2016

On Minimizing Data Forwarding Schedule in Multi Transmit/ Receive Wireless Mesh Networks

He Wang
University of Wollongong, hw407@uowmail.edu.au

Kwan-Wu Chin
University of Wollongong, kwanwu@uow.edu.au

Sieteng Soh
Curtin University of Technology, Curtin University, s.soh@curtin.edu.au

On Minimizing Data Forwarding Schedule in Multi Transmit/Receive Wireless Mesh Networks

Abstract

A key problem in wireless mesh networks is forwarding packets to/from one or more gateways with connectivity to the Internet. In this respect, a short link schedule, which determines the transmission time of links, is critical. To date, existing link schedulers do not consider routers that incorporate advances in multiple input multiple output communications, i.e., interference cancellation and spatial multiplexing. In particular, these routers are able to transmit or receive distinct packets on all their links concurrently as well as deliver multiple packets to a neighbor simultaneously. To this end, we consider the problem of deriving a time division multiple access schedule that forwards packets to their respective destination quickly. We first consider the personalized broadcast problem, which assumes a single gateway and present Algo-PB, a solution that produces a schedule that is within 34.5% of the lower bound, and is 45.5% shorter than those computed by the state-of-the-art algorithms. We then extend the problem to consider multiple gateways. This so-called forest construction problem is modeled as an integer linear program (ILP). We then outline Algo-FC, a novel heuristic that generates a balanced forest by repeatedly picking a node from the heaviest tree and migrating it to another tree if doing so reduces the overall load. Experiment results show that the resulting forest generated by Algo-FC is within 9.1% of the ILP solution.

Publication Details

H. Wang, K. Chin & S. Soh, "On Minimizing Data Forwarding Schedule in Multi Transmit/Receive Wireless Mesh Networks," IEEE Access, vol. 4, pp. 1570-1582, 2016.

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 15, 2016, accepted April 4, 2016, date of publication April 12, 2016, date of current version April 29, 2016.

Digital Object Identifier 10.1109/ACCESS.2016.2553048

On Minimizing Data Forwarding Schedule in Multi Transmit/Receive Wireless Mesh Networks

HE WANG[1], KWAN-WU CHIN[1], AND SIETENG SOH[2], (Member, IEEE)
[1]School of Electrical, Computer and Telecommunication Engineering, University of Wollongong, Wollongong, NSW 2522, Australia
[2]Department of Computing, Curtin University of Technology, Bentley, WA 6102, Australia

Corresponding author: H. Wang (hw407@uowmail.edu.au)

ABSTRACT A key problem in wireless mesh networks is forwarding packets to/from one or more gateways with connectivity to the Internet. In this respect, a short link schedule, which determines the transmission time of links, is critical. To date, existing link schedulers do not consider routers that incorporate advances in multiple input multiple output communications, i.e., interference cancellation and spatial multiplexing. In particular, these routers are able to transmit or receive distinct packets on all their links concurrently as well as deliver multiple packets to a neighbor simultaneously. To this end, we consider the problem of deriving a time division multiple access schedule that forwards packets to their respective destination quickly. We first consider the personalized broadcast problem, which assumes a single gateway and present Algo-PB, a solution that produces a schedule that is within 34.5% of the lower bound, and is 45.5% shorter than those computed by the state-of-the-art algorithms. We then extend the problem to consider multiple gateways. This so-called forest construction problem is modeled as an integer linear program (ILP). We then outline Algo-FC, a novel heuristic that generates a balanced forest by repeatedly picking a node from the heaviest tree and migrating it to another tree if doing so reduces the overall load. Experiment results show that the resulting forest generated by Algo-FC is within 9.1% of the ILP solution.

INDEX TERMS Link scheduler, integer linear program (ILP), NP-complete, multiple gateways, balanced tree.

I. INTRODUCTION

Wireless Mesh Networks (WMNs) are ideal for use in enterprises, campuses, metro and rural areas. They serve as an easy-to-setup multi-hop wireless backbone that allows applications to upload/download data to/from one or more gateways with connectivity to the Internet. Hence, it is critical that a gateway delivers buffered packets to their respective receiver or mesh router quickly. Conversely, it is important that wireless mesh routers send buffered packets, which have been uploaded by clients, to a gateway promptly. Hence, deriving a link activation schedule that ensures high throughput is important. Figure 1(a) shows an example schedule. The nodes marked s_1 and s_2 are gateways. Assume s_1 has one packet to deliver to all non-gateway nodes. Moreover, each node can only transmit one packet at a time. We see that a Time Division Multiple Access (TDMA) schedule of five slots is used to deliver all packets.

Deriving such a schedule is equivalent to solving the personalized broadcast problem [1]. Briefly, consider an arbitrary tree rooted at a gateway. Each node has a set of packets buffered at the gateway. The aim is to derive a collision-free link schedule to transfer these packets to their receiver in the shortest possible time or makespan. We emphasize that 'personalized broadcast' refers to the fact that the packets to be delivered are unique and has a designated destination.

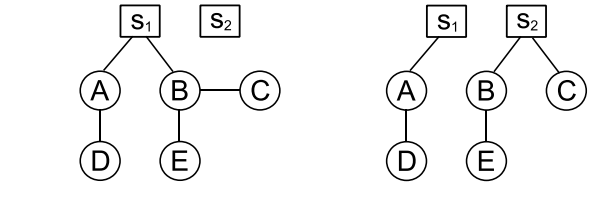

FIGURE 1. Forest and personalized broadcast schedule. (a) Single tree and schedule. (b) Multiple trees and schedule.

Fundamentally, it is different from the traditional broadcast scheduling problem [2] where all nodes receive the *same* packet from a gateway. As an aside, the authors of [1] have shown that the personalized broadcast problem is equivalent to the data collection or data gathering problem. In particular, we can ''reverse'' any schedule derived for the personalized broadcast problem and use it for data collection/gathering. That is, links activated in the first, second and subsequent slots in the personalized broadcast schedule are activated last, second last and so forth time slots [3]. Given this fact, we only need to consider deriving a link schedule for packets flowing in one direction.

In general, we need to consider multiple gateways. We call this the forest construction problem. To see the advantages of multiple gateways, reconsider Figure 1. Using both s_1 and s_2, we see that the schedule length or makespan reduces to three slots; see Figure 1(b). Observe that the time to deliver all packets is determined by the personalized broadcast schedule with the longest makespan.

To date, there are several link schedulers or Medium Access Control (MAC) protocols that aim to minimize the length of a data collection or personalized broadcast schedule [3]–[7]. In [4], the authors analyze the lower bound for line, multiline and tree topologies and derive the optimal data collection time. The authors of [5] analyze the theoretical lower bound for data collection in line and tree topologies and propose a data collection algorithm for binary tree topologies that achieves the lower bound. In [6], the authors study the fast data collection problem in duty-cycled wireless sensor networks. They first propose a centralized algorithm that achieves the optimal data collection time; i.e., in the non-duty-cycled case. They then propose a distributed algorithm to generate a minimal data collection schedule. The authors of [3] study the data gathering problem with multi-directional antennas. They give an algorithm to construct a schedule for tree topologies. Moreover, they consider nodes with more than one packet and no buffer. The authors in [7] 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. The second algorithm considers one-shot data collection, which means the generated schedule is only used once.

With regards to generating a balanced forest, we are only aware of a few works. In [8], the authors propose a distributed load balancing algorithm that ensures each gateway has equal number of nodes. The algorithm in [9] considers the average queue length and expected availability of gateways. In [10], the authors propose a distributed forest construction approach to create a ''top-load balanced forest''. This is a forest where the set of links connected to gateways have similar traffic load. We note that the forest construction problem is similar to the load-balanced routing problem [11], [12]. In [11], each gateway monitors its queue length and notifies associated nodes to look for an alternative gateway if its queue length exceeds a given threshold. The authors of [12] maximize network utilization by balancing the traffic load between gateways. They first find a set of paths that maximize the normalized bandwidth allocation of nodes. Then they employ the single-source unsplittable flow algorithm from [13] to extract one path for each node from the generated paths.

Different from these works, we consider a fundamentally different communication model; in contrast, prior works assume nodes have an omni-directional or directional antenna that only afford *one* data transmission/reception at a time. In our case, routers are able to transmit or receive from their neighbors simultaneously over the *same* frequency [14]. This ability, called multiple transmit or receive (MTR), can be achieved via Multi-User Multiple Input and Multiple Output (MIMO) [15] technology. That is, as routers have multiple antennas, due to spatial multiplexing, they can transmit multiple data streams to different neighbors [16]. Moreover, they can dedicate some streams to the same neighbor; we refer to this ability as *link upgrade*. A key constraint, however, is that a node must not transmit *and* receive concurrently, so called *no-tx-rx* or half-duplex constraint. Moreover, nodes have finite number of antennas. This means we need to consider whether to upgrade a link at the expense of reducing the number of neighbors that a router can transmit to. Lastly, we will have to use some antenna elements to null nearby interfering transmissions.

