A Novel Link Scheduler for Personalized Broadcast in Multi Tx/Rx Wireless Mesh Networks

He Wang, Kwan-Wu Chin
School of Electrical, Computer and Telecommunications Engineering
University of Wollongong, NSW, Australia
Email: hw407@uowmail.edu.au, kwanwu@uow.edu.au

Sieteng Soh
Department of Computing
Curtin University, WA, Australia
Email: s.soh@curtin.edu.au

***Abstract*—The personalized broadcast problem calls for a link schedule with the shortest makespan or slots to deliver all data located at a gateway destined for nodes in a multi-hop wireless network. In this paper, we address this fundamental problem with consideration for the multiple transmit or receive capability of nodes as well as their ability to boost the capacity of a link via spatial multiplexing or multiple radios. We derive new makespan bounds for arbitrary tree topologies and propose a new link scheduler called Algo-PB to generate a personalized broadcast schedule with minimal schedule length. Simulation results show that the schedule length generated by Algo-PB outperforms state-of-the-art algorithms by at most 20% and the difference between Algo-PB and the theoretical lower bound is at most 10%.**

I. Introduction

Nodes in Multi-Transmit-Receive (MTR) Wireless Mesh Network (WMNs) are able to transmit to *or* receive from multiple neighbors concurrently. This is enabled by the advances in Multi-User Multiple Input and Multiple Output (MIMO) [1] technology, or by equipping nodes with multiple radios and directional antennas [2]. A key constraint of a MIMO or [2] based WMN is that a node must not transmit *and* receive concurrently, so called *no-tx-rx* or half-duplex constraint. Interestingly, the capacity of links can be boosted via spatial multiplexing, a key feature of MIMO technology, or additional radios. Consequently, nodes are able to transmit multiple data streams to the same neighbor, which we refer to as *link upgrade*. Figure 2 shows the MTR and link upgrade capability of nodes.

Data forwarding to/from a gateway node, with access to the Internet, is a fundamental operation in WMNs. Crucially, data must be forwarded to/from the gateway in minimum time. In this respect, we address the personalized broadcast problem [3]. Briefly, given a network with one gateway, and each node has a set of packets to be received from the gateway, the aim is to generate a collision free, i.e., no-tx-rx, link schedule with the minimum makespan (in terms of slots) to transfer all data packets located at the gateway to their respective destination node. Note, 'personalized broadcast' refers to the fact that the packets to be delivered are unique and has a designated destination. This problem is thus different from the traditional broadcast scheduling problem [4] where all nodes receive the *same* packet from the gateway. Figure 1 shows an example schedule for the case where each node has one packet and one antenna to transmit/receive to/from each neighbor; solid arrows indicate packet transmissions. The makespan for the schedule is four slots. We note that the authors of [5] have shown this problem is equivalent to the data collection or data gathering problem. In particular, we can "reverse" any derived schedule for the personalized broadcast problem and use it for data collection/evacuation.

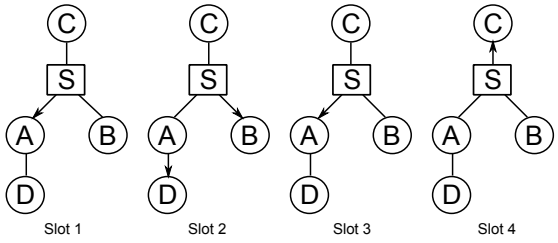


Fig. 1: An example of personalized broadcast

We now highlight the advantages of MTR and link upgrade for the personalized broadcast problem. First consider the case where MTR and link upgrade is not applied. In this case, each node only transmits one packet in each slot; see Figure 1. Next consider Figure 2, where each node is able to send or forward multiple packets to one or more neighbors in one slot with the benefit of MTR and link upgrade; the latter allows links to have a higher capacity via the addition of unused antennas. Specifically, in Slot 1, link (S,A) has two radios/antennas. Further, as shown in Slot 2, using MTR, S can transmit to B and C at the same time. Notice that due to the *no-tx-rx* constraint, node A can transmit to D but cannot receive at the same time. As a result, the personalized broadcast schedule requires two slots with MTR and link upgrade. Otherwise, we will need a schedule with four slots; see Figure 1.

We note that there are many link scheduling algorithms or Medium Access Control (MAC) protocols for data collection or personalized broadcast problem in Wireless Sensor Networks (WSNs). These approaches focus on energy consumption, reliability and quality of services (QoS) [6]. We only review those approaches that aim to derive the shortest schedule length. In [7], the authors analyze the lower bound for line, multiline and tree topologies and derive the optimal data collection time. The authors of [7], however, did not consider MTR or link upgrade. The authors of [8] analyze the data collection theoretical lower bounds for line and tree topologies.

