IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 9, 2018, accepted October 22, 2018, date of publication October 25, 2018,
date of current version November 19, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2878052

Minimizing Completion Time in Wireless Networks With In-Band Full Duplex Links

YIFEI REN[1], KWAN-WU CHIN[1], AND SIETENG SOH[2], (Member, IEEE)
[1]School of Electrical, Computer and Telecommunications Engineering, University of Wollongong, Wollongong, NSW 2522, Australia
[2]School of Electrical Engineering, Computing and Mathematical Sciences, Curtin University, Perth, WA 6102, Australia

Corresponding author: Yifei Ren (yr955@uowmail.edu.au)

ABSTRACT One approach to improve network capacity in future wireless networks is to deploy access points (APs) in a dense manner and employ a controller to manage these APs. Another innovation is to equip nodes with an in-band full-duplex (IBFD) radio, which allows them to transmit and receive simultaneously over the *same* frequency band. A key challenge, however, is interference. To this end, a link scheduler plays a critical role in ensuring the benefits of dense APs deployment and IBFD are not negated by severe interference. Henceforth, this paper proposes three centralized algorithms that aim to drain a given set of packets from links in minimum time. We have compared these algorithms against a schedule, where links transmit independently, and an algorithm that schedules links at slot boundaries. We studied the impact of varying node densities, transmission power, and signal-to-interference-plus-noise ratio thresholds on the link schedule. Our results show the overall completion time can be reduced by 68% as compared to scheduling links individually. Moreover, our algorithms reduce the completion time by 13% as compared to scheduling links on a slot-by-slot basis.

INDEX TERMS In-band full duplex transmissions, link scheduling, heuristics, network capacity, interference.

I. INTRODUCTION

Wireless Local Area Networks (WLANs) have grown significantly over the last two decades. They can now be found in enterprises, apartment complexes, and public areas, and are an indispensable communication infrastructure [1]. According to [2], around 78% of communication devices will be connected through a Wi-Fi WLAN by 2019. Moreover, as per [3]–[6], in this decade, mobile traffic will experience a significant increase in demand. In particular, in order to meet this significant traffic growth, the capacity of current WLANs will have to increase by a thousand fold [7].

An architecture to meet increasing demands from users is to deploy more Access Points (APs) and employ a controller to manage these APs [8]. An example is shown in Figure 1. The central controller is responsible for monitoring and scheduling transmissions to/from clients and APs. For example, Cisco's wireless controllers[1] can monitor and optimize the performance of thousands of APs. Critically, these controllers are responsible for both control and data plane operations [9]. To improve performance, they make use of information such as channel gains, gathered via IEEE 802.11v [10], network statistics such as packet loss and the number of associated stations. Controllers are able to set an appropriate transmission power and channel for each AP, and are aware of queue states. Moreover, as shown in [9], they are able to maximize spatial reuse by arbitrating data flows. However, due to the dense deployment of APs and finite spectrum, nodes may experience severe interference. To this end, a link scheduler is a key component of any controllers as it ensures mutually interfering links are scheduled at different times. Moreover, it is responsible for maximizing the number of concurrent links and their data rate.

Another approach to improve capacity is In-band-Full-Duplex (IBFD) technology. It allows nodes to transmit and receive data concurrently over the *same* frequency band [11]. Figure 2 shows an advantage of IBFD. Let us first consider the half-duplex case shown in Figure 2. These links correspond to those in Figure 1. We see that only two links can be activated at the same time. As indicated, the four transmissions end at time $2T$. However, with IBFD, these four links require only T time because they can be activated at the same time. Indeed, works such as [11]–[14] have shown that

[1]https://www.cisco.com/c/en/us/products/wireless/wireless-lan-controller/index.html

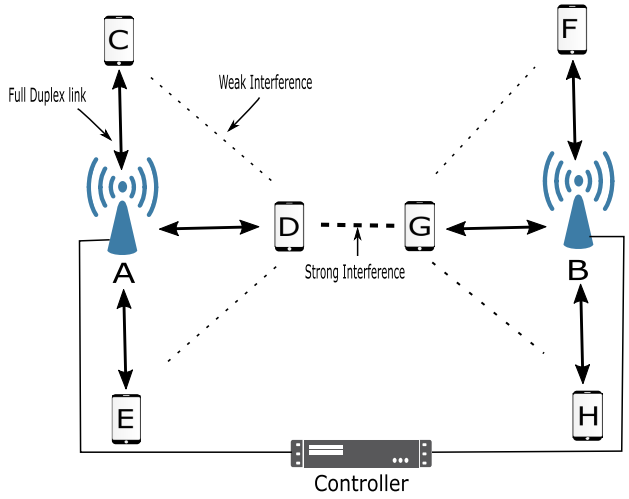


FIGURE 1. A dense WLAN with a central controller, and all devices operate on the same frequency. Also shown are full-duplex links, as indicated by double headed arrows.

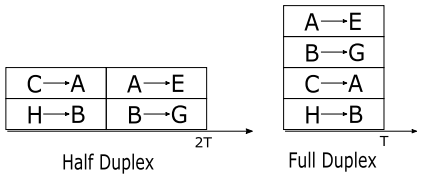


FIGURE 2. Link schedule for half-duplex versus full-duplex links.

IBFD radios increase network capacity without extra bandwidth requirement. For example, Xie and Zhang [15] showed that nodes equipped with an IBFD radio have a significant gain over the case where nodes have a half-duplex radio. Weeraddana *et al.* [16] showed that equipping nodes with an IBFD radio doubles the average sum rate and reduces network congestion significantly. Hence, it is important that future networks take advantage of IBFD radios. A key consideration is the interference caused by active links, which limits the number of links that can be activated simultaneously.