Given MTR routers, in this paper, we aim to address the personalized broadcast problem by deriving a schedule that allows nodes to transmit/receive packets to/from their neighbors. In addition, when appropriate, a node dedicates multiple antennas to boost the number of packets delivered to a neighbor. We also seek to construct a forest that minimizes the longest schedule. In a nutshell, we make the following contributions:

- We present Algo-PB, the first link scheduler that generates the minimal personalized broadcast schedule for arbitrary tree topologies constructed in an MTR WMN. Algo-PB generates up to 45.5% shorter schedule lengths as compared to using the algorithm in [3], and the difference between the schedule produced by our algorithm and the theoretical lower bound is at most 34.5%.
- We present an Integer Linear Program (ILP) for the forest construction problem. We also present a heuristic, called Algo-FC, that generates a forest by repeatedly migrating a node and all its descendants from the gateway with the highest load to another tree with a lower load. Compared with the ILP, Algo-FC has lower complexity and generates a near-optimal balanced forest. Experimental results show that when using Algo-FC, the gateway with the highest load has at most 9.1% higher load as compared to the optimal solution generated by ILP.

Note, a preliminary version of our work has appeared in [17]. This paper extends the said prior work as follows. First, it adopts a different interference model. Specifically, we consider suppressing interference caused by

a transmitter and also those from its neighbors by taking advantage of the Degree of Freedom (DoFs) or antenna elements available on each node. Second, it presents the forest construction problem, its solution, analysis and corresponding results.

Next, in Section II, we present our network model and formalize the personalized broadcast and forest construction problem. Then, in Section III, we present the lower and upper bound for any computed schedule. We present our solutions for both problems in Section IV and V, respectively. Section VI contains an extensive analysis of our solutions. Results supporting both solutions are presented and discussed in Section VII. Section VIII concludes the paper.

II. PRELIMINARIES

We represent an MTR-WMN as a graph $G(V, E)$ where V denotes the set of static vertices/nodes/routers and E denotes the set of directional links, i.e., link $e_{v,u} \in E$ denotes a link from node $v \in V$ to $u \in V$. Each node $v \in V$ has a transmission range R_t. We say the links $e_{v,u}$ and $e_{u,v}$ exist if the distance between node v and u are within R_t. We assume a node's interference range R_i is equal to its transmission range, i.e., $R_t = R_i$. In addition, we assume nodes have the channel state information (CSI) of their neighbors. This is reasonable as mesh routers are fixed and they can send out pilot symbols periodically to obtain the required CSI [18]. Each node $v \in V$ has Δ radios/antennas or Degree of Freedoms (DoFs). The number of antennas used by node v for transmissions, receptions and to cancel interference caused by neighbors is denoted as Δ_v^+, Δ_v^- and Δ_v^*, respectively. Note that the end nodes of a link must dedicate an antenna whenever it transmits or receives a packet. Let $\mathcal{I}_v$ be the set of neighbors of a transmitter v that are receiving at least one packet from their neighbor(s) except node v. Then node v must cancel any interference it causes to nodes in $\mathcal{I}_v$. This paper adopts the interference cancellation rule proposed in [19]. Specifically, for each node $u \in \mathcal{I}_v$, node v needs to use Δ_u^- antennas to cancel interference it causes to node u. We then have the following interference cancellation constraint:

Constraint 1: A transmitting node v must cancel all interference it causes to its neighboring receivers, meaning it must dedicate $\Delta_v^* = \sum_{u \in \mathcal{I}_v} \Delta_u^-$ antennas for this purpose.

We also have the following constraint that bounds the number of antennas used by a node:

Constraint 2: The total number of antenna elements that a node uses for data transmission/reception and to cancel interference from neighbors must not exceed the total number of antenna elements it has, i.e., $\Delta_v^+ + \Delta_v^- + \Delta_v^* \leq \Delta$.

The third constraint mandates that each node only transmits *or* receives; so called *no-tx-rx* requirement. Formally,

Constraint 3: $\Delta_v^+ \times \Delta_v^- = 0$.

Let $S \subset V$ be a set of nodes designated as gateways. For a given node $v \in V - S$, let $r_v \geq 0$ be the number of requests/packets to be received from one of the gateways. We assume that in the resulting schedule, each non-gateway node receives packets from only one gateway. Let T^s be the tree rooted at gateway $s \in S$ and $V_s \subset V$ be the set of descendants of gateway s. Note that V_s contains only non-gateway nodes. Formally, $w_v^s = a \times r_v + b \times dist(v, s)$ denotes the weight of node $v \in V_s$. Here $dist(v, s)$ is the distance (in hops) between node v and its gateway s, while a and b are coefficients that are set depending on the value of Δ, see Section V for a detailed discussion. Let $\mathcal{W}_s = \sum_{v \in V_s} w_v^s$ be the weight/load of gateway s. We assume time is slotted where each slot corresponds to the transmission of one packet. Here, the required synchronization can be achieved using a GPS module.

We now formally define the personalized broadcast problem and the forest construction problem.

Definition 1: For a given T^s, the *personalized broadcast problem* asks for a collision-free link schedule with the minimal makespan, called *personalized broadcast schedule*, that allows gateway s to transfer $r_v \geq 0$ packets to each destination node $v \in V_s$ subject to constraints 1, 2 and 3.

Definition 2: For a given MTR-WMN $G(V, E)$ with $|S|$ gateways, the *forest construction problem* aims to build a forest containing $|S|$ trees, i.e., a set of T^s for each gateway $s \in S$, such that the maximum makespan among all personalized broadcast schedules in G is minimized.

We remark that the personalized broadcast problem is NP-complete. The authors in [3] proved the NP-hardness of the personalized broadcast problem by reduction from the well-known Partition Problem. Specifically, the decision version addressed in [3] is as follows: given a network $G(V, E)$, integer weights $r_v \geq 0$, where $v \in V_s$, and an integer bound K, is there a routing tree in G and a multi-hop personalized broadcast schedule that transmits r_v packets from a gateway s to each node v in a collision free manner and has a makespan less than K? In their network model, only one transmission is allowed at a node at a time; i.e., $\Delta = 1$. In this paper, we generalized this NP-complete problem by considering $\Delta \geq 1$.

III. UPPER AND LOWER BOUND

In this section, we analyze the theoretical bounds of the personalized broadcast schedule for a tree rooted at s, i.e., T^s, for $s \in S$. We first focus on the upper bound. Denote L to be the number of levels or height of tree T^s, and D_l to be the total packets destined for nodes at level l of T^s, where $1 \leq l \leq L$. We assume gateways are at level zero. We then have the following proposition.

Proposition 1: The schedule length for a tree T^s is upper bounded by (i) D_1 slots, for $L = 1$, (ii) $D_1 + 2D_2$ slots, for $L = 2$, or (iii) $D_1 + 2D_2 + \sum_{l=3}^{L} l + 3(D_l - 1)$ slots, for $L \geq 3$.

Proof: The worst case is when all of the following conditions apply: (a) gateway s transmits packets on a level-by-level basis, i.e., gateway s first transmits all packets for nodes at level L, followed by those for nodes at level $L - 1$, and so forth, (b) each node can transmit only one packet at a time, and (c) any transmitting node at level $l \geq 1$ interferes with all nodes at level $l - 1$, l and $l + 1$.

We start with $L = 1$. Following (b), as the gateway s transmits one packet in each slot, we will require D_1 slots; the proposition is thus true for $L = 1$. Next, consider $L = 2$. For this case, as per (c), a transmitting node at level $l = 1$ interferes with all nodes at level 1 as well as nodes at level 2. Thus, the first packet for nodes at level 2 arrives in slot 2, the second packet arrives in slot $2 + 2 = 4$, the third packet arrives at level 2 in slot $2 + 2 + 2 = 6$, and so forth. The last packet arrives in slot $2D_2$. This means packets for nodes at level $l = 2$ require $2D_2$ slots. As the gateway uses D_1 slots to deliver packets destined for nodes at level $l = 1$, part (ii) of the proposition is thus true.

For case (iii) of the proposition, first consider $L = 3$. The first packet from gateway s arrives at level $l = 3$ in slot 3. In this case, as per (c), a transmitting node at level $l = 2$ interferes with all nodes at level 1, 2 and 3. Thus, the second packet to level $l = 3$ arrives in slot $l + 3 = 6$, the third packet arriving in slot $l + 6 = 9$ and so forth. Consequently, transferring D_3 packets to level $l = 3$ requires no more than $3D_3$ slots. In general, when a node at level l is receiving a packet from a node at level $l1$, the next packet destined to any node at level l can be arriving at a node at level $l3$ without causing interference, and thus the packet needs three additional slots to arrive at level l. Consequently, for any level $l \geq 3$, after the first packet that requires l slots, each subsequent packet arrives at level l every three slots. Thus, packet transmission to a node at level $l \geq 3$ is upper bounded by $l + 3(D_l 1)$ slots. Lastly, as D_2 and D_1 require $2D_2$ and D_1 slots, the schedule upper bound is at most $D_1 + 2D_2 + \sum_{l=3}^{L} l + 3(D_l - 1)$ slots. This proves case (iii) of the proposition. □

Our next results concern the lower bound of the personalized broadcast schedule. We use T_i^s to denote a sub-tree of T^s rooted at the i-th child of s. Further, let $\delta(v, s)$ be the makespan of the schedule used to transmit r_v packets from s to v. We then have the following propositions. Note that the propositions assume there is no neighboring interference among nodes and thus constitute the best case scenario. The lower bound in the presence of neighboring interference remains an open question.