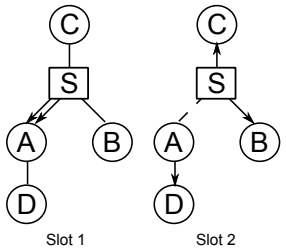


Fig. 2: An example of personalized broadcast with MTR and link upgrade. Solid arrows indicate packet transmission and dashed lines indicate the *no-tx-rx* constraint.

Different from [7], the authors in [8] do not consider spatial reuse and they exploit multiple channels to eliminate interference. They then propose a data collection algorithm that achieves the theoretical lower bound. However, their proposed algorithm requires a strict balanced binary tree topology, and they do not consider MTR or link upgrade. In [9], the authors study the fast data collection problem in duty-cycled wireless sensor networks. They first propose a centralized algorithm that achieves the optimal collection time; i.e., in the non-duty-cycled case. However, they assume nodes have an omni-directional antenna. The authors in [10] study the fast data collection problem in tree-based wireless sensor networks. They propose two data scheduling algorithms to generate a minimal data collection schedule. The first algorithm considers the case where each node generates data periodically and data is aggregated by parent nodes. However, they only consider the case where each node has only one packet to be collected and they do not consider MTR and link upgrade. The authors of [5] study the data gathering problem with multi-directional antennas. They give an algorithm to construct the transmission schedule makespan for tree topologies where each node has a non-zero number of packets to be delivered and all nodes do not have a buffer. However, they do not consider MTR and link upgrade. To sum up, approaches [5] [7] [8] [9] and [10] are not optimized for MTR WMNs as they do not consider MTR or link upgrade. Thus when we apply these approaches to MTR WMNs, they do not generate the shortest link schedule. As an aside, we point out that our problem is different to the minimum-latency broadcast scheduling (MLBS) [11] and the delay constrained multicast scheduling [12] problem. They are concerned with delivering the same packet to all nodes, whereas our nodes may have zero or more distinct packets. Nevertheless, deriving the minimum broadcast schedule for MTR WMNs is an interesting future work.

Given the lack of scheduling approaches for personalized broadcast in MTR WMNs, this paper makes the following contributions: (i) we give the theoretical personalized broadcast lower bound for tree topologies over a MTR capable network model; see Section III; (ii) we present Algo-PB, a link scheduler that generates a personalized broadcast schedule for arbitrary tree topologies with minimal schedule length; see Section IV; Our simulation results, see Section V, show that Algo-PB generates up to 20% shorter schedule lengths as compared to the algorithm proposed in [5], and the difference between our algorithm and the theoretical lower bound is at most 10%. We conclude in Section VI.

II. Background

Let a MTR WMN be represented as a graph $G(V, E)$ where V denotes the set of vertices/nodes/routers and E denotes the set of directional links. Each node $u \in V$ has Δ_u radios/antennas, meaning it can transmit/receive to/from Δ_u neighbors simultaneously. A key consideration is that the end nodes of a link must dedicate an antenna for transmission/reception, and in the case of a MIMO link, it has sufficient antennas to null/suppress any interference. Let $s \in V$ be a node designated as a sink/gateway. For a given node $u \in V-s$, let $r_u \geq 0$ be the number of requests/packets to be received from a gateway. Time is slotted where each slot corresponds to the transmission of one packet over an antenna. This means a node u can transmit or receive up to Δ_u packets at the same time. We remark that synchronization can be achieved by equipping each node with a GPS module. We denote the spanning tree of G rooted at s by T^s and $V_s \subset V$ is the set of descendants for gateway 's'. We use T_i^s to denote a subtree of T rooted at the i-th child of s. For an arbitrary topology with a gateway s, the tree is established by performing a breadth-first search (BFS) at the gateway.

Consider a MTR WMN G(V, E) with one gateway s and each non-gateway node u has a demand of $r_u \geq 0$ packets that are to be received from the gateway. Our aim is to solve the personalized broadcast problem by deriving the shortest *no-tx-rx* schedule (in slots) that delivers r_u packets from s to each node u, where $u \in V_s$.

III. Lower Bound

In this section, we analyze the theoretical lower bound of the personalized broadcast schedule rooted at s, i.e., T^s. Remember T_i^s denotes a subtree of T^s rooted at the i-th child of s. Further, let $\delta(u, s)$ be the makespan to transmit r_u packets from s to u. We then have following propositions.

