all $\text{flag}_{ab} = false$.
7. Repeat from Step 2.

After initialization in Step 1, BDA finds a bucket with the heaviest weight in Step 2 and find a slot S_k that contains the bucket's corresponding edge. The step selects a link with the heaviest weight so that the link's activation can be more spread throughout the super-frame. Step 3 drains the buckets whose edges are in slot S_k, and Step 4 moves the slot from S to R in order. BDA terminates and returns R in either Step 5 or Step 6. When S contains only one slot, BDA directly knows the position of the slot in R, and therefore it returns after appending the slot to R. An empty bucket means that its corresponding edge e_{ij} has been activated w_{ij} times as required, and thus Step 6 terminates when all buckets are empty. For this case, the step appends all remaining slots in S to R. Step 7 resets all flags to *false* to start a new round of link activations once each of the links has been activated one time. It can be shown that the running time of these steps is at most $O(|E|.|S|^2)$ where $|E|$ is the number of edges in the WMN and $|S|$ is the number of slots in the given super-frame.

We now show how BDA re-arranges Schedule A of Fig. 1 with $S = (S_1, S_2, S_3)$. Step 1 constructs eight buckets, corresponding to the eight links of the WMN: $\{b_{12}, b_{21}, b_{13}, b_{31}, b_{23}, b_{32}, b_{34}, b_{43}\}$. Their corresponding weights are $\{1, 1, 1, 1, 1, 1, 2, 1\}$ and $\text{flag}_{ij} = false$ for all edge e_{ij}. Step 2 chooses bucket b_{34} because it has the highest weight. Since $e_{34} \in S_2 = \{e_{12}, e_{32}, e_{34}\}$, Step 3 drains b_{12}, b_{32} and b_{34} and sets flag_{12}, flag_{32} and flag_{34} to *true*. Step 4 removes S_2 from S and adds it to R, i.e., $R = (S_2)$. At this stage, the bucket weights are {0, 1, 1, 1, 1, 0, 1, 1} and $S = (S_1, S_3)$, and thus Step 5 and Step 6 are skipped. Step 7 is also skipped since not all flags are set to *true* and BDA repeats from Step 2. Since all non-empty buckets have equal weight, it chooses any of them, except b_{34} whose flag is *true* since it was recently drained. Assume it selects b_{13}, and since $S_3 = \{e_{13}, e_{23}, e_{43}\}$, Step 3 drains the corresponding buckets and flags them. Step 4 removes S_3 from S and sets $R = (S_2, S_3)$. Since there is only one remaining slot in S, Step 5 assigned this last slot to R and terminates BDA that returns the newly sequenced schedule $R = (S_2, S_3, S_1)$ with $\delta_{ave} \cong 2.33$, which is lower than the $\delta_{ave} \cong 2.75$ of Schedule A.

IV. Evaluation

In order to produce a fair comparison, our simulations conform to that of [3] and [4]. We generate random networks, where each network has six nodes with density (*i.e.*, number of links divided by max number of links) varying from 0.1 to 1.0. Note that a fully connected network has a density of 1.0. We generate 50 random networks for each density value. All link weights have values in the range [1,10].

To evaluate the performance of Algo-2, we have compared derived schedules against those produced by HWF and MDF [3] in terms of their average super-frame length and link activations/capacity. Further, we benchmarked Algo-2, HWF and MDF against the Linear Programming (LP) approach presented in [3]. Note, the LP approach generates schedule S with optimal super-frame length but is computationally feasible only for networks with $|V| \leq 6$.

Table 1 shows the super-frame length produced by each scheduler. Algo-2 consistently outperforms the MIS-based heuristics of [3]. Further, the super-frame length generated by Algo-2 is only 1.06% worse than the optimal one generated by the exponential time LP approach in [3]. Table 1 also shows that the 2P restriction, used in Algo-1, significantly increases the super-frame length when compared to the non-2P approaches.