Proposition 2: The makespan lower bound to transmit $r_v > 0$ packets from s to v for T^s is (i) $\delta(v, s) \geq dist(v, s) + 2 \times \lceil \frac{(r_v - \Delta)}{\Delta} \rceil$ slots for $dist(v, s) \geq 2$, or (ii) $\delta(v, s) \geq dist(v, s) + \lceil \frac{(r_v - \Delta)}{\Delta} \rceil$ slots, for $dist(v, s) = 1$.

Proof: First consider the case where $dist(v, s) \geq 2$. Observe that the first Δ packets from s reach v no faster than $dist(v, s)$ slots. For $r_v \leq \Delta$, we have $\lceil \frac{(r_v - \Delta)}{\Delta} \rceil = 0$, and thus the proposition is true for this case. However, for $r_v > \Delta$, we must consider the *no-tx-rx* constraint. Specifically, any node in the path from s to v can only transmit or receive up to Δ packets at a time. Thus, v receives the next Δ packets no earlier than two slots after it receives the first Δ packets. In general, in the best case, v can receive up to Δ of the remaining $r_v - \Delta$ packets every two slots. Thus, v receives the remaining $r_v - \Delta$ packets in no more than $2 \times \lceil \frac{(r_v - \Delta)}{\Delta} \rceil$ slots, giving a makespan of $dist(v, s) + 2 \times \lceil \frac{(r_v - \Delta)}{\Delta} \rceil$ slots. For case (ii), since v is only one hop away from s, it receives up to Δ packets every slot. Specifically, for $r_v \leq \Delta$, node v receives its r_v packets in $dist(v, s) = 1$ slot, while for $r_v > \Delta$, v receives all r_v packets in $dist(v, s) + \lceil \frac{(r_v - \Delta)}{\Delta} \rceil$ slots, proving the proposition for case (ii). □

Let L_i be the number of levels in sub-tree T_i^s, and D_i^l be the total number of packets to be transmitted to level l of sub-tree T_i^s. We define $\mathcal{F}_i^l$ to be the *last slot* in which a node at level l of sub-tree T_i^s receives its last packet. For example, in network shown in Figure 2, suppose node G and H on level 3 of subtree T_A^s receive their last packet at slot 4 and 5 respectively, thus $\mathcal{F}_A^3 = 5$.

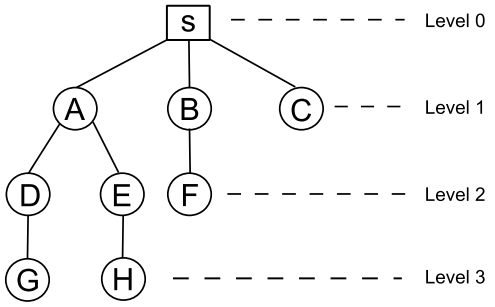

FIGURE 2. Example topology.

Proposition 3: For any node v at level l in a T_i^s and $D_i^l > 0$, (i) $\mathcal{F}_i^l = l + 2 \times \lceil \frac{D_i^l - \Delta}{\Delta} \rceil + 2 \times \lfloor \frac{\sum_{m=l+1}^{L_i} D_i^m}{\Delta} \rfloor$, for $l \geq 2$, or (ii) $\mathcal{F}_i^l = l + \lceil \frac{D_i^l - \Delta}{\Delta} \rceil + 2 \times \lfloor \frac{\sum_{m=l+1}^{L_i} D_i^m}{\Delta} \rfloor$, for $l = 1$.

Proof: For a node v at level l and the i-th child of gateway s, applying Proposition 2, we obtain (i) $\delta(v, i) \geq l + 2 \times \lceil \frac{\sum_{m=l+1}^{L_i} D_i^m}{\Delta} \rceil$ slots for $l \geq 2$, or (ii) $\delta(v, i) \geq l + \lceil \frac{\sum_{m=l+1}^{L_i} D_i^m}{\Delta} \rceil$ slots, for $l = 1$. Node v will first relay the packets destined to its descendants. This incures $2 \times \lfloor \frac{\sum_{m=l+1}^{L_i} D_i^m}{\Delta} \rfloor$ 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., $\mathcal{F}_i^l$, for level l of sub-tree 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 2 and 3, if $r_v = 0$, the number of slots for node v to receive packets located at itself is 0. Also, if $D_i^l = 0$, the number of slots needed to transfer packets to level l of sub-tree T_i^s is zero.

Proposition 4: The makespan of the personalized broadcastschedule for a sub-tree T_i^s is lower bounded by $\mathcal{F}_i = MAX(\mathcal{F}_i^l)$ for $l \in 1, \ldots, L_i$.

Proof: According to Proposition 3, any node v at level l of sub-tree T_i^s has $\mathcal{F}_i^l$. This implies that the last busy slot of sub-tree T_i^s cannot be earlier than $MAX(\mathcal{F}_i^l)$, for $l \in 1, \ldots, L_i$. □

Proposition 5: The makespan of the personalized broadcast schedule for a tree T^s with θ sub-trees is lower bounded by $MAX(\mathcal{F}_{max}, \lceil \frac{\sum_{v \in V_s} r_v}{\Delta} \rceil)$, where $\mathcal{F}_{max} = MAX(\mathcal{F}_i)$ for $i \in [1, 2, \ldots \theta]$.

Proof: Without lost of generality, the first sub-tree of T^s, i.e., T_1^s, produces $\mathcal{F}_{max}$. According to Proposition 3, 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, ... and so forth. In Proposition 4 we have shown that $\mathcal{F}_i$ 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: $\mathcal{F}_1 > \mathcal{F}_2 + \mathcal{F}_3 + \ldots + \mathcal{F}_\theta$. 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 $\mathcal{F}_1$, thus the total number of slots needed to deliver all packets to nodes in T is lower bounded by $\mathcal{F}_1$, i.e., $\mathcal{F}_{max}$.
- Case 2: $\mathcal{F}_1 \leq \mathcal{F}_2 + \mathcal{F}_3 + \ldots + \mathcal{F}_\theta$. 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 $\lceil \frac{\sum_{v \in V_s} r_v}{\Delta} \rceil$. Thus, the number of slots needed to deliver all packets will not be shorter than $\lceil \frac{\sum_{v \in V_s} r_v}{\Delta} \rceil$.

We note that the schedule makespan will not be shorter than either $\mathcal{F}_{max}$ or the number of slots needed for the gateway to inject all packets into the network, and thus we have the personalized broadcast lower bound $MAX(\mathcal{F}_{max}, \lceil \frac{\sum_{v \in V_s} r_v}{\Delta} \rceil)$. □

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

We now focus on sub-tree T_1^s. The total number of packets destined at level 3 of sub-tree T_1^s is $D_1^3 = 2$, and nodes located at level 3 do not need to forward packets as there are no nodes located beyond level 3. According to Proposition 3, we have $\mathcal{F}_1^3 = 3$. 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 $D_1^3 = 2$, nodes located at level 2 need to forward packets to level 3 $\lfloor \frac{\sum_{m=l+1}^{L_i} D_i^m}{\Delta} \rfloor$ times which incurs $2 \times \lfloor \frac{\sum_{m=l+1}^{L_i} D_i^m}{\Delta} \rfloor$ slots. We then calculate $\mathcal{F}_1^2 = 4$. Similarly, $\mathcal{F}_1^1 = 5$. According to Proposition 4, the lower bound for sub-tree T_1^s is $\mathcal{F}_1 = 5$. We apply the same procedure for sub-tree T_2^s and T_3^s and we get $\mathcal{F}_2 = 2$ and $\mathcal{F}_3 = 1$. According to Proposition 5, as $\mathcal{F}_{max} = \mathcal{F}_1 > \mathcal{F}_2 + \mathcal{F}_3$, the lower bound of tree T^s is $\mathcal{F}_{max} = 5$.

IV. PERSONALIZED BROADCAST SCHEDULING

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. Our algorithm, called Algo-PB, generates the schedule in a path-by-path manner. Its key idea is to always schedule one packet to the farthest node that has not received all its packets. It then labels each link along the path from the gateway to that node using our new version of the Conflict Free Coloring (CF-Coloring); see [20]. 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. Our novel CFLU-Coloring considers both MTR as well as link upgrade. When CFLU-Coloring colors node v's adjacent links, each color, represented by a natural number, can be used at most Δ times. The coloring result must follow constraints 1, 2 and 3. Further, the number/color assigned to each link along path p is strictly increasing to ensure the final schedule is conflict-free. Specifically, for each path p, CFLU-Coloring generates a tuple $<(e_{1,2}, c_1), (e_{2,3}, c_2) \ldots (e_{i,j}, c_i)>$, where $e_{i,j}$ denotes a link from node i to j and c_i is the color/number assigned to link $e_{i,j}$. Links assigned color c_i is scheduled in the c_i-th slot, thus the maximum number of colors used in CFLU-Coloring is the makespan of the personalized broadcast schedule.