In this paper, we aim to design a link scheduler for use by the controller of a WLAN. It takes advantage of IBFD radios to minimize the transmission *completion time* in a WLAN. Here, given a set of links with pending data, transmission *completion time* is when the last link in the set completes its transmission. An example is shown in Figure 3, where we compare three schedules. Also shown is the completion time for each schedule. Specifically, for Schedule-0, Schedule-1 and Schedule-2, their completion time is respectively, T_e^{s0}, T_e^{s1} and T_e^{s2}. The six links correspond to those in Figure 1 and have the same amount of data to transmit, Signal to Noise Ratio (SNR) and data rate. The transmission time is indicated by the length of each rectangular box. When multiple links are active, these links may experience interference; see Figure 1. Weak interference, e.g., between clients C and D, is not sufficient to reduce the data rate of links. On the other hand, strong interference, e.g., between clients D and G, is likely

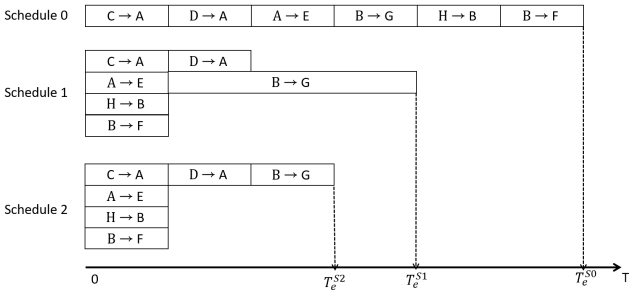


FIGURE 3. Completion time minimization example.

to cause a reduction in data rate, which leads to a longer transmission time. We can see that Schedule-0, where all links transmit one by one and ends at time T_e^{s0}, has the longest completion time. Advantageously, links do not experience interference. Schedule-1 has a shorter completion time as compared to Schedule-0 because all links use full-duplex communications. We see that the link from node B to G has to co-exist with the link from node D to A. Consequently, node B has to reduce its data rate to account for the increased interference. As a result, the link from node B to G has a long transmission time. Schedule-2 has the shortest completion time because the first four links can be activated simultaneously; this is possible because they do not interfere with each other significantly and they employ an IBFD radio. The last two links are activated independently due to strong mutual interference.

To date, a number of works have proposed full-duplex Media Access Protocols (MAC) protocols. Their overall aim is to maximize the number of activated full-duplex links. For example, the MAC in [17] modifies the IEEE 802.11 MAC header to contain a field used by nodes to negotiate their transmission mode. In [18], the proposed MAC modifies the RTS/CTS control frame of IEEE 802.11 to initiate full-duplex transmissions. The MAC protocol in [19]–[21] and [22] not only schedules full-duplex transmissions but also aims to reduce interference or improve data rates. However, none of the aforementioned MACs aim to reduce transmission completion time. Shrivastava *et al.* [9] aim to maximize spatial reuse in a WLAN. They developed a centralized approach to address the hidden and exposed terminal problems in WLANs. However, they do not consider full-duplex transmissions nor aim to minimize completion time. Apart from that, a number of works have considered scheduling links subject to Signal-to-Interference-plus-Noise Ratio (SINR) constraint. Goussevskaia *et al.* [23] introduce a concept called 'Affectedness'. They then developed a scheduling algorithm that picks links according to their affectedness. However, the algorithm is designed for arbitrary non-full-duplex capable wireless networks. Moreover, links must start and end in the same slot. Goussevskaia *et al.* [24] extend the work in [23] to consider multiple slots but they assume half-duplex communications. Goussevskaia *et al.* [23] and [24] assume all links have the same fixed transmission time of one time slot. Therefore, their algorithms only check whether it is

feasible to activate a set of links on a slot-by-slot basis and do not consider different start and end times and varying data rates.

There are only a handful of works that consider completion time. Angelakis *et al.* [25] give a general framework for solving the Minimum-Time Links Scheduling Problem (MTLSP) problem [25]. Their framework consists of two tasks: 1) select the set of links that can be activated concurrently, and 2) set their activation duration. Each task can be solved by a linear program solver. A key limitation is that concurrently activated links in each set share a common data rate and activation time. He *et al.* [26] develop a scheduling algorithm that uses the framework in [25]. Each feasible set of links must use the same data rate, as determined by the set of links in the set. Critically, in both [25] and [26], the corresponding authors did not consider full-duplex transmissions. Chi *et al.* [27] consider charging and transmission strategies in order to minimize completion time in Wireless Powered Communication Networks (WPCNs). The work in [27], however, does not consider interference from neighboring cells and focuses only on efficient use of harvested energy. In our work, we consider interference emanating from links located in other basic service sets, and our nodes have no energy constraint. Xu *et al.* [28] consider minimizing the overall transmission time. However, their algorithm is designed for routers that can receive *or* transmit to multiple neighbors at the same time; i.e., routers have no full-duplex capability. The closest work to ours is Janus [29], where its centralized Rate-Timing Allocator (RTA) algorithm aims to reduce the overall transmission completion time by pairing an uplink with a downlink. However, Janus [29] only considers scheduling links in a single, isolated cell or basic service set. It does not consider interference to/from other APs or users in nearby basic service sets.

Unlike prior works, we consider interfering APs and clients. Our *aim* is to minimize the transmission completion time of packets in a dense WLAN where nodes/stations are equipped with an IBFD radio. In this context, controllers play a critical role and are in need of a scheduler that is able to drain the queue of links quickly. Henceforth, we make the following contributions:

1) We present three novel link scheduling algorithms. Given a set of links with a number of buffered packets, the aim is to drain all packets from these links in minimum time. Moreover, once a link finishes transmission, another link is able to start transmission, assuming acceptable interference from active links. For the first algorithm, aka Algorithm-1, it only enables full-duplex transmissions whenever possible. However, Algorithm-2 utilizes full-duplex transmissions only if doing so leads to a reduction in completion time. Lastly, Algorithm-3 further improves on Algorithm-2 where it greedily finds the best SINR threshold or data rate for scheduled links.
2) We adopt for the first time the concept of 'affectedness' [23] for scheduling both half-duplex and full-duplex links. Unlike past works, e.g., [23] and [24], we consider three types of interference: 1) self, 2) cross, and 3) exogenous. Compared to [23] and [24], we also consider links with different amounts of data. Our algorithms are also able to add a set of links at any time instead of on a slot-by-slot basis. Compared to [25] and [26], we allow links to have different data rates. Lastly, these works do not consider minimizing completion time.
3) We study the impact of different node densities and transmission power levels on link schedules; both of which govern the interference experienced by nodes, and hence, their data rates or transmission times. We also consider different SINR thresholds, which affect the data rate employed by a link given its SINR value. Our results show Algorithm 1 has the second best average performance, with a reduction in completion time of around 40% as compared to having all links transmit individually. Algorithm 2 performs better than Algorithm 1 if the interference between links is strong. Algorithm 3 has the best average performance under all scenarios but incurs the longest computation time.