Proposition 1. *The makespan to transmit $r_u > 0$ packets form s to u in personalized broadcast schedule of T^s is (i) $\delta(u,s) \geq dist(u,s) + 2 \times \lceil \frac{(r_u - \Delta_u)}{\Delta_u} \rceil$ slots for $dist(u,s) \geq 2$, or (ii) $\delta(u,s) \geq dist(u,s) + \times \lceil \frac{(r_u - \Delta_u)}{\Delta_u} \rceil$ slots, for $dist(u,s) = 1$.*

Proof. First consider the case where $dist(u,s) \geq 2$. For the best case, we consider $\Delta_v \geq \Delta_u$ for each node v, including s, in the path from s to u. Observe that the first Δ_u packets from s reach u no faster than $dist(u,s)$ slots. For $r_u \leq \Delta_u$, $\lceil \frac{(r_u - \Delta_u)}{\Delta_u} \rceil = 0$, and thus the proposition is true for this case. However, for $r_u > \Delta_u$, we must consider the *no-tx-rx* constraint. Specifically, any node in the path from s to u can only transmit or receive up to Δ_u packets at a time. Thus, u receives the next Δ_u packets no earlier than two slots after $dist(u,s)$; i.e., the time it receives the first Δ_u packets. In general, in the best case, u can receive up to Δ_u of the remaining $r_u - \Delta_u$ packets every two slots. Thus, u receives the remaining $r_u - \Delta_u$ packets in $\lceil \frac{r_u - \Delta_u}{\Delta_u} \rceil$ receptions, giving

a makespan of $dist(u,s) + 2 \times \lceil \frac{(r_u - \Delta_u)}{\Delta_u} \rceil$ slots. For case (ii), since u is only one hop away from s, it receives up to Δ_u packets every slot. Specifically, for $r_u \leq \Delta_u$, it receives the r_u packets in $dist(u,s) = 1$ slot, while for $r_u > \Delta_u$, it receives all r_u packets in $\psi + \lceil \frac{(r_u - \Delta_u)}{\Delta_u} \rceil$ slots, proving the proposition for case (ii). □

Let L_i be the number of levels in sub-tree T_i^s, and r_i^l be the total number of packets to be transmitted to level l of sub-tree T_i^s for $l \leq l \leq L_i$. We define D_i^l to be the *last slot* that a node at level l of sub-tree T_i^s receives its last packet. For example, in network shown in Figure 3, suppose node G and H receive their last packet at slot 4 and 5 respectively, thus $D_A^3 = 5$.

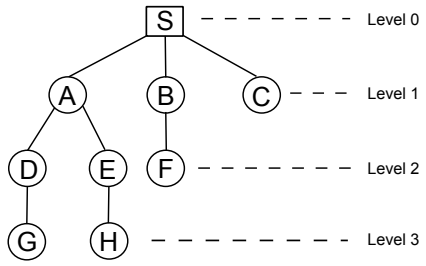


Fig. 3: Example topology

Proposition 2. *For any node u at level l in a T_i^s and $r_i^l > 0$, (i) $D_i^l = l + 2 \times \lceil \frac{r_i^l - \Delta_u}{\Delta_u} \rceil + 2 \times \lfloor \frac{\sum_{m=l+1}^{L_i} w_i^m}{\Delta_u} \rfloor$, for $l \geq 2$, or (ii) $D_i^l = l + \lceil \frac{r_i^l - \Delta_u}{\Delta_u} \rceil + 2 \times \lfloor \frac{\sum_{m=l+1}^{L_i} w_i^m}{\Delta_u} \rfloor$, for $l = 1$.*

Proof. For a node u at level l and the i-th child of gateway s, Proposition 1 obtains (i) $\delta(u,i) \leq l + 2 \times \lfloor \frac{\sum_{m=l+1}^{L_i} w_i^m}{\Delta_u} \rfloor$ slots for $l \geq 2$, or (ii) $\delta(u,i) \geq l + \lfloor \frac{\sum_{m=l+1}^{L_i} w_i^m}{\Delta_u} \rfloor$ slots, for $l = 1$. Node u will first relay the packets destined to its descendants. This incurs $2 \times \lfloor \frac{\sum_{m=l+1}^{L_i} w_i^m}{\Delta_u} \rfloor$ slots. After that, it starts receiving its first packet from the gateway. Each of the relayed packets takes two slots because the gateway transmits at an interval of two slots due to the *no-tx-rx* constraint. The last busy slot, i.e., D_i^l, for subtree rooted at i is thus the summation of the time required to relay descendants' packets plus the time to receive all its packets. □

As a result of Proposition 1 and Proposition 2, if $r_u = 0$, the number of slots for node u to receive packets located at itself is 0. Also, if $r_u^l = 0$, the number of slots needed to transfer packets to level l of sub-tree T_i is zero; i.e., $D_i^l = 0$ if $r_i^l = 0$.