Algorithm 1 CFLU-Color()

```
   Input  : p
   Output: Schedule Sched(p)
1  Initialization: c = 1, Sched(p) = ∅;
2  for i ← 1 to |p| − 1 do
3      while DoFCheck(p, i, c) do
4          c = c + 1 ;
5      end
6      Sched(p) = Sched(p) ∪ (e_{p(i),p(i+1)}, c);
7      c = c + 1;
8  end
```

We start by explaining the function CFLU-Coloring using Algorithm 1. Its input is a path p, and it outputs the tuple $Sched(p)$. CFLU-Coloring keeps an integer c that denotes the color CFLU-Coloring assigns to a link; see Line 1. Then CFLU-Coloring colors each link along path p, starting with the first link in p; see Line 2-8. Function $DoFCheck()$ is used to test whether assigning color c to link $e_{p(i),p(i+1)}$ satisfies constraints 1 to 3. If color c can be assigned to link $e_{p(i),p(i+1)}$, $DoFCheck()$ returns zero, otherwise $DoFCheck()$ returns one; see Line 3-5. After assigning color c to link $e_{p(i),p(i+1)}$, CFLU-Coloring updates $Sched(p)$; see Line 6. Line 7 of Algorithm 1 is used to ensure two consecutive links are not assigned the same color.

We now describe Algo-PB in detail with the aid of the pseudocode presented in Algorithm 2. To do this, we need

Algorithm 2 Algo-PB

Input : V_s, r_u *for each* $u \in V_s$
Output: Schedule F, Schedule Length t_{max}
// Sort nodes in V_s in decreasing hop length order from s, and store them in tuple $\mathcal{V}$
1 **Initialization:** $\mathcal{V} = Sort(V_s)$, $F = \emptyset$;
// Label each path and its schedule
2 $k = 1$;
3 **while** $\mathcal{V} \neq \emptyset$ **do**
4 $v = \mathcal{V}(1)$;
5 **if** $r_v = 0$ **then**
6 $\mathcal{V} = \mathcal{V} - v$;
7 continue;
8 **else**
9 $p_k = path(s \rightarrow v)$;
10 $Sched(p_k) =$ CFLUColor(p_k);
11 $r_v = r_v - 1$;
12 $F = F \cup Sched(p_k)$;
13 $k = k + 1$;
14 **end**
15 **end**
// Function *makespan*(F) will return the maximum color number c_i among all $(e_{i,j}, c_i) \in F$
16 $t_{max} = makespan(F)$;
17 return$(makespan(F), F)$;

a few notations and definitions. Let $\mathcal{V}$ be a sequence of nodes in decreasing distance order from gateway s. Let $Sched(p)$ be a tuple containing the output of CFLU-Coloring given the input p. Denote P as a collection of paths, and F is a set that records all the results from CFLU-Coloring; i.e., $F = \{Sched(p) \mid p \in P\}$. Let t_{max} be the maximum slot used in CFLU-Coloring; i.e., the makespan of the personalized broadcast schedule.

Algo-PB first sorts all nodes in V and stores them in $\mathcal{V}$ in decreasing distance order from gateway s; 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 $\mathcal{V}$, 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 $\mathcal{V}$ in each round. If $r_v = 0$, Algo-PB removes v from $\mathcal{V}$ 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 $\mathcal{V} = \emptyset$, meaning all packets are scheduled, the set F contains all links and their scheduled slots. Lastly, Algo-PB computes t_{max} and returns the resulting t_{max} and F; see Line 16 and 17 respectively.

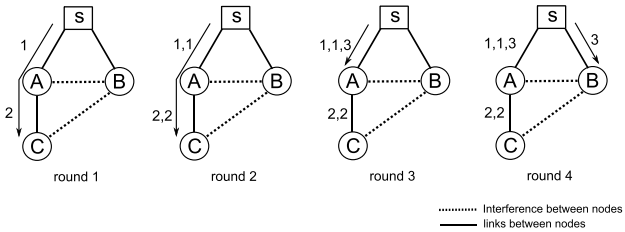

FIGURE 3. Example topology.

We show how Algo-PB works using the tree shown in Figure 3, where s is the gateway and each node has $\Delta = 2$ antennas. We have the node set $V = \{A, B, C\}$. Suppose $r_A = r_B = 1$ and $r_C = 2$. Solid lines denote links in the network and dotted lines indicate interferences. 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 $\mathcal{V} = < C, A, B >$, see Line 1. We show all four rounds of Algo-PB in Figure 3. Each arrow denotes the path picked in that round. Initially, the counter k is set to 1; see Line 2. In the first round, C is the first node in $\mathcal{V}$; thus $v = C$. As $r_C = 2$, Algo-PB then schedules a packet to C; see Line $8-14$. The path from the gateway s to node C is $p_1 = s \rightarrow A \rightarrow C$; 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 $e_{A,C}$ as 2; see Line 10. The result of CFLU-Coloring is $Sched(p_1) = < (e_{s,A}, 1), (e_{A,C}, 2) >$. Given that Algo-PB has scheduled one packet to C, it reduces r_C by one; see Line 11. It then adds $Sched(p_1)$ into F and increases k by one; see Line $12-13$. In the second round, C is still the first node in $\mathcal{V}$ and $r_C = 1$; thus Algo-PB schedules a packet to C along path $p_2 = s \rightarrow A \rightarrow C$. Assigning color 1 to link $e_{s,A}$ and color 2 to link $e_{A,C}$ satisfies constraints 1 to 3. Thus, the result of CFLU-Coloring on p_2 is $Sched(p_2) = < (e_{s,A}, 1), (e_{A,C}, 2) >$. Path $p_3 = s \rightarrow A$ is to be scheduled in the third round. Assigning color 1 to link $e_{s,A}$ does not satisfy constraint 2. This is because two of ss links are assigned color 1. Further, color 2 cannot be assigned to link $e_{s,A}$ because node A cannot transmit and receive simultaneously according to constraint 2. Thus, the result of CFLU-Coloring is $Sched(p_3) = < (e_{s,A}, 3) >$. Lastly, Algo-PB schedules a packet to node B along $p_4 = s \rightarrow B$. Color 1 cannot be assigned to link $e_{s,B}$ because it does not satisfy constraint 2. Further, Color 2 cannot be assigned to link $e_{s,B}$ either. This is because node B is interfered by transmitting node A and node A does not have enough antennas to cancel the interference it caused to node B. The result of CFLU-Coloring on p_4 is $Sched(p_4) = < (e_{s,B}, 3) >$. After scheduling all packets, Algo-PB sets $t_{max} = makespan(F) = 3$ as the schedule length, see Line 16, and returns it together with $F = < (e_{s,A}, 1), (e_{A,C}, 2), (e_{s,A}, 1), (e_{A,C}, 2), (e_{s,A}, 3), (e_{s,B}, 3) >$ in Line 17. Table 1 shows the resulting link schedule for the topology in Figure 3.

The gateway is responsible for informing nodes the latest schedule. This can be achieved by piggybacking the schedule

TABLE 1. Link schedule for Figure 3.

Slot 1	*Slot 2*	*Slot 3*
$e_{s,A}$	$e_{A,C}$	$e_{s,A}$
$e_{s,A}$	$e_{A,C}$	$e_{s,B}$

in downstream packets. Also included is the start time of the schedule in which these slots take effect. Note, if there are no downlink packets, then a dummy packet can be created before the schedule is computed. The schedule generated by Algo-PB will thus include any dummy packets. We note that Algo-PB will be used in conjunction with a protocol that schedules packets in batches. This means any new packets that arrive when a schedule is in effect will not be transmitted in the current schedule; i.e., they will be scheduled in the next batch using a newly derived schedule. Note that the evaluation of such a protocol is out-of-scope because its goal is to determine the optimal batch size that ensures queues are stable, i.e., they do not grow to infinity or ensure packets arrive before their expiration time. It is important to note that at its core, the said protocol needs our algorithm to derive a minimal makespan. We leave the evaluation of the said protocol as a future work.

V. FOREST CONSTRUCTION

Algo-PB assumes a tree or routing information is available. We now show how this can be constructed for WMNs with multiple gateways. Proposition 2 indicates the data transfer time will decrease if a node chooses to receive packets from a nearby gateway. Proposition 4 and 5 show that it is important to minimize the number of packets as well as hop count to the gateway. That is, the length of the personalized broadcast schedule will decrease if the total number of packets of the tree who has the most packets in the forest is minimized. Further, for a small Δ, say one, gateway node is able to send only one packet every two slots. Thus, the number of packets to be delivered dictates the makespan of the personalized broadcast schedule. However, for a large Δ value, i.e., each node has sufficient antennas to transmit all its packets to its descendants as well as cancel interference caused by neighbors, the personalized broadcast time is equal to the number of hops. Recall that the weight of a node v with gateway s is defined as $w_v^s = a \times r_v + b \times dist(v, s)$, where $dist(v, s)$ is the distance (in number of hops) between node v and the gateway s and r_v is the number of packets destined for node v. According to Proposition 3, when Δ is large, the personalized broadcast makespan is determined by the number of levels of the tree. Moreover, when Δ is small, the personalized broadcast makespan is affected by the number of packets buffered at each node. To this end, for a large Δ, a is set to a small value while b needs to be large. For a small Δ, a needs to be large and b is set to a small value. To simplify, we set $a = 1$ and $b = 1$ in the following sections. Lastly, recall that given the descendants of gateway s, i.e., V_s, the weight/load of gateway s is defined as $\mathcal{W}_s = \sum_{v \in V_s} w_v^s$.