The remainder of this paper is structured as follows. Section II and III present the system model and problem definition, respectively. Section IV presents three algorithms and shows how they operate on a simple example. The section also presents our run time analysis of the proposed algorithms and their theoretical performance. Section V presents our evaluation methodology. After that, Section VI discusses our results. Section VII concludes the paper.

II. SYSTEM MODEL

This paper considers a set of APs, denoted as $AP = \{ap_1, ap_2, ap_3, \ldots, ap_{|AP|}\}$, and a set of clients, $\mathcal{C} = \{c_1, c_2, c_3, \ldots, c_{|\mathcal{C}|}\}$. Both APs and clients are equipped with an IBFD radio. Similar to [9], these APs are managed by a controller. Specifically, the controller is responsible for determining the transmission schedule of each AP and client. Moreover, it is aware of the queue corresponding to each link. This queue information is then used by our algorithms, which are run by the controller to determine a transmission schedule.

The set of directed links is denoted as $L = \{l_1, l_2, l_3, \ldots, l_{|L|}\}$. We also define $l_i(s, r)$, where s and r are respectively the sender and receiver of link l_i. We will write P_{wi} to denote the received power at the receiver of link l_i when the transmitter of link l_w transmits. Hence, for a given link l_i, when the transmitter of link l_i transmits, the received power at the receiver of link l_i is denoted as P_{ii}.

In order to calculate the received power, say from the transmitter of link l_a to the receiver of link l_b, i.e., P_{ab}, we use the following formula,

$$P_{ab} = P_t G_r G_t \left(\frac{c}{4\pi f d}\right)^2 \quad (1)$$

where P_t is the fixed transmission power by the transmitter of link l_a. The receive and transmit antenna gain is G_r and

TABLE 1. SINR thresholds and their corresponding data rate [30].

Thresholds (dB)	Data Rate R_i (Mbps)
$4 \leq$ SINR < 6	6
$6 \leq$ SINR < 8	9
$8 \leq$ SINR < 10	12
$10 \leq$ SINR < 12	18
$12 \leq$ SINR < 16	24
$16 \leq$ SINR < 20	36
$20 \leq$ SINR < 21	48
SINR ≥ 21	54

G_t, respectively. The Euclidean distance between the sender of link l_a and receiver of link l_b is denoted as d. The carrier frequency is f and the speed of light is c.

Each link l_i has a start and end time of $t_s(l_i)$ and $t_e(l_i)$, respectively. The transmission time of a link l_i is therefore,

$$t_c(l_i) = t_e(l_i) - t_s(l_i) = \frac{q_i}{R_i} \tag{2}$$

where q_i is an integer number representing the amount of data to be transmitted over link l_i. The symbol R_i represents the data rate; its exact value is defined later. We define S as a set or a schedule containing *valid* links. Specifically, a link is called *valid* if it satisfies the following definition:

Definition 1: A link l is **valid** if *both* of the following conditions are true: $t_s(l) \geq 0$ and $t_e(l) \geq t_s(l)$.

Let $L(t)$ be a set of *valid* links that are transmitting at time t. Specifically, given a schedule S, we have $L(t) = \{l_i \mid t_s(l_i) \leq t \leq t_e(l_i), l_i \in S\}$. In other words, the function $L(t)$ returns those links in the set S with a start and end time that overlap with time t.

For a given link l_i, its SINR is defined as,

$$SINR_i = \frac{P_{ii}}{\sum_{w \in L(t)} P_{wi} + N_o} \tag{3}$$

where the denominator comprises of the ambient noise N_o and the sum of interference from the transmitter of active links; i.e., it is the sum of received power from the transmitter of link $w \in L(t)$ to the receiver of link l_i. Recall that P_{ii} denotes the received power at the receiver of link l_i when the transmitter of link i transmits with power P_t. The data rate R_i of link l_i is dependent on its SINR; see Table 1. A link is considered collision-free if its SINR is greater than or equal to 4 dB.

A key concept used in this paper is *affectedness* [23]. As we will see later, *affectedness* is used to determine whether a set of links can transmit concurrently. Formally, the affectedness of link l_v is defined as

$$A(l_v, L(t)) = \beta \frac{\sum_{w \in L(t)} P_{wv} + N_o}{P_{vv}} \tag{4}$$

where β is the SINR threshold, N_o is the ambient noise and $\sum_{w \in L(t)} P_{wv}$ is the *total interference* from other simultaneously activated links. Note that unlike [23], our definition of *total interference* is different due to the use of IBFD radios. Specifically, the total interference suffered by a link l_v includes:

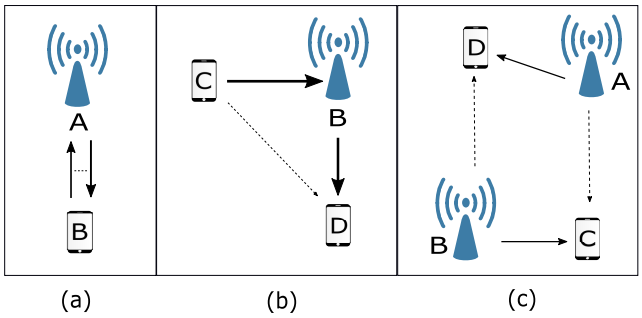


FIGURE 4. Interference scenarios: (a) self, (b) cross, (c) exogenous.