Proposition 3. *The makespan of the personalized broadcast schedule for a subtree T_i^s is lower bounded by $D_i = MAX(D_i^l)$ for $l \in [1, \ldots, L_i]$.*

Proof. According to Proposition 2, any node u at level l of sub-tree T_i^s has D_i^l. This implies that the last busy slot of subtree T_i^s cannot be earlier than $MAX(D_i^l)$, where for $l \in [1, \ldots, L_i]$. □

Proposition 4. *The makespan of the personalized broadcast schedule for a tree T^s with θ subtrees is lower bounded by $MAX(D_{max}, \lceil \frac{\sum_{u \in V_s} r_u}{\Delta_u} \rceil)$, where $D_{max} = MAX(D_i)$ for $i \in [1, 2, \ldots \theta]$*

Proof. Without lost of generality, the first subtree of T^s, i.e., T_1^s, produces D_{max}. According to Proposition 2, the root of each sub-tree T_i^s will first receive packets located at its sub-tree and forward them onwards. This means the gateway node s cannot serve the same sub-tree for two continuous slots due to the *no-tx-rx* constraint. Consequently, we can *interleave* the transmissions of two sub-trees. For example, node s transmits to sub-tree T_1^s in slots 1, 3, 5, and so forth and transmits to sub-tree T_2^s in slot 2, 4, 6, ... In Proposition 3 we have shown that D_i is the total number of slots needed to deliver all packets to sub-tree T_i^s. We now show the following two facts.

- Case 1: $D_1 > D_2 + D_3 + \ldots + D_\theta$. All other sub-trees can be interleaved with the transmissions to T_1^s. That is, the transmissions to all sub-trees T_2^s to T_θ^s will finish earlier than D_1, thus the total number of slots needed to deliver all packets to nodes in T is lower bounded by D_1, i.e., D_{max}.
- Case 2: $D_1 \leq D_2 + D_3 + \ldots + D_\theta$. In this case, all other sub-trees cannot be fully interleaved with the transmission to T_1^s. Thus we prove a *loose* lower bound here, i.e., we consider only the number of slots needed for the gateway to transmit all packets to the nodes located at the first level, which is $\lceil \frac{\sum_{u \in V_s} r_u}{\Delta_u} \rceil$. Thus the number of slots needed to deliver all packets will not be shorter than $\lceil \frac{\sum_{u \in V_s} r_u}{\Delta_u} \rceil$.

We note that the schedule makespan will not be shorter than either D_1, i.e., D_{max} or the number of slots needed for the gateway to inject all packets into the network, thus we have the personalized broadcast lower bound $MAX(D_{max}, \lceil \frac{\sum_{u \in V_s} r_u}{\Delta_u} \rceil)$. □

We now give a brief example and show how each lower bound is calculated. For the topology shown in Figure 3, assume s is the gateway and each node has one packet; $r_u = 1$ for $u \in V_s = \{A, B, C, D, E, F, G, H\}$. We denote the sub-tree rooted at node A, B and C as as T_1^s, T_2^s and T_3^s, respectively. We also assume each node has $\Delta_u = 2$ antennas.

We now focus on sub-tree T_1. The total weight of level 3 of sub-tree T_1^s is $r_1^3 = 2$, and nodes located at level 3 do not need to forward packets as there are no nodes located beyond level 3. According to Proposition 2, we have $D_1^3 = 3$. Now consider level 2 of T_1^s. Nodes at level 2 need to forward packets to nodes located at level 3 first. For nodes located at level 2, they need to forward packets located at level 3 before receiving packets located at level 2. As $r_1^3 = 2$, nodes located at level 2 need to forward packets to level 3 $\lfloor \frac{\sum_{m=l+1}^{L_i} w_i^m}{\Delta} \rfloor$ times which incurs $2 \times \lfloor \frac{\sum_{m=l+1}^{L_i} w_i^m}{\Delta} \rfloor$ slots. We then calculate $D_1^2 = 4$. Similarly, $D_1^1 = 5$. According to Proposition 3, the lower bound for sub-tree T_1^s is $D_1 = 5$. We apply the same

procedure for sub-tree T_2^s and T_3^s and we get $D_2 = 2$ and $D_3 = 1$. According to Proposition 4, as $D_{max} = D_1 > D_2 + D_3$, the lower bound of tree T^s is $D_{max} = 5$.