A. AN ILP

We will use the binary decision variable X_v^s to indicate whether node v is a descendant of gateway s. Consequently, there are $|S|$ decision variables associated with each node v, i.e., X_v^s, where $s = 1 \ldots |S|$. Moreover, as noted before, we assume that each node can only receive packets from one gateway. Thus, each node can only belong to one gateway. Let P_v^s be the shortest path from gateway s to node v, and is represented as a set of links $\{e_{(s,i)}, e_{(i,j)}, \ldots, e_{(m,v)}\}$, where for each link $e_(i, j)$, the endpoint or node i is the parent of node j. Note, we exclude the link from gateway s to its children; e.g., link (s, i), in our ILP formulation, because the gateways are always on. Define the link set $\mathcal{L}_s = \bigcup_{v \in V_s} P_v^s$. Thus, the total load of a gateway s is $\mathcal{W}_s = \sum_{v \in V} w_v^s X_v^s$. The ILP for the problem at hand is as follows,

$$\textbf{MIN} \quad MAX\{\mathcal{W}_s | s \in S\} \tag{1}$$

Subject to:

$$\sum_{v \in V_s} X_v^s = 1, \quad \forall s \in S \tag{2}$$

$$X_a^s \geq X_b^s, \quad \forall e_{a,b} \in \mathcal{L}_s, \quad \forall s \in S \tag{3}$$

$$X_v^s \in \{0, 1\}, \quad \forall v \in V - S, \quad \forall s \in S \tag{4}$$

Constraint (2) ensures each node is connected to only one gateway. Inequality (3) means if a node b is a descendant of gateway s, its parent node a must be a descendant of the same gateway s. Constraint (4) restricts all decision variables to be binary.

In the ILP given above, each node $v \in V - S$ chooses one of the $|S|$ gateways to receive packets, thus there is $|V - S||S|$ decision variables in total. This means the number of decision variables will increase with the network size. According to Equ. (2), the number of constraints is equal to the number of non-gateway nodes $|V - S|$. The number of constraints acquired from Equ. (3) is dependent on the 'shape' of the network. In the worst case, i.e., each gateway only has one neighbor. Thus, in this case, each gateway $s \in S$, incurs $|V - S| - 1$ constraints. In total, there are $|S|(|V - S| - 1)$ constraints. According to Equ. (4), there are $|V - S||S|$ additional constraints. So in total, the number of constraints are $(2|S| + 1)|V - S| - |S|$. Thus, when the network size is large, it is computationally intractable. In the next section, we propose a greedy heuristic algorithm, called Algo-FC, to construct a routing forest.

B. ALGO-FC

Our heuristic solution is based on the well-known Breadth-First Search (BFS) algorithm. Given a network G and $|S|$ gateways, Algo-FC first creates a tree rooted at a virtual node v' where nodes in S are the virtual node's children. Then Algo-FC creates $|S|$ sub-trees, each rooted at one gateway $s \in S$ by performing a BFS of the tree rooted at v'. After removing the virtual node v', we have $|S|$ trees rooted at each gateway. Algo-FC then *balances* the weight of each gateway starting with the heaviest tree. For the heaviest tree,

say T^k, Algo-FC checks nodes level by level, starting from the top, to determine whether a node can be *migrated* to a different tree.

Node migration is required to balance the load between different gateways. Algo-FC *migrates* a node v and all its descendants from tree T^k to another tree T^m if a) node v has one neighbor that is associated to T^m, and b) the load of gateway m, $\mathcal{W}_m$, plus the weight of node v and all its descendants is less than $\mathcal{W}_k$ before migrating nodes. Here T^m is selected from the tree with lowest weight among all trees. Algo-FC *migrates* node v from T^k to T^m by removing the link between v and its parent on T^k and connects node v to another parent node that belongs to T^m. Note that node v relays packets for all its descendants. This means when node v is migrated to T^m, all its descendants receive packets from T^m.

We now describe how our proposed heuristic forest construction algorithm, Algo-FC, works in detail with the aid of Algorithm 3. Algo-FC first creates a BFS tree rooted at the virtual node v', where gateway nodes in S are the first level nodes of the BFS tree. After performing BFS and removing the virtual node, we have $|S|$ trees, each rooted at one gateway; see Line 1. Algo-FC then calculates the load of the gateway for each tree by summing up its descendants' weight; see Line $2-4$. Algo-FC works in rounds and uses a counter c to record the current tree in which Algo-FC is trying to balance. Initially, c is equal to one, which means Algo-FC starts at the heaviest tree; see Line 5. In each round, Algo-FC picks the c-th heaviest tree, denoted as T^k, and checks whether there is a node on T^k that can be migrated to another tree T^m. Algo-FC uses two variables, namely *NF* and *flag*, to indicate whether the current round is finished and whether Algo-FC successfully migrates a node in the current round, respectively; see Line $7-8$. In each round, Algo-FC checks nodes level-by-level (starting at the root's children). Algo-FC migrates a node v from T^k to T^m, if after this migration, the $\mathcal{W}_m$ is less than the $\mathcal{W}_k$; see Line $9-23$. Note, in each round, Algo-FC migrates at most one node. If Algo-FC successfully migrates a node in the current round, c is set to 1, otherwise the counter increases by one; see Line $23-28$. Algo-FC stops when no node can be migrated.

To aid our exposition, we now use Figure 4 to provide a concrete example of Algo-FC. Figure 4(a) shows a network with three gateways s_1, s_2 and s_3. Assume each node has $r_v = 1$. The first step is to create a virtual node v' and perform BFS; see Line 1. This results in three trees, rooted at s_1, s_2 and s_3 respectively; see Figure 4(b). We use T^1, T^2 and T^3 to represent the tree rooted at s_1, s_2 and s_3 respectively. Algo-FC then calculates the weight of each tree, thus we have $\mathcal{W}_1 = 11$, $\mathcal{W}_2 = 8$ and $\mathcal{W}_3 = 2$; see Line $2-4$. Initially c is set to one, meaning Algo-FC starts at the heaviest tree, in this example, T^1; see Line 5. In the first round, Algo-FC checks whether a node on T^1 can be migrated. Starting from the first level, i.e., s_1's children, Algo-FC ignores node A. This is because migrating A means we have $\mathcal{W}_2 = 19$, which is greater than $\mathcal{W}_1 = 11$ before migration. For the same reason,

Algorithm 3 Algo-FC

Input : G, S, V, r_v for each $v \in V - S$
Output: Routing trees T^s for each $s \in S$

```
// Create a BFS tree from G rooted at
   a virtual node v' where nodes in S
   are the first level nodes
1  Initialization: {T^1, T^2 ... T^|S|} = BFS(G, S, v');
2  for s = 1 to |S| do
3  |  W_s = sum_{v in T^s} w_v^s;
4  end
5  c = 1;
6  while c <= |S| - 1 do
   |  // Balance the weight between
   |     different trees
7  |  NF = 1;
8  |  flag = 1;
9  |  while NF = 1 do
10 |  |  NF = 0;
11 |  |  k = c_th heaviest tree;
   |  |  // L_k denotes the number of
   |  |     levels of the tree rooted
   |  |     at k
12 |  |  for i = 1 to L_k do
   |  |  |  // U_v denotes the descendants
   |  |  |     of v, N_i the set of nodes
   |  |  |     at level i, and T^m is
   |  |  |     another tree that has a
   |  |  |     descendant as v's neighbor
13 |  |  |  for each v in N_i do
14 |  |  |  |  if W_m + w_v^k + sum_{d in U_v} w_d^k < W_k then
   |  |  |  |  |  // Migrate node u from
   |  |  |  |  |     T^k to T^m
15 |  |  |  |  |  Migrate(v, T^k, T^m);
16 |  |  |  |  |  Update(T^k, T^m);
17 |  |  |  |  |  NF = 1;
18 |  |  |  |  |  flag = 0;
19 |  |  |  |  |  break;
20 |  |  |  |  end
21 |  |  |  end
22 |  |  end
23 |  end
24 |  if flag = 0 then
25 |  |  c = 1;
26 |  else
27 |  |  c = c + 1;
28 |  end
29 end
```

Algo-FC will not migrate node F. In addition, node D and E do not have neighboring nodes associated to either tree T^2 or T^3, and thus they cannot be migrated; see Line $9-23$. We see that in this round, Algo-FC did not migrate any node. It thus increases c by one, see Line $24-28$, meaning Algo-FC will check nodes on the second heaviest tree rooted