1) *Self-interference* – this occurs if another link $l_w \in L(t)$ forms a bi-directional full-duplex transmission with l_v; in Figure 4(a), we see that node A and B have formed a full-duplex link with each other. Consequently, these links interfere with one another. We assume all nodes have perfect self-interference cancellation abilities.
2) *Cross-interference* – this occurs when there is another link $l_w \in L(t)$ that forms a "relay" full-duplex transmission with l_v; from Figure 4(b), we see that the transmission from node-C may interfere with the reception at node-D. The dotted line represents cross-interference. The cross-interference from node-C to node-D is calculated by Equ. (1), with node-C as the transmitter and node-D the receiver. Then, the received power from node-C is considered as cross-interference.
3) *Exogenous* – this is the interference from active links emanating from adjacent basic service sets. In Figure 4(c), we see that the reception at node-C and node-D is respectively interfered by the transmission from node-A and node-B.

Lastly, we have the following definition:

Definition 2: A link l_v can *co-exist* with the links in the link set $L(t)$ if the condition $A(l_v, L(t)) \leq 1$ is true. Otherwise, there is too much interference for link l_v to co-exist with the links in the set $L(t)$.

III. PROBLEM DEFINITION

Our aim is to construct a schedule S containing all links in the set L, where each link in the schedule S is valid, and the transmission completion time, i.e., $t_e(l)$, of the last scheduled link l in the schedule S is minimum. Formally, our problem is

$$\min[\max\{t_e(l_i) \mid l_i \in S\}] \tag{5}$$

To illustrate the problem at hand and our notation, consider Figure 5. The schedule starts at $t = 0$. Its completion time is $\max\{t_e(l_i) \mid l_i \in \{l_1, l_2, \cdots, l_6\}\} = t_e(l_6)$. Initially, all links have an undefined start and end time. This means they do not belong to the schedule S. At time $t = 0$, several links are added into the schedule S. All these links must satisfy Definition 2. In this example, we see that links l_1, l_2, l_3 and l_4 can co-exist with each other. Hence, they have a start time of

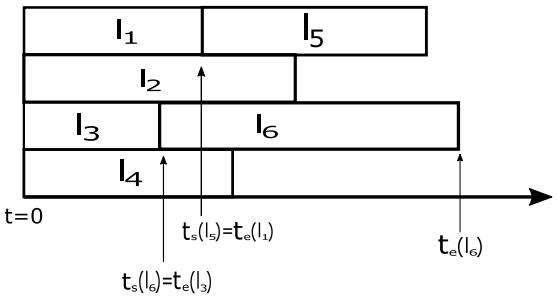


FIGURE 5. An example link schedule.

$t_s(l_i) = 0$, where $i = \{1, 2, 3, 4\}$. Assume link l_3 finishes its transmission first. At this point, there is an opportunity to add another link. However, doing so may cause the SINR or data rate of existing links to degrade. For this reason, after adding a new link, all links in the set S must continue to satisfy Definition 2. In this example, we see that link l_6 can be added successfully after link l_3 completes. Similarly, we see link l_5 is added after link l_1 has transmitted all its data. We also see that the start time of link l_5 is set to $t_s(l_5) = t_e(l_1)$. As for link l_6, its start time is $t_s(l_6) = t_e(l_3)$.

From the above example, we see that the set of links and the resulting interference affect the completion time. This is because different sets of links will yield different interference, which impact the SINR or data rate of simultaneously active links. Moreover, a set of active links may delay new links from being added due to excessive interference.

We like to note that a special case of our problem is where all transmissions complete at slot boundaries. In this case, the problem has been proven to be NP-hard; see [24]. Specifically, assume all links have the same data length and have the same data rate or SINR threshold. Then the problem is to derive a schedule with minimum length such that all links are activated once. This is exactly the NP-hard problem in [24]. However, in this paper, we consider a more general problem where links have different data lengths and a link can be added into a schedule whenever another link completes its transmission. By contrast, in [24], links can only be added at slot boundaries. That is, all links within a slot must finish transmission before another set of links start. As we will show in Section V, relaxing this restriction leads to smaller completion times.

Lastly, our work remains applicable when traffic arrives randomly, and links have different amounts of data. Let $Q_1(t)$ and $Q_2(t)$ be the queue of two APs at time t. Our goal is thus to derive a schedule that transmits packets in $Q_1(t)$ and $Q_2(t)$ in the shortest possible time. This is important because a fast completion time means a high throughput or network capacity. Once the packets in $Q_1(t)$ and $Q_2(t)$ have a transmission schedule or time, we can then consider the next batch of unscheduled packets at time $t + 1$. Note that $Q_1(t + 1)$ and $Q_2(t + 1)$ contain a random number of unscheduled packets that have arrived in the period $[t, t + 1]$ according to some traffic load distribution. We then have the same problem at time $t + 1$, which is to calculate a schedule for newly arrived packets. For this reason, in this paper, we only consider scheduling a set of links with some random amounts of data.

Algorithm 1 Maximize Concurrent Transmissions

Data: Unscheduled links set L
Result: Scheduled links set S

1 $S = L_g = L_b = \emptyset$;
2 **for** *each link* $l_i \in L$ **do**
3 **if** $SNR(l_i) \leq \beta$ **then**
4 $L_b \cup l_i$;
5 **else**
6 $L_g \cup l_i$;
7 $n = 0$;
8 **while** $L_g \neq \emptyset$ **do**
9 $\Delta = \{t_e(l_a), t_e(l_b), \cdots, t_e(l_m) \mid t_e(l_a) < t_e(l_b) < \cdots < t_e(l_m), l_a, l_b, \ldots, l_m \in S\}$;
10 $S^* = \{l_i \mid \Delta(n) \leq t_s(l_i) \leq \Delta(n+1) \vee \Delta(n) \leq t_e(l_i) \leq \Delta(n+1), l_i \in S\}$;
11 **for** *each link* $l_i \in L_g$ **do**
12 **if** ***Coexist***$(S^*, l_i, \beta) = true$ **then**
13 **AssignParams1**(l_i);
14 $S \cup l_i$;
15 $L_g \setminus l_i$;
16 **else**
17 continue;
18 $n = n + 1$;
19 return $S \cup$ **AssignParams2**(L_b);

IV. SCHEDULING ALGORITHMS

Next, we present three novel algorithms that are run at the controller of a WLAN. Their basic idea is to add one or more links into the schedule whenever a link finishes subject to links meeting their SINR requirement. The second algorithm further considers whether adding a link reduces the overall completion time. The last algorithm also identifies the best data rate for each link when it is activated along with other active links.