IV. Solution

We now propose a centralized link scheduling algorithm to solve the aforementioned personalized broadcast problem. Note, we do not consider a distributed solution because information such as the topology and buffered packets are located conveniently at the gateway. We assume the tree rooted at a gateway is given; i.e., the routing information for each node to the gateway is known. Our algorithm, called Algo-PB, schedules links in a path-by-path manner. Its key idea is to always schedule one packet to the farthest node who has not received all its packets, and labels each link along the path from the gateway to that node using a revised version of the Conflict Free Coloring (CF-Coloring); see [13]. Briefly, given a path p, our new coloring method, Collision Free Link Upgrade Coloring (CFLU-Coloring), assigns an increasing positive integer to each link along p starting from the source node. The novelty of CFLU-Coloring is its consideration for the link upgrade and MTR capability of nodes. In particular, for a given node u with Δ_u antennas, each color, represented by a natural number, can be used Δ_u times with the constraint that no incoming and outgoing links use the same set of colors, i.e., *no-tx-rx*. This ensures the final schedule is conflict-free. For example, suppose we have two paths $p_1 = A \rightarrow B \rightarrow C \rightarrow D$, where A is the source and D is the destination and $p_2 = A \rightarrow B \rightarrow E$. CFLU-Coloring will assign link $e_{A,B}$ with the number 1, link $e_{B,C}$ with 2, and 3 to link $e_{C,D}$ for p_1. We assume each node u has $\Delta_u = 2$ antennas, then color 1 is assigned to $e_{A,B} \in p_2$ and 2 is assigned to $e_{B,E} \in p_2$. An instance of CFLU-Coloring on path p is a tuple $< (e_{1,2}, t_1), (e_{2,3}, t_2) \dots (e_{i,j}, t_i) >$, where $e_{i,j}$ denotes a link from node i to j and t_i is the transmission slot. The number assigned to each link along path p is strictly increasing, that is $t_1 < t_2 < \cdots < t_i$. The maximum number of colors used in CFLU-Coloring is the makespan of the personalized broadcast schedule.

We now describe Algo-PB in detail with the aid of the pseudocode presented in Algorithm 1. To do this, we need a few notations and definitions. Define the distance of a node $u \in V$ to be the number of hops via the shortest path from the gateway to u. Let V' be a tuple to store nodes in decreasing distance order. Let $Sched(p)$ be a tuple containing the output from applying CFLU-Coloring to p. Denote P as a collection of paths, and F is a set that records all results from applying CFLU-Coloring; i.e., $\{Sched(p) \mid p \in P\}$. Let t_{max} be the maximum slot used in CFLU-Coloring; i.e., it is the makespan of the personalized broadcast schedule.

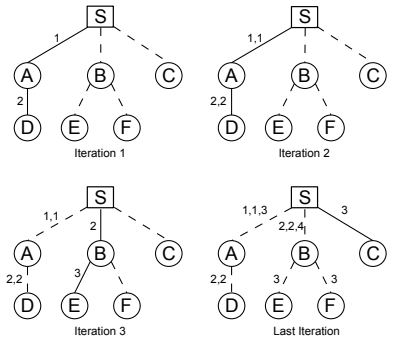
Algo-PB first sorts all nodes in V and stores them in V' in decreasing distance order; see Line 1. Then Algo-PB sets a counter k to label each path and its schedule; see Line 2. Algo-PB works in rounds. In each round, Algo-PB picks the first node in V', say v, and checks whether $r_v = 0$; see Line $4-5$. In our discussion, we will always use v to denote the first node in V' in each round. If $r_v = 0$, Algo-PB removes v from V' and moves to the next round; see Line $5-7$. If $r_v \neq 0$, Algo-PB schedules one packet from the gateway to node v along the path p_k using CFLU-Coloring; see Line $9-10$. Then Algo-PB reduces r_v by one as it has scheduled one packet to v; see Line 11. The scheduling result is a tuple $Sched(p_k)$ and Algo-PB adds the result to F. After increasing k by one, Algo-PB moves to the next round; see Line $12-13$. When $V' = \emptyset$, meaning all packets are scheduled, the set F contains all links and their scheduled slots. Lastly, Algo-PB returns t_{max} and F; see Line 16.