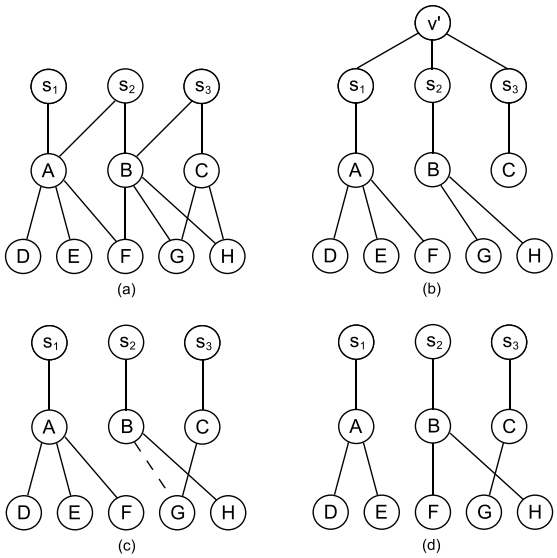

FIGURE 4. Example topology. (a) Example network. (b) After BFS. (c) Node migration. (d) Balanced forest.

at s_2 in round two. During round two, Algo-FC first checks s_2's children. Algo-FC does not migrate node B because doing so means we will have $\mathcal{W}_3 = 10$, which is larger than $\mathcal{W}_2$ before migration. However, if Algo-FC migrates node G to tree T^3, we have $\mathcal{W}_3 = 5$, which is less than $\mathcal{W}_2 = 8$. Thus, Algo-FC migrates node G to tree rooted at s_3 and the current round finishes; see Line $9-23$ and Figure 4(c). As Algo-FC successfully migrated one node in round two, the counter is reset to one; see Line $24-28$. In round three, we have $c = 1$, $\mathcal{W}_1 = 11$, $\mathcal{W}_2 = 5$ and $\mathcal{W}_3 = 5$. Algo-FC checks whether nodes on the heaviest tree T^1 can be migrated. During round three, Algo-FC migrates node F to T^2. Again as Algo-FC successfully migrates a node in round three, *counter* is reset to 1. In the following rounds, no more nodes can be migrated, thus the forest construction result of Algo-FC is shown on Figure 4(d).

VI. ANALYSIS

In this section, we discuss several properties of Algo-PB and Algo-FC. We also analyze their computational complexity.

Proposition 6: Te computational complexity of Algo-PB on a tree T^s rooted at s is $O(|V-\{s\}|(\log|V-\{s\}|+1)+\sum_{v\in V-s} r_v)$.

Proof: Referring to Algorithm 2, the time complexity of line 1 is $O(|V-\{s\}|\log|V-\{s\}|)$. According to lines $3-15$, during each iteration, Algo-PB either schedules one packet, see line $9-13$, or removes a node whose $r_v = 0$, see line $5-7$. In total, there are $\sum_{v\in V_s} r_v$ packets scheduled, which requires $\sum_{v\in V_s} r_v$ iterations. In addition, Algo-PB removes $|V_s|$ nodes with $r_v = 0$. This incurs $|V_s|$ iterations. According to line $5-7$, removing $|V_s|$ nodes has a complexity of $O(|V_s|)$. We remark that CFLU-Color has a time complexity of $O(|E|)$. In line $9-13$, scheduling $\sum_{v\in V_s} r_v$ packets has a complexity of $O(\sum_{v\in V_s} r_v \times |E|)$. Thus, in total, the computational complexity of Algo-PB is $O(|V-\{s\}|(\log|V-\{s\}|+1)+\sum_{v\in V-s} r_v \times |E|)$. □

A key computation performed by Algo-FC is balancing trees. This involves a non-negligible number of migrations. We have the following propositions.

Proposition 7: The BFS in Step 1 of Algo-FC sets each non-gateway node u to at least one tree T^s, for any gateway $s \in S$.

Proof: Step 1 of Algo-FC runs BFS from a virtual gateway v' which is the parent of all gateway nodes in S. The BFS will first connect v' to all of its one-hop neighbors, i.e., all gateway nodes in S. Since we consider only connected networks, BFS will visit each non-gateway node v and thus each node v must be a descendant of v'. In addition, each v must be a descendant of at least one gateway node $s \in S$ to be reachable from v', proving the proposition. □

Proposition 8: For an MTR-WMN that contains $|V|$ nodes and $|S|$ gateways, Algo-FC requires at most $1\times\lfloor\frac{|V|-|S|}{|S|}\rfloor + 2\times\lfloor\frac{|V|-|S|}{|S|}\rfloor + 3\times\lfloor\frac{|V|-|S|}{|S|}\rfloor\ldots+(|S|-1)\times\lfloor\frac{|V|-|S|}{|S|}\rfloor = \frac{|S|(|S|-1)}{2}\times\lfloor\frac{|V|-|S|}{|S|}\rfloor$ migrations to get a balanced forest.

Proof: To prove the above proposition, we start from the case with $|S| = 2$ gateways, say gateway 1 and 2. From Proposition 7, in the worst case, BFS sets all $|V|-2$ non-gateway nodes to only one tree, say T^1, and T^2 contains only gateway 2. For this case, Algo-FC needs to migrate $\lfloor\frac{|V|-2}{2}\rfloor$ nodes from T^1 to T^2 to balance the forest. Further, in the worst case, each migration moves only one node, i.e., migrating a leaf node, and thus it requires $\lfloor\frac{|V|-2}{2}\rfloor$ migrations to get a balanced forest, showing that the proposition is correct is correct for $|S| = 2$.

Next, consider the case with $|S| = 3$ gateways, say gateway 1, 2 and 3. In the worst case, BFS sets all non-gateway nodes to only one tree, e.g., T^1, and Algo-FC needs to migrate $2\times\lfloor\frac{|V|-3}{3}\rfloor$ nodes from T^1 to T^2 and T^3. We note that migrating $\lfloor\frac{|V|-3}{3}\rfloor$ nodes from T^1 to another tree, say T^2, requires at most $\lfloor\frac{|V|-3}{3}\rfloor$ migrations. However, in the worst case, Algo-FC may need to migrate up to $\lfloor\frac{|V|-3}{3}\rfloor$ nodes to T^2 before migrating them to T^3. Thus, migrating the nodes from T^1 to T^3 may require up to $2\times\lfloor\frac{|V|-3}{3}\rfloor$ migrations. Therefore, in total, Algo-FC requires $\lfloor\frac{|V|-3}{3}\rfloor+2\times\lfloor\frac{|V|-3}{3}\rfloor = 3\times\lfloor\frac{|V|-3}{3}\rfloor$ migrations to get a balanced forest, and thus the proposition is correct for $|S| = 3$.

Finally, consider the general case with $|S|$ gateways, say $1, 2, \ldots s$. In the worst case, all non-gateway nodes connect to one tree, e.g., T^1 after the BFS. However, to get a balanced forest, Algo-FC needs to migrate $\lfloor\frac{|V|-|S|}{|S|}\rfloor$ nodes from T^1 to each tree. Algo-FC may need to migrate up to $\lfloor\frac{|V|-|S|}{|S|}\rfloor$ nodes to tree T^{i-1} in the worst case before migrating them to T^i, where $3 \le i \le |S|$. Thus, Algo-FC requires at most $1\times\lfloor\frac{|V|-|S|}{|S|}\rfloor+2\times\lfloor\frac{|V|-|S|}{|S|}\rfloor+3\times\lfloor\frac{|V|-|S|}{|S|}\rfloor+\ldots+(|S|-1)\times\lfloor\frac{|V|-|S|}{|S|}\rfloor = \frac{|S|(|S|-1)}{2}\times\lfloor\frac{|V|-|S|}{|S|}\rfloor$ node migrations to generate a balanced forest for an MTR-WMN with $|V|$ nodes and $|S|$ gateways. □

As Proposition 8 bounds the number of migrations carried out by Algo-FC, we thus have the following corollary,

Corollary 1: Algo-FC is guaranteed to stop no later than the node migration upper bound.

VII. EVALUATION

In this section, we evaluate the performance of Algo-PB and Algo-FC in Matlab with the Matgraph [21] toolkit. Our results are an average of 50 simulation runs. For each simulation run, we use a different topology. We plot the confidence interval of 50 simulation runs, where 95% of the results are within the indicated error bar. Next, we describe our evaluation of Algo-PB before focusing on Algo-FC in Section VII-B.

A. ALGO-PB

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 the personalized broadcast schedule length generated by Algo-PB with the theoretical upper and lower bound listed in Section III as well as the link scheduler proposed in [3]. To ensure a fair comparison, we have modified Bermond et al.'s algorithm [3], which we refer to as *ScheTree*, to include MTR capability, which means in each time slot, a node v is now able to transmit up to Δ packets. We generate the routing tree by performing a BFS at the gateway. We record the schedule length of all algorithms and compute the theoretical upper and lower bound of each topology.

1) NODE DEMAND

To study the impact of different node demand, i.e., r_v for each node v, we fix the network size to 30 and each node has $\Delta = 3$. We conduct six group of experiments by varying the demand of each node from one to six. Note, we assume all nodes have the same demand 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 34.5%. Our algorithm outperforms ScheTree by generating superframe lengths that is up to 33.3% 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 of nodes to produce shorter schedules.