A. ALGORITHM 1

Algorithm 1 aims to maximize simultaneous transmissions. It takes the link set L as input. Firstly, it selects the links that have a higher SINR than their chosen threshold and includes them into the set L_g. The rationale here is that these links have the best chance of supporting full-duplex transmissions. Then, Algorithm 1 adds as many links as possible from the set L_g into the final schedule S whenever a valid link ends its transmission. Links that cannot be activated concurrently are then scheduled to transmit one after another.

Algorithm 1 operates as follows. In line-1, it initializes three empty sets: (i) S, which records the final schedule, (ii) L_g, which stores possible full-duplex links, and (iii) L_b,

which contains links capable of half-duplex transmissions only. In lines 2-6, the function SNR(l_i) calculates the SNR for each link in L; i.e., each link transmits by itself without interference. If the obtained SNR value is less or equal to a given threshold β, then L_i cannot transmit concurrently with other links and it is added into the set L_b. Otherwise, the link l_i will be added into the set L_g.

The goal of lines 8-18 is to add as many concurrent links as possible when a link finishes transmission. In line-9, the set Δ contains the end time of all links in S sorted in increasing order. The n-th element of Δ is denoted as $\Delta(n)$. The set S^* includes all active links in the time period $\Delta(n)$ to $\Delta(n+1)$. Lines 11 to 17 iterate through links in L_g. The function **Coexist**(S^*, l_i, β) determines whether the links in S^* satisfy Definition 2 after adding the link $l_i \in L_g$ into S^*. If all links satisfy Definition 2, then l_i can be added into the schedule S and function **Coexist**(S^*, l_i, β) returns true. Otherwise, the function returns false. If **Coexist**(S^*, l_i, β) is true, the function **AssignParams1**(l_i) gives the link l_i a start time of $t_s(l_i) = \Delta(n)$. Then, **AssignParams1**(l_i) determines the SINR value for link l_i according to Equ. (3) and assigns link l_i a data rate R_i according to Table 1. **AssignParams1**(l_i) also sets $t_e(l_i) = \frac{q_i}{R_i} + t_s(l_i)$ as the end time for link l_i; i.e., the end time of link l_i is its start time plus the time required to transmit q_i bits. This link is then added into the schedule S (line-14). Finally, link l_i is removed from L_g and the while loop (lines 8 to 18) continues until L_g becomes empty. Lastly, before returning the set of links in L_b, the function **AssignParams2**(L_b) assigns a start and end time, and a data rate to each link in the set L_b. These links transmit one after another at the highest possible data rate.

We will now show how Algorithm 1 generates a schedule S for the links shown in Figure 6. Clients C_1, C_2 and C_3 are connected to AP A_1, and clients C_4, C_5 and C_6 are connected to AP A_2. The dotted lines represent possible interference between clients. A thin dotted line means weak interference and a thick dotted line means strong interference. In this example, there are six links; namely $l_1(C_1, A_1)$, $l_2(C_2, A_1)$, $l_3(A_1, C_3)$, $l_4(A_2, C_4)$, $l_5(C_6, A_2)$, $l_6(A_2, C_5)$. To simplify our exposition, we assume that if a link suffers from weak interference, its current data rate does not change. On the other hand, if there is strong interference, its data rate drops by $\frac{2}{3}$.

Algorithm 1 will return the schedule shown in Figure 7. After the first iteration of the while loop (lines 8 to 18), the function **Coexist**(S^*, l_i, β) determines that links l_1, l_3, l_4 and l_5 can transmit concurrently. Two full-duplex transmissions are formed by links l_1 and l_3, and also links l_4 and l_5. The function **AssignParams1**(l_i) assigns a start and end time to these four links, as well as a suitable data rate based on their SINR. In this example, we assume the data rate remains the same as if the links transmit independently because we assume all clients and APs have perfect self-interference cancellation.

Then, the function **Coexist**(S^*, l_i, β) determines whether links l_2 and l_6 can transmit concurrently. When client C_2 transmits, its signal at A_1 will not be interfered by the transmission from A_2. Therefore, link l_2 is able to retain its current data rate. Unfortunately, link l_2 will cause a strong interference towards link l_6 because the receiver of link l_6 is client C_5 which is close to the transmitter of link l_2. Assume the data rate drops to $1/3$ of the current data rate. Therefore, the function **AssignParams1**(l_i) assigns link l_2 with the data rate that this link uses when it transmits by itself; i.e., no interference. Similarly the function **AssignParams1**(l_i) assigns link l_6 with $\frac{1}{3}$ of the data rate that it would have used if it transmitted independently. However, the overall completion time T_e^{S1} is still smaller than when there are no concurrent transmissions, which ends at time T_e.

Algorithm 1 uses the lowest possible SINR threshold, meaning links transmit at the lowest data rate. In some case, their overall completion time may exceed the case where they transmit independently without interference. The next algorithm overcomes this weakness.