Algorithm 1: Algo-PB

```
Input  : V, r_u u ∈ V_s
Output: Schedule F, Schedule Length t_max
// Sort nodes in V in decreasing hop
   length order and store them in tuple
   V'
1  V' = Sort(V);
// Label each path and its schedule
2  k = 1;
3  while V' ≠ ∅ do
4      v = V'(1);
5      if r_v = 0 then
6          V' = V' − v;
7          continue;
8      else
9          p_k = path(s → v);
10         Sched(p_k) = CFLUColor(p_k);
11         r_v = r_v − 1;
12         F = F ∪ Sched(p_k);
13         k = k + 1;
14     end
15 end
// Function makespan(F) will return
   maximum t ∈ F
16 t_max = makespan(F);
```

We now show how Algo-PB works using the tree shown in Figure 4, where S is the gateway and each node u has $\Delta_u = 2$ antennas. We have the node set $V = \{A, B, C, D, E, F\}$. Suppose $r_D = 2$ and $r_A = r_B = r_C = r_E = r_F = 1$. Algo-PB first calculates the distance from the gateway S to each node and sorts nodes in V in decreasing hop length order. We then have $V' =< D, E, F, A, B, C >$, see Line 1. The first three and the last round of Algo-PB is shown in Figure 4. Solid lines denote the path picked in that round and dashed lines denote other links, both scheduled and unscheduled. Initially, the counter k is set to 1; see Line 2. In the first round, D is the first node in V', thus $v = D$. As $r_D = 2$, Algo-PB then schedules a packet to D; see Line $8-14$. The path from the gateway S to node D is $p_1 = S \rightarrow A \rightarrow D$; see Line 9. Algo-PB then schedules p_1 using CFLU-Coloring. As p_1 is the first path to be scheduled, i.e., no color is used yet, the smallest positive available number is 1. Algo-PB colors link

$e_{S,A}$ as 1 and link $e_{A,D}$ as 2; see Line 10. The result of CFLU-Coloring is $Sched(p_1) =< (e_{S,A}, 1), (e_{A,D}, 2) >$. Given that Algo-PB has scheduled one packet to D, it reduces r_D by one; see Line 11. It then adds $Sched(p_1)$ into F and increases k by one; see Line $12 - 13$. In the following rounds, paths $p_2 = S \to A \to D$, $p_3 = S \to B \to E$, $p_4 = S \to B \to F$, $p_5 = S \to A$, $p_6 = S \to B$ and $p_7 = S \to C$ are scheduled respectively. The result of CFLU-Coloring for each round is $Sched(p_2) =< (e_{S,A}, 1), (e_{A,D}, 2) >$, $Sched(p_3) =< (e_{S,B}, 2), (e_{B,E}, 3) >$, $Sched(p_4) =< (e_{S,B}, 2), (e_{B,F}, 3) >$, $Sched(p_5) =< (e_{S,A}, 3) >$, $Sched(p_6) =< (e_{S,B}, 4) >$ and $Sched(p_7) =< (e_{S,C}, 3) >$. After scheduling all packets, Algo-PB sets $t_{max} = makespan(F) = 4$ as the schedule length, see Line 16.



Slot 1	Slot 2	Slot 3	Slot 4
$e_{S,A}$	$e_{A,D}$	$e_{S,A}$	$e_{S,B}$
$e_{S,A}$	$e_{A,D}$	$e_{S,C}$	
	$e_{S,B}$	$e_{B,E}$	
	$e_{S,B}$	$e_{B,F}$	

Fig. 4: Example topology and link schedule

V. Evaluation

We evaluate the performance of Algo-PB in Matlab and the Matgraph [14] toolkit. We assume all nodes are static and randomly placed on a $100m \times 100m$ square area. If two nodes are placed within the transmission range of each other, which is $25m$, they are considered to be neighbors. The gateway node is placed at the center of the square area. We compare Algo-PB with the theoretical lower bound derived in Section III and the state-of-the art link scheduler proposed in [5]. To ensure a fair comparison, we have modified Bermond et al.'s algorithm, which we refer to as *ScheTree-MTR*, to include MTR capability, which means in each time slot, a node is now able to transmit a packet to Δ neighbors. We also show the results for Bermond et al.'s algorithm without link upgrade, which we refer to as *ScheTree*.

Our results are an average of 20 simulation runs. For each simulation run, we use a different topology. We generate the routing tree by performing a BFS at the gateway. We record the schedule length of all algorithms and compute the theoretical lower bound of each topology. We plot the confidence interval of 20 simulation runs, where 95% of the results are within the indicated error bar.