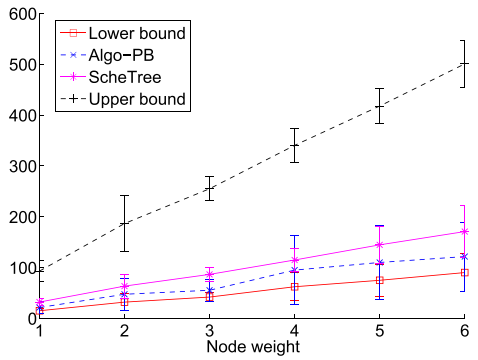

FIGURE 5. Schedule length under different node demand.

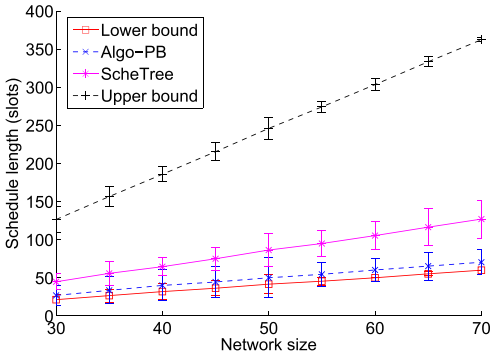

FIGURE 6. Schedule lengths under different node densities.

2) NODE DENSITY

We assume each node v has three antennas and has $r_v = 4$. We run 9 groups of experiments by varying the network size from 30 to 70, with a step size of 5. Figure 6 shows the

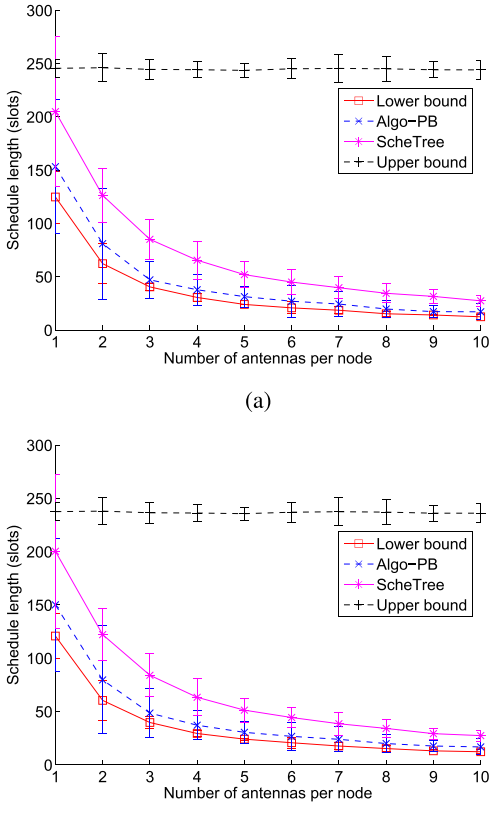

FIGURE 7. Schedule length versus Δ_u.

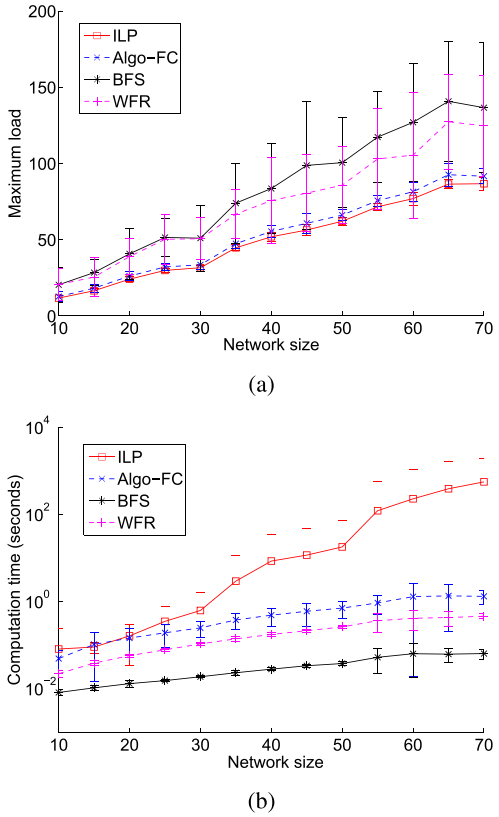

FIGURE 8. Performance under different network sizes.

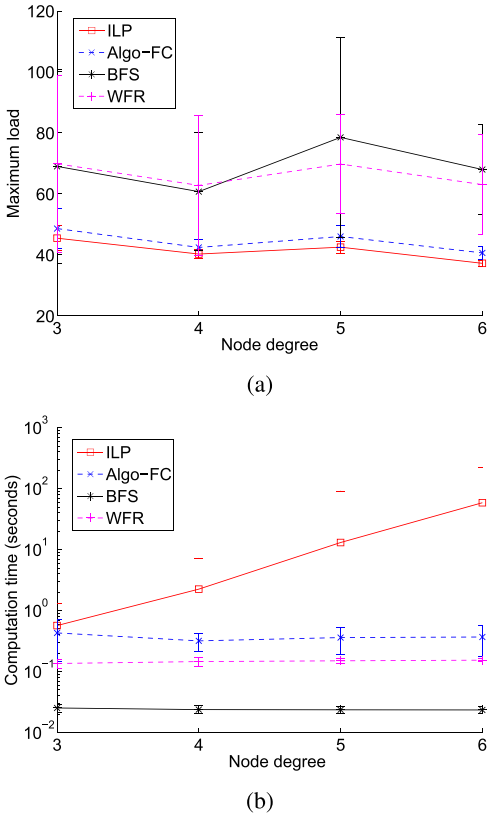

FIGURE 9. Performance under different node densities.

resulting schedule length. Simulation results show that the schedule length increases with network size. Algo-PB outperforms ScheTree by producing up to 44.6% shorter schedule. The difference between Algo-PB and the theoretical lower bound is at most 20.9%.

3) 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 antennas increases, the generated schedule length decreases. This is because nodes are able to receive/forward more packets in each slot with increasing number of antennas. Compared with ScheTree, Algo-PB generates at most 45.5% shorter schedule lengths when all nodes have a weight of three, and 44.2% shorter schedule lengths when each node has a random weight. The difference between Algo-PB and the theoretical lower bound is at most 26.5%.

B. FOREST CONSTRUCTION

We now turn our attention to Algo-FC. We assume all nodes are stationary and randomly connected. For each node, the number of requests, i.e., r_v, is randomly chosen from the range $[0, 5]$. There are three sets of evaluation. First, we vary the number of nodes from 10 to 70. We fix the number of gateways to four and node degree to four. Second, we conducted experiments with 35 nodes and four gateways. We vary the degree of each node from three to six. Third, we fix the network size to 35, node degree to four and vary the number of gateways from two to five.

We compare Algo-FC and ILP with two other solutions, breadth-first searching (BFS) and weight focus routing (WFR). Briefly, BFS and WFR work as follows:

- *BFS*. First we create a virtual node and connect it to all gateways. Then we perform a BFS from the virtual node. Finally, we remove the virtual node. This process results in multiple trees where each node is associated with the nearest gateway.
- *WFR*. It repeatedly picks the node v with the largest r_v that is not connected to any tree. It then finds a path p from node v to the nearest gateway s. WFR connects all nodes along path p to gateway s. Note that if a node n on path p is already connected to another tree rooted at gateway k, WFR connects nodes from v to n to gateway k and ignores other nodes.

In each experiment, we compute the following metrics:

- *Maximum gateway load*. This is the load of the heaviest gateway. It is calculated by summing the weight of all nodes served by a gateway.
- *Computation time*. This is the time consumed to generate the forest, measured in seconds. Matlabs optimization toolbox is used to solve the ILP.

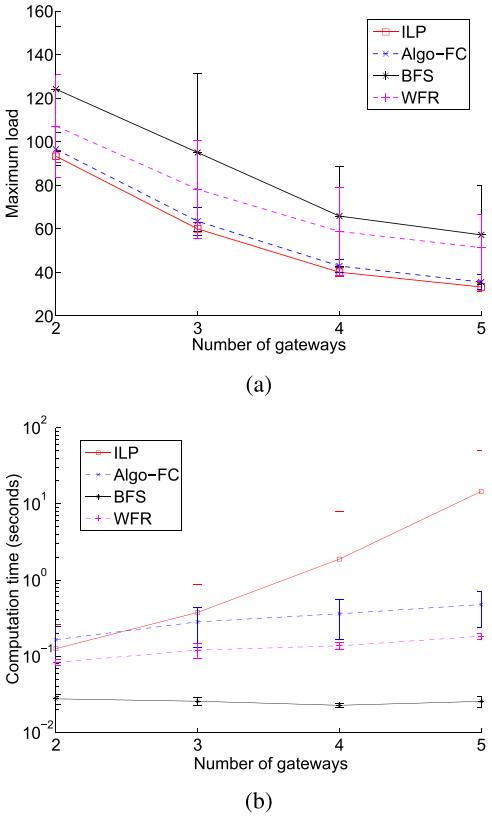

FIGURE 10. Performance under different number of gateways.