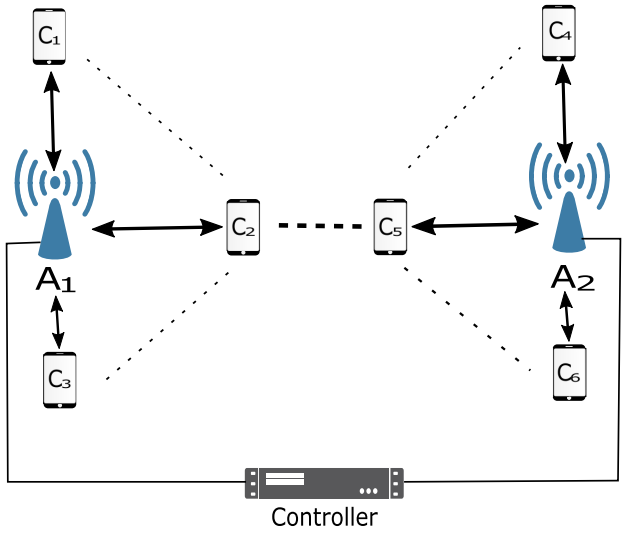


FIGURE 6. An example WLAN with multiple APs. Strong and weak interferences are denoted by thick and thin dotted lines, respectively.

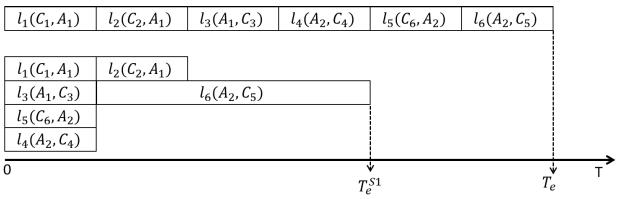


FIGURE 7. Result of Algorithm 1. As a comparison, also shown is a schedule whereby links transmit one after another; i.e., no concurrent transmissions. This schedule ends at T_e.

B. ALGORITHM 2

Algorithm 2 has two major differences from Algorithm 1. Firstly, it selects the SINR threshold β according to a base SNR value that corresponds to the case where all links transmit individually, meaning there is no interference. Secondly, it only allows multiple links to transmit concurrently if doing so reduces the overall completion time.

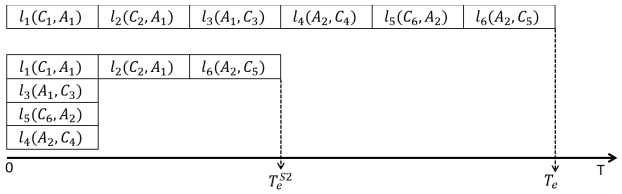


FIGURE 8. Result of Algorithm 2. This figure also shows a schedule without concurrent transmissions that ends at time T_e.

Line 2 of Algorithm 2 sets the concurrent transmission SINR threshold β to the average SNR value of all links in L. This corresponds to all links transmitting individually, where there is no interference. Then, lines 3-7 construct two sets: L_b and L_g. The set L_g comprises of all links that satisfy Definition 2. Otherwise, these links are included in the set L_b. The while loop from lines 9 to 29 aim to find sets of type a_i and the corresponding completion time reduction, denoted as t_i^*. Specifically, each a_i contains links that can be added into the schedule S at time t_n without violating Definition 2. The variable t_i^* represents how much the overall completion time can be reduced if all links in a_i transmit concurrently. The calculation is achieved by the function **RecordTime**(a_i). A larger t_i^* value means a higher reduction in completion time if all links $l_i \in a_i$ are added into the schedule S at t_n and transmit concurrently. Therefore, the function **BestCandidate(A)** returns the $a_i \in A$ that yields the highest reduction in overall completion time. Let this set be the set a_b. The function **AssignParams**1($\mathbf{a_b}$) gives each link l_i in a_b a start time $t_s(l_i) = t_n$. Then, **AssignParams**1($\mathbf{a_b}$) determines the SINR value for each link l_i in a_b according to Equ. (3). Next, the function **AssignParams**1($\mathbf{a_b}$) gives each link l_i in a_b a suitable data rate R_i according to Table 1, as well as assigns $t_e(l_i) = \frac{q_i}{R_i} + t_s(l_i)$ as the end time for each link. Finally, all links in a_b are included into the schedule S.

We will show how Algorithm 2 generates a schedule for the same example we used to illustrate Algorithm 1. Figure 8 shows that Algorithm 2 returns a different schedule. During the first iteration of its while loop, Algorithm 2 will find $a_i = \{l_1, l_3, l_5, l_4\}$ and the overall completion time can be reduced if all these links transmit concurrently. Thus, the result is the same as Algorithm 1. Then, in the second iteration, Algorithm 2 finds $a_i = \{l_2, l_6\}$. Links l_2 and l_6 can transmit concurrently but the overall completion time will not be reduced as compared to the case when link l_2 and l_6 transmit individually. Thus, Algorithm 2 rejects $a_i = \{l_2, l_6\}$ and simply lets link l_2 transmits by itself. In the next iteration, there is only link l_6. Thus, Algorithm 2 schedules link l_6 to transmit right after link l_2 finishes its transmission. Finally, we have the overall completion time $T_e^{S2} < T_e^{S1} < T_e$.

The foregone example shows that Algorithm 2 is able to find a better schedule as compared to Algorithm 1; it, however, incurs additional computation time; see Section IV-D. Also, Algorithm 2 is able to dynamically choose a SINR threshold based on the average SNR of links in L. However, this can reduce the number of concurrent transmissions when