A. Node Weight

We first study the impact of different node weights, i.e., r_u for each node u. We fix the network size to 30 and each node has $\Delta_u = 3$. We conduct six group of simulations by varying the weight of each node from 1 to 6. Note, we assume all nodes have the same weight in each simulation group, i.e., in the first group, the weight of each node is one. From Figure 5, the difference between Algo-PB and the theoretical lower bound is within 4%. Our algorithm outperforms ScheTree-MTR by generating superframe lengths that is at most 19% shorter. This is because Algo-PB always preferentially schedules packets to the node farthest from the gateway and does not switch to another node until a node is fully serviced. Thus our algorithm makes the best use of the link-upgrade capability and results in a shorter schedule length. We note the superframe generated by Algo-PB is 66% shorter than ScheTree. This is because ScheTree does not exploit either MTR or link upgrade. Thus in each slot, nodes using ScheTree are only able to transmit one packet.

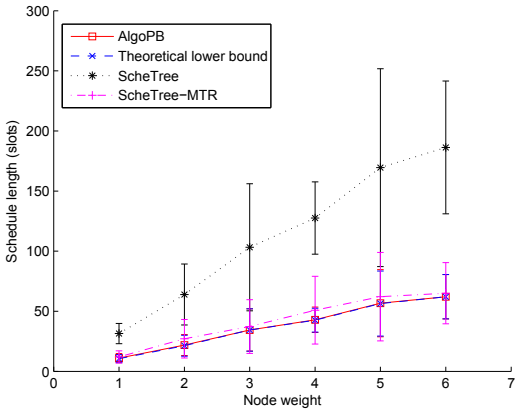


Fig. 5: Schedule length under different node weights

B. Node Density

We now study the effect of node density. We assume each node has three antennas and has a weight of three. We run 14 groups of simulations by varying the network size from 5 to 70, with a step size of 5. Figure 6 shows the resulting schedule length. Simulation results show that the schedule length increases with network size. Algo-PB outperforms ScheTree-MTR and ScheTree with at most 20% and 66% shorter schedule lengths respectively. The difference between Algo-PB and the theoretical lower bound is at most 0.7%.

C. Number of Antennas

Lastly, we study the effect of the number of antennas. We fix the network size to 30 nodes and vary the number of antennas. Figure 7(a) shows the schedule length when the weight of each node is three. On the other hand, in Figure 7(b), the weight of each node is randomly chosen from the range $[0, 5]$. When the number of antenna is one, Algo-PB and ScheTree-MTR have the same performance. When the number of antennas increases, the generated schedule length decreases. This because nodes are able to receive/forward more packets in each slot. Compared with ScheTree-MTR, Algo-PB generates

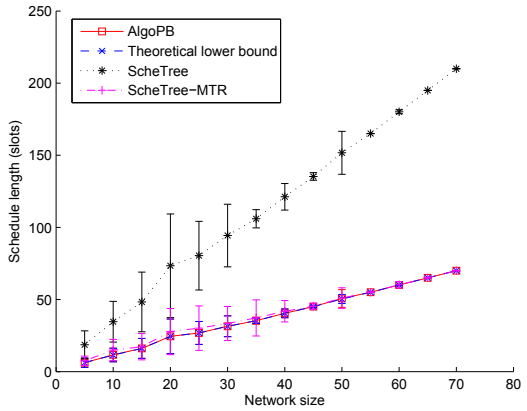
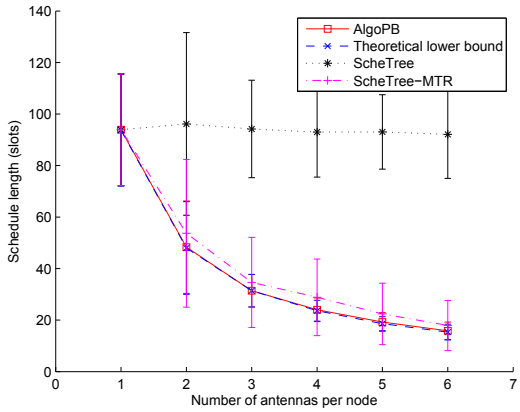
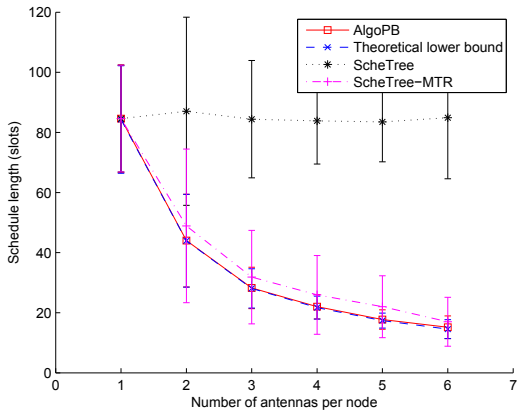