1) NETWORK SIZE

We vary the network size from 10 to 70 while fixing the number of gateways to four and node degree to four. Results in Figure 8 show that with increasing network size, the load of the heaviest gateway increases and the computation time increases. Algo-FC, outperforms BFS and WFR by at least 60.6% and 62.2%, respectively. This is because BFS only considers the shortest path and WFR only focuses on the heaviest nodes. Also, BFS and WFR do not consider the load of gateways. Compared to the ILP, Algo-FC results in the heaviest gateway having at most 8.6% higher load than the ILP. In terms of computation time, BFS has the best performance. As shown in Figure 8(b), when the network size increases, the computation time for ILP increases exponentially as expected. When the network size is 70, it takes 570.1 seconds on average for the ILP solver to get the optimal solution. However, Algo-FC only requires 1.34 seconds to get the solution when the network size is 70. Algo-FC balances the load between gateways, unlike BFS and WFR, and thus the computation time for Algo-FC is higher than BFS.

2) NODE DENSITY

Increasing node density results in a higher node degree. Thus, in this group of experiments, we fix the network size to 35 and use four gateways. Figure 9(a) shows that higher node densities tend to decrease the load of the heaviest gateway. Compared to ILP, Algo-FC results in the heaviest gateway having at most 9.1% higher load. Algo-FC outperforms BFS and WFR by at least 43.1% and 43.7%, respectively as BFS and WFR do not seek a balanced forest. In terms of computation time, Figure 9(b) shows that ILP requires the longest time as expected. When each node has six neighbors, ILP takes 58.8 seconds. The computation time increases exponentially with node degree. This is because a higher node degree leads to the ILP having an exponential increase in the number of decision variables. However, Algo-FC only needs 0.36 second under the same condition.

3) NUMBER OF GATEWAYS

Finally, we test the performance of the four approaches under different number of gateways. Figure 10(a) indicates that the load of the heaviest gateway decreases with increasing number of gateways. This is because adding more gateway nodes results in more sources that are transmitting packets. Algo-FC generates at most 6.7% higher load at the heaviest gateway as compared to ILP and outperforms BFS and WFR by 60.1% and 44.4% respectively. From Figure 10(b), on average, Algo-FC runs in 0.48 seconds when there are five gateways versus 14.6 seconds for ILP.

VIII. CONCLUSION

In this paper, we have addressed the personalized broadcast problem and the forest construction problem. We derive bounds for the personalized broadcast schedule in tree-based MTR-WMNs and propose a novel link-scheduling algorithm called Algo-PB to generate a personalized broadcast schedule with minimal makespan. We also formulate the forest construction problem using an ILP and propose a heuristic algorithm called Algo-FC to generate the routing forest. Through comprehensive experiments we show that Algo-PB outperforms current algorithms and is within 34.5% of the theoretical lower bound. Also, compared to the optimal solution generated by the ILP, the schedule makespan produced by Algo-FC is at most 9.1% longer. As a future work, we plan to investigate the personalized broadcast problem together with the data collection problem [1] in full-duplex wireless networks.

REFERENCES

[1] V. Bonifaci, P. Korteweg, A. Marchetti-Spaccamela, and L. Stougie, "An approximation algorithm for the wireless gathering problem," *Oper. Res. Lett.*, vol. 36, no. 5, pp. 605–608, Sep. 2008.

[2] D. Zhao, K.-W. Chin, and R. Raad, "Approximation algorithms for broadcasting in duty cycled wireless sensor networks," *Wireless Netw.*, vol. 20, no. 8, pp. 2219–2236, Nov. 2014.

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

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

[5] P. Soldati, H. Zhang, and M. Johansson, "Deadline-constrained transmission scheduling and data evacuation in WirelessHART networks," in *Proc. Eur. Control Conf. (ECC)*, Budapest, Hungary, Aug. 2009, pp. 4320–4325.

[6] Z. Shen, H. Jiang, and Z. Yan, "Fast data collection in linear duty-cycled wireless sensor networks," *IEEE Trans. Veh. Technol.*, vol. 63, no. 4, pp. 1951–1957, May 2014.

[7] O. D. Incel, A. Ghosh, B. Krishnamachari, and K. Chintalapudi, "Fast data collection in tree-based wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 11, no. 1, pp. 86–99, Jan. 2012.
[8] V. Mhatre, H. Lundgren, F. Baccelli, and C. Diot, "Joint MAC-aware routing and load balancing in mesh networks," in *Proc. ACM CoNEXT*, New York, NY, USA, Dec. 2007, Art. no. 19.
[9] S. Maurina, R. Riggio, T. Rasheed, and F. Granelli, "On tree-based routing in multi-gateway association based wireless mesh networks," in *Proc. IEEE Int. Symp. Pers., Indoor Mobile Radio Commun.*, Tokyo, Japan, Sep. 2009, pp. 1542–1546.
[10] Y.-F. Wen and F. Y.-S. Lin, "The top load balanced forest routing in mesh networks," in *Proc. IEEE Consum. Commun. Netw. Conf.*, Las Vegas, NV, USA, Jan. 2006, pp. 468–472.
[11] D. Nandiraju, L. Santhanam, N. Nandiraju, and D. P. Agrawal, "Achieving load balancing in wireless mesh networks through multiple gateways," in *Proc. IEEE Int. Conf. Mobile Adhoc Sensor Syst. (MASS)*, Vancouver, BC, Canada, Oct. 2006, pp. 807–812.
[12] Y. Bejerano, S.-J. Han, and A. Kumar, "Efficient load-balancing routing for wireless mesh networks," *Comput. Netw.*, vol. 51, no. 10, pp. 2450–2466, Jul. 2007.
[13] Y. Dinitz, N. Garg, and M. X. Goemans, "On the single-source unsplittable flow problem," *Combinatorica*, vol. 19, no. 1, pp. 17–41, Jan. 1999.
[14] H. Wang, K.-W. Chin, R. Raad, and S. Soh, "A distributed maximal link scheduler for multi Tx/Rx wireless mesh networks," *IEEE Trans. Wireless Commun.*, vol. 14, no. 1, pp. 520–531, Jan. 2015.
[15] Q. H. Spencer, C. B. Peel, A. L. Swindlehurst, and M. Haardt, "An introduction to the multi-user MIMO downlink," *IEEE Commun. Mag.*, vol. 42, no. 10, pp. 60–67, Oct. 2004.
[16] X. Ge, H. Cheng, M. Guizani, and T. Han, "5G wireless backhaul networks: Challenges and research advances," *IEEE Netw.*, vol. 28, no. 6, pp. 6–11, Nov./Dec. 2014.
[17] H. Wang, K.-W. Chin, and S. Soh, "A novel link scheduler for personalized broadcast in multi Tx/Rx wireless mesh networks," in *Proc. IEEE Int. Conf. Commun. Workshop (ICCW)*, London, U.K., Jun. 2015, pp. 532–537.
[18] S. Chu and X. Wang, "Opportunistic and cooperative spatial multiplexing in MIMO ad hoc networks," *IEEE/ACM Trans. Netw.*, vol. 18, no. 5, pp. 1610–1623, Oct. 2010.
[19] H. Zeng *et al.*, "A scheduling algorithm for MIMO DoF allocation in multi-hop networks," *IEEE Trans. Mobile Comput.*, vol. 15, no. 2, pp. 264–277, Feb. 2016.
[20] L. Gargano and A. A. Rescigno, "Optimally fast data gathering in sensor networks," in *Mathematical Foundations of Computer Science*. Berlin, Germany: Springer, 2006, pp. 399–411.
[21] E. R. Scheinerman. (2007). *Matgraph: A MATLAB Toolbox for Graph Theory*. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/19218-matgraph

HE WANG received the B.E. (Hons.) degree in telecommunications from the University of Wollongong, Australia, and Zhengzhou University, China. He is currently pursuing the Ph.D. degree with the University of Wollongong. His research interest includes media access protocol for wireless mesh networks and load balance routing in wireless mesh networks.

KWAN-WU CHIN is an Associate Professor with the University of Wollongong. He received the B.Sc. and Ph.D. (Hons.) degrees from the Curtin University of Technology, Australia. He is the vice-chancellor's commendation. After obtaining his Ph.D., he joined the Motorola Research Laboratory as a Senior Research Engineer, where he developed zero configuration home networking protocols and designed new medium access control protocols for wireless sensor networks and next generation bandwidth managers. In addition, he holds nine U.S. patents and won a major grant from DARPA. In 2004, he joined the University of Wollongong as a Senior Lecturer before his promotion to Associate Professor in 2011. He is also the Head of Postgraduate Studies with the School of Electrical, Computer and Telecommunications Engineering, and the Director of the Wireless Technologies Laboratory. His current research areas include medium access control protocols for wireless networks, resource allocation algorithms/policies for communications networks, routing protocols for delay tolerant networks, and mathematical programming. He holds four U.S. patents, and has authored more than 100 conference and journal articles.

SIETENG SOH (M'98) received the B.S. degree from the University of Wisconsin–Madison, and the M.S. and Ph.D. degrees from Louisiana State University, Baton Rouge, all in electrical engineering. From 1993 to 2000, he was a Faculty Member with Tarumanagara University, Indonesia, where he was the Director of the Research Institute from 1998 to 2000. He is currently a Senior Lecturer with the Department of Computing, Curtin University of Technology, Perth, WA. His research interests include network reliability, and parallel and distributed processing.