Algorithm 2 Dynamic SINR Threshold β

Data: Unscheduled links set L
Result: Scheduled links set S
1 $S = L_g = L_b = \emptyset$;
2 $\beta = \frac{\sum_{l_i \in L} \text{SNR}(l_i)}{|L|}$;
3 **for** *each link* $l_i \in L$ **do**
4 **if** $SNR(l_i) \leq \beta$ **then**
5 $L_b \cup l_i$;
6 **else**
7 $L_g \cup l_i$;
8 $n = 0$;
9 **while** $L_g \neq \emptyset$ **do**
10 $\Delta = \{t_e(l_a), t_e(l_b), \cdots, t_e(l_m) \mid t_e(l_a) < t_e(l_b) < \cdots < t_e(l_m), l_a, l_b, \ldots, l_m \in S\}$;
11 $S^* = \{l_i \mid \Delta(n) \leq t_s(l_i) \leq \Delta(n+1) \vee \Delta(n) \leq t_e(l_i) \leq \Delta(n+1), l_i \in S\}$;
12 $L_g^* = L_g$;
13 $A = \emptyset$;
14 **while** $L_g^* \neq \emptyset$ **do**
15 $a_i = \emptyset$;
16 $t_i^* = 0$;
17 **for** *each link* $l_i \in L_g^*$ **do**
18 **if** *Coexist*$(S^*, l_i, \beta) = true$ **then**
19 $a_i \cup l_i$;
20 $L_g^* \setminus l_i$;
21 **else**
22 continue;
23 $t_i^* =$ **RecordTime**(a_i);
24 $A \cup (a_i, t_i^*)$;
25 $a_b =$ **BestCandidate(A)**;
26 **AssignParams1**(a_b);
27 $S \cup a_b$;
28 $L_g \setminus a_b$;
29 $n = n + 1$;
30 return $S \cup$ **AssignParams2**(L_b);

Algorithm 2 selects a high SINR threshold, e.g., 12 dB. This situation occurs when links in L have a high SNR. To overcome this weakness, the next algorithm will iterate through all SINR thresholds recorded in Table 1 to determine whether there exists a data rate that reduces the overall completion time.

C. ALGORITHM 3

Algorithm 3 is a slightly modified version of Algorithm 2. There are only two differences as compared to Algorithm 2. Firstly, in lines 2-6, the links $l_i \in L$ are divided into L_b and L_g by the minimum SNR requirement 4 dB. Secondly, Algorithm 3 greedily searches all transmission SINR thresholds value $\beta_i \in \Omega$, where the set Ω contains the SINR thresholds shown in Table 1. This is achieved by an additional loop

outside the while loop from lines 13 to 24 in Algorithm 2. Thus, lines 14 to 24 of Algorithm-3 are carried out to find all tuples (a_i, t_i^*) under different β_i values.

Algorithm 3 Greedy Search

Data: Unscheduled links set L
Result: Scheduled links set S
1 $S = L_g = L_b = \emptyset$;
2 **for** *each link* $l_i \in L$ **do**
3 **if** $SNR(l_i) \leq 4\ dB$ **then**
4 $L_b \cup l_i$;
5 **else**
6 $L_g \cup l_i$;
7 $n = 0$;
8 **while** $L_g \neq \emptyset$ **do**
9 $\Delta = \{t_e(l_a), t_e(l_b), \cdots, t_e(l_m) \mid t_e(l_a) < t_e(l_b) < \cdots < t_e(l_m), l_a, l_b, \ldots, l_m \in S\}$;
10 $S^* = \{l_i \mid \Delta(n) \leq t_s(l_i) \leq \Delta(n+1) \vee \Delta(n) \leq t_e(l_i) \leq \Delta(n+1), l_i \in S\}$;
11 $L_g^* = L_g$;
12 $A = \emptyset$;
13 **for** *each* β_i *in* Ω **do**
14 **while** $L_g^* \neq \emptyset$ **do**
15 $a_i = \emptyset$;
16 $t_i^* = 0$;
17 **for** *each link* $l_i \in L_g^*$ **do**
18 **if** ***Coexist***(S^*, l_i, β)=*true* **then**
19 $a_i \cup l_i$;
20 $L_g^* \setminus l_i$;
21 **else**
22 continue;
23 $t_i^* =$ **RecordTime**(a_i);
24 $A \cup (a_i, t_i^*)$;
25 $a_b =$ **BestCandidate(A)**;
26 **AssignParams1**(a_b);
27 $S \cup a_b$;
28 $L_g \setminus a_b$;
29 $n = n + 1$;
30 return $S \cup$ **AssignParams2**(L_b);

We remark that Algorithm 3 will give the same schedule for the example used to illustrate Algorithm 1 and 2. However, it incurs a higher computation time because it needs to search through all possible SINR thresholds. For more complex scenarios, see Section VI, Algorithm 3 is able to find a better result as compared to both Algorithm 1 and 2.

D. ANALYSIS

The propositions to follow concern the properties of our proposed algorithms.

Proposition 1: The run time complexity of Algorithm 1 is $(\mathcal{O}|AP||L|^2)$.

Proof: Lines 2 to 6 have complexity $\mathcal{O}(|L|)$ because they check each link in L. Note that function **SNR()** has $\mathcal{O}(1)$ time complexity. The worst case for Algorithm 1 is when all links in L are included into the set L_g, i.e., $|L_b| = 0$, and when each iteration of the while loop (lines 8 to 18) only adds one link into the schedule. In iteration $m+1$, schedule S contains m links, for $m = 1, 2, \ldots, |L| - 1$. Thus, line 9 and 10 will both have a complexity of $\mathcal{O}(m)$ because they search through every link in S, or in total, over all iterations of the while loop, the two lines require $(1+2+\cdots+|L|-1)$ searches, or $O(|L|^2)$. In iteration $m+1$ of the while loop (lines 8 to 18), the for loop (lines 11 to 18) has $|L| - m$ iterations.