TABLE I
SUPER-FRAME LENGTH OF SCHEDULES BY NETWORK DENSITY

Network Density	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Algo-1	16.60	20.72	25.20	29.20	30.72	33.80	34.88	37.20	40.00	41.68
Algo-2	12.66	15.30	17.84	20.26	21.58	23.02	23.52	24.66	26.10	27.24
HWF	12.80	15.40	18.56	20.94	22.16	23.76	24.22	25.24	26.76	28.06
MDF	12.76	15.34	18.24	20.96	22.40	23.94	24.60	26.02	28.06	29.42
LP	12.64	15.26	17.84	20.18	21.52	22.92	23.32	24.32	25.52	26.44

Table 2 shows the performance among the five evaluated approaches in terms of capacity, calculated as

$$\text{Capacity} = \frac{\text{Total Number of Link Activations}}{\text{Superframe Length}} \quad (3)$$

As shown in the table, Algo-2 outperforms HWF and MDF. Surprisingly, Algo-2 also outperforms LP by an average of 1.97% for networks with density above 0.6. Note that Algo-2 uses MAX-CUT to generate maximum link activations for each slot, while LP utilizes the MIS approach. Our results show that MAX-CUT is more effective than MIS at producing super-frames with good capacity. This observation is further supported by the capacity produced by Algo-1, another MAX-CUT based algorithm for 2P. Algo-1 and Algo-2 produce similar capacity.

TABLE II
NETWORK CAPACITY OF SCHEDULES BY NETWORK DENSITY

Network Density	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Algo-1	1.84	2.68	3.72	4.65	5.38	6.19	6.62	7.56	8.24	8.92
Algo-2	1.79	2.70	3.72	4.67	5.40	6.29	6.69	7.60	8.31	8.90
HWF	1.75	2.52	3.40	4.24	4.94	5.74	6.16	7.08	7.69	8.37
MDF	1.75	2.53	3.47	4.25	4.94	5.73	6.06	6.90	7.44	8.05
LP	1.97	3.00	4.09	4.90	5.56	6.35	6.64	7.52	8.03	8.70

Figure 2 shows the end-to-end delay, calculated using δ_{ave}; the solid lines show the delay without BDA, while the dashed lines show the benefits of BDA. Although Table 1 shows that the LP based approach optimally minimizes super-frame lengths, the schedules generated resulted in the highest end-to-end delays. In contrast, Algo-2 produces the lowest delay. Note

that, by nature, LP, HWF, MDF and Algo-1 may schedule the same set of link activations repeatedly consecutively [3] [4], which may starve other links. On the other hand, Algo-2 avoids this by scheduling each slot independently and thus fairly distributes link activations, which, as expected, lowers the end-to-end delay. The figure shows that, using our delay model, as defined by (2), a shorter super-frame length or higher capacity is not a necessary condition to produce shorter end-to-end delays; *e.g.*, LP versus Algo-1 or LP versus MDF, respectively.

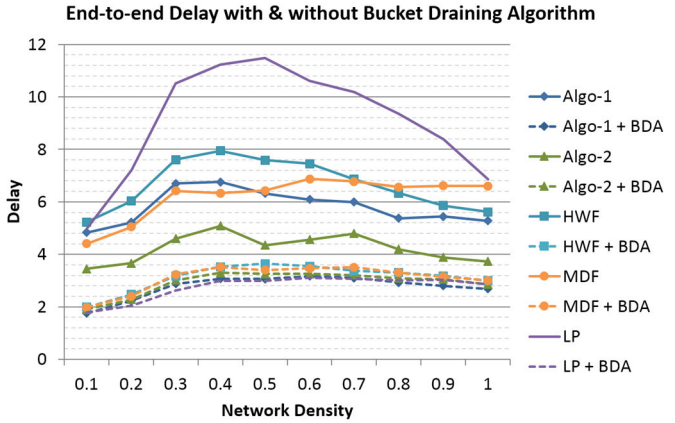

Fig. 2. End-to-end delay of the schedulers from [*3*] and [*4*] before (solid lines) and after (dashed lines) applying BDA.

Figure 2 shows that BDA is effective in reducing the average end-to-end delay of schedules generated by the five scheduling algorithms. Since BDA only reorders the slots in each schedule, it does not compromise on super-frame length and capacity. Our results show that BDA reduces the average delay of the schedules generated by Algo-1, Algo-2, HWF, MDF, LP, by 52%, 31%, 53%, 50%, and 70% respectively. Thus, BDA is a good complement to any MTR WMN TDMA-based link scheduler. As shown, in terms of delay, all algorithms complemented by BDA produce schedules with comparable average end-to-end delay. Thus, when using our delay model in (2), the order of slots in a super-frame is the main factor that affects the end-to-end delay.

Since the average delay, δ_{ave}, considers all possible demands and does not prioritize individual demands, a minimal δ_{ave} is not optimal for specific end-to-end throughput for a single demand or subset of demands. Thus, to further evaluate the effect of our BDA on delay performance, we ran simulations to compute change in delay for each (s,t) demand before and after using BDA on 500 fully-connected 6 node networks with weights in the range [1, 10]. A total of 14965 individual demands' delay changes were considered.