Fig. 6: Schedule lengths under different node densities



(a)



(b)

Fig. 7: Schedule length versus Δ_u

at most 17% shorter schedule lengths when all nodes have a weight of three, and 19% shorter schedule lengths when each node has a random weight. Algorithm ScheTree is only able to utilize one antenna per node in each time slot. Thus the schedule lengths generated by ScheTree do not decrease when the number of antennas increases. When each node has six antennas, Algo-PB generates 81% shorter schedule lengths than ScheTree. The difference between Algo-PB and the theoretical lower bound is at most 2%.

VI. Conclusion

We have highlighted the benefits of spatial multiplexing or link upgrade when deriving the minimal makespan for the personalized broadcast problem. We have derived new lower bounds for arbitrary tree topologies, and presented an algorithm called Algo-PB to generate the minimal length schedule for MTR WMNs. Simulation results show that the schedule length generated by Algo-PB outperforms ScheTree-MTR, a modified version of the algorithm proposed in [5], by at most 20% and the difference between Algo-PB and theoretical lower bound is at most 10%. As an immediate future work, we plan to develop approximation algorithms. This will be an interesting research direction as no such algorithms exist for the problem at hand.

References

[1] Q. H. Spencer, C. B. Peel, A. L. Swindlehurst, and M. Haardt, "An introduction to the multi-user mimo downlink," *IEEE Communications Magazine*, vol. 42, no. 10, pp. 60–67, 2004.

[2] B. Raman and K. Chebrolu, "Design and Evaluation of a New MAC Protocol for Long-Distance 802.11 Mesh Networks," in *ACM MOBICOM*, Cologne, Germany, Aug. 2005.

[3] V. Bonifaci, P. Korteweg, A. Marchetti-Spaccamela, and L. Stougie, "An approximation algorithm for the wireless gathering problem," *Operations Research Letters*, vol. 36, no. 5, pp. 605–608, 2008.

[4] D. Zhao and K.-W. Chin, "Approximation algorithm for data broadcasting in duty cycled wireless sensor networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2013, no. 1, pp. 1–14, 2014.

[5] J.-C. Bermond, L. Gargano, S. Perénnes, A. A. Rescigno, and U. Vaccaro, "Optimal time data gathering in wireless networks with multidirectional antennas," *Theoretical Computer Science*, vol. 509, pp. 122–139, 2013.

[6] P. Huang, L. Xiao, S. Soltani, M. W. Mutka, and N. Xi, "The evolution of mac protocols in wireless sensor networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 101–120, 2013.

[7] C. Florens, M. Franceschetti, and R. J. McEliece, "Lower bounds on data collection time in sensory networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 6, pp. 1110–1120, 2004.

[8] P. Soldati, H. Zhang, and M. Johansson, "Deadline-constrained transmission scheduling and data evacuation in wirelesshart networks," in *European Control Conference (ECC)*, Budapest, Hungary, August 2009.

[9] Z. Shen, H. Jiang, and Z. Yan, "Fast data collection in linear duty-cycled wireless sensor networks," *IEEE Trans. on Vehicular Technology*, 2014.

[10] O. D. Incel, A. Ghosh, B. Krishnamachari, and K. Chintalapudi, "Fast data collection in tree-based wireless sensor networks," *IEEE Trans. on Mobile Computing*, vol. 11, no. 1, pp. 86–99, 2012.

[11] S. C.-H. Huang, P.-J. Wan, X. Jia, H. Du, and W. Shang, "Minimum-latency broadcast scheduling in wireless ad hoc networks," in *IEEE INFOCOM*, Anchorage, USA, May 2007.

[12] S. Zhou and L. Ying, "On delay constrained multicast capacity of large-scale mobile ad-hoc networks," in *IEEE INFOCOM*, San Diego, USA, March 2010.

[13] L. Gargano and A. A. Rescigno, "Optimally fast data gathering in sensor networks," in *Mathematical Foundations of Computer Science*. Springer, 2006, pp. 399–411.

[14] E. R. Scheinerman, "Matgraph: a matlab toolbox for graph theory," *Software available at: http://www. ams. jhu. edu/˜ ers/matgraph*, 2007.