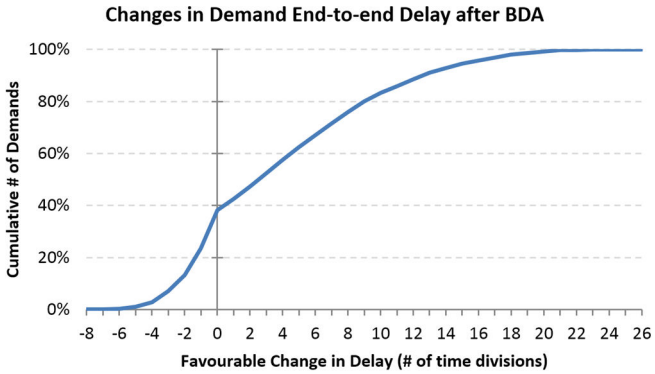

Fig. 3. CDF showing number of demands (as a percentage of the total number of demands) by their change in delay. Change in delay is measured in TDMA slots; *i.e.*, the difference between two δ_{ab} values, before and after BDA, as shown in (1) in Section II. A positive change in delay means a delay improvement, and a negative change in delay means a delay sacrifice.

The CDF in Figure 3 further illustrates the effect that BDA has on the delay of demands. Approximately 40% of demands suffer a delay sacrifice of 1 to 8 time units after running BDA, with 25% of demands having a delay sacrifice of no more than two time units. On the other hand, approximately 40% of demands show an improvement in delay of 1 to 8 time units after BDA, while another 20% of demands show extreme delay improvement of 8 to 26 time units. Hence, it is evident that the delay improvements outweigh the delay sacrifices by a significant margin, which results in a reduction in δ_{ave}.

Our analysis shows that there is possible room for improvement in BDA, despite the fact that it achieves our goals as outlined in Section II. Ideally, we would want to improve the delay of all demands without sacrificing any. On the other hand, we can justify these sacrifices as long as the delay improvement on the other demands are sufficient, *e.g.*, a delay sacrifice of 4 time units is reasonable if we also achieve a delay improvement of 25 time units on one or more other demands.

As a future work, we would like to develop a variant of the algorithm which achieves similar overall delay improvement without having to sacrifice the delay of any demands. This restriction would theoretically limit the achievable amount of delay improvement; however we would then be able to guarantee that no demand's delay is worsened.

V. Conclusion

This paper has formally defined the end-to-end delay problem, and has presented a two-step solution using two new heuristic algorithms, Algo-2 and BDA, to optimize the delay and capacity of TDMA-based MTR WMN schedules. Algo-2, a MAX-CUT based approach, produces better capacity than two state-of-the-art MIS-based approaches. Simulations show that BDA is very effective at reducing average delay of the schedules generated by the existing MTR schedulers without compromising their super-frame length as well as capacity.

VI. References

[1] K. Chin, S. Soh, and C. Meng, "Novel scheduling algorithms for concurrent transmit/receive wireless mesh networks," *Computer Networks*, vol. 56, no. 4, pp. 1200-1214, Mar 2012.

[2] S. Rahman, A. Yarali, and B. Ahsant, "Wireless Mesh Networking: A Key Solution for Emergency & Rural Applications," in *Advances in Mesh Networks*, 2009, pp. 143-149.

[3] H. Dai, S. C. Liew, and L. Fu, "Link Scheduling in Multi-Transmit-Receive Wireless Networks," in *IEEE Local Computer Networks*, Oct 2011, pp. 199-202.

[4] K. Chin, S. Soh, and C. Meng, "A Novel Scheduler for Concurrent Tx/Rx Wireless Mesh Networks with Weighted Links," *IEEE Communications Letters*, vol. 16, no. 2, pp. 246-248, Feb 2012.

[5] B. Raman and K. Chebrolu, "Design and evaluation of a new MAC protocol for long-distance 802.11 mesh networks," in *ACM MOBICOM*, New York, 2005, pp. 156-169.

[6] P. Djukic and S. Valaee, "Delay aware link scheduling for multi-hop TDMA wireless networks," *IEEE/ACM Transactions on Networking*, vol. 17, no. 3, pp. 870-883, June 2009.

[7] V. Gabale, A. Chiplunkar, B. Raman, and P. Dutta, "Delay Contained: Scheduling Voice Over Multi-hop Multi-channel Wireless Mesh Networks," in *COMSNETS*, Bangalore, 2011, pp. 1-10.

[8] M. Goemans and D. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *Journal of the ACM*, vol. 42, no. 6, pp. 1115-1145, November 1995.