Function **CoExist()** has complexity $\mathcal{O}(|AP|)$ because S^* must contain $2 \times |AP| - 1$ links in the worst case in order to obtain schedule S. Function **AssignParams1()** as well as lines 14 and 15 each has a complexity of $O(1)$. Thus, lines 11 to 17 in total, over all iterations of the while loop, are iterated $\mathcal{O}(|L| - 1 + |L| - 2 + \cdots + 1)$ times. Since each iteration of the for loop takes $\mathcal{O}(|AP|)$, the complexity of lines 11 to 17 is $\mathcal{O}(|AP||L|^2)$. Finally, the overall complexity for Algorithm 1 is $\mathcal{O}(|L^2|+|L^2|+|AP||L^2|) = \mathcal{O}(|AP||L^2|)$. Note that function **AssignParams2()** has complexity $\mathcal{O}(|L_b|)$, and for this case, i.e., $|L_b| = 0$, the function takes $\mathcal{O}(1)$. □

Proposition 2: The run time complexity of Algorithm-2 is $\mathcal{O}(|AP||L|^3)$

Proof: Line 2 as well as lines 3 to 7 have a time complexity of $\mathcal{O}(|L|)$. The worst case time for Algorithm 2 is when all links in L belong to set L_g, i.e., $|L_b| = 0$ *and* each iteration of the while loop (lines 9 to 29) inserts only one link into the schedule S. The while loop (lines 9 to 29) has $|L|$ iterations, and after iteration m, for $m = 1, 2, \cdots, |L| - 1$, the schedule S contains m links. Thus, lines 10 and 11 each takes $\mathcal{O}(m)$ to search S, or in total, each line takes $\mathcal{O}(|L|^2)$. Further, the while loop (lines 14 to 24) iterates for $|L|-m$ times, and in each iteration, the for loop (lines 17 to 22) also has $|L| - m$ iterations. Similar to Algorithm 1, function **CoExist()** has a complexity of $O(|AP|)$, and lines 19 and 20 both have complexity $\mathcal{O}(1)$. Thus, the for loop (lines 17 to 22) takes $O(|AP||L|^2)$ time for each iteration of the while loop (lines 14 to 24) or in total, over all iterations of the while loop in lines 9 to 29, has $O(|AP||L|^3)$ time complexity. Function **RecordTime**(a_i) has complexity $\mathcal{O}(1)$ because a_i contains only one link in the worst case for obtaining a schedule, and line 24 has complexity $\mathcal{O}(1)$. Function **BestCandidate(A)** has complexity $\mathcal{O}(|L| - m)$ because set A contains $((|L| - m)|a_i|)$ links and each a_i contains only one link. As in Algorithm 1, function **AssignParams1()** and **AssignParams2()** for this case each has time complexity of $\mathcal{O}(1)$. The remaining lines have complexity $\mathcal{O}(1)$. Finally, Algorithm 2 has complexity $\mathcal{O}(|L^2| + |AP||L|^3 + |L^2|) = \mathcal{O}(|AP||L|^3)$ □

Proposition 3: The run time complexity of Algorithms 3 is $\mathcal{O}((|AP| + |\Omega|)|L^3|)$.

Proof: Algorithm 3 is a modified version of Algorithm 2. The only difference is that Algorithm 3 searches through all $|\Omega|$ SINR thresholds instead of using only a single SINR threshold. This means the for loop from lines 13 to 24

forces the secondary while loop from lines 14 to 24 to execute at most $|\Omega|$ times. The remaining lines are exactly the same as Algorithm 2. Thus, the run time complexity of Algorithm 3 is $\mathcal{O}(|L^2| + (|AP| + |\Omega|)|L^3|) = \mathcal{O}((|AP| + |\Omega|)|L^3|)$. □

Proposition 4: All algorithms produce a schedule S that ensures all links satisfy their SINR threshold.

Proof: In each algorithm, the function **Coexist**(S^*, l_i, β) determines whether a link l_i can be added into the schedule S when an active link l_w completes its transmission. The subset S^* contains all links that are activated at $t_e(l_w)$, which is the end time of l_w. When **Coexist**(S^*, l_i, β) returns true, all links in S^* and link l_i have an affectedness $A(l_i, S^*)$ that is less than one. According to Definition 2, all links in S^* and link l_i are able to transmit concurrently. In addition, according to Equ. (6), all links must meet their SINR threshold, i..e, $SINR_i > \beta$, when function **Coexist**(S^*, l_i, β) returns true because,

$$\frac{P_{vv}}{\sum_{w \in L(t)} P_{wv} + N_o} = SINR_i = \frac{\beta}{A(l_i, S^*)} \tag{6}$$

The value of β is always larger or equal to 4 dB, which is the minimum requirement for links to coexist. The affectedness $A(l_i, S^*)$ must be less than one because the function **Coexist**(S^*, l_i, β) returned true. Therefore, the condition $SINR_i = \frac{\beta}{A(l_i, S^*)} > \beta$ is true. Thus, all links in the schedule are able to successfully transmit because they must have a SINR value that is at least 4 dB. □

The last proposition outlines a key relationship between the time gained from scheduling concurrent transmissions and the time loss due to the increased in transmission time resulting from higher interference.

Let τ_i denote the transmission time when link i transmits by itself; i.e., this is the transmission time corresponding to the data rate used when there is no interference. Let Scheduler-0 returns a schedule where links transmit one after another by themselves. We write T_m^0 as the completion time of m links as computed by Scheduler-0. Next, consider an arbitrary scheduler referred to as Scheduler-z that schedules the same m links. Let the completion time of these m links be T_m^z. Consider the scenario where Scheduler-z activates link i concurrently with $m - 1$ links. We term τ_i as the *saved* time. That is, the completion time T_m^z is now potentially τ_i shorter with respect to T_m^0. As an example, consider two links with transmission time τ_1 and τ_2. For simplicity, assume $\tau_1 = \tau_2$. Using Scheduler-0, we have $T_2^0 = \tau_1 + \tau_2$. However, if both links are scheduled concurrently, then the completion time is τ_1. In other words, we have *saved* τ_2 time. Equivalently, we have reduced T_2^0 by τ_2 time. Let S_m^+ be the total *saved* time when m links have been added into the schedule. For example, if there are three links with transmission time $\tau_1 > \tau_2 > \tau_3$, and we schedule link-2 and link-3 to transmit concurrently with link-1, then we have $S_3^+ = \tau_2 + \tau_3$.

Let $\phi_i \geq 1$ be a multiplicative factor that indicates the increased in transmission time when a link is scheduled with another link. As an example, consider two links that are scheduled together and also interfere with each other. Then we may have $\phi_1 = 1.1$, where $\phi_i \tau_i$ means the transmission time has increased by 10%. Equivalently, $(\phi_i - 1)\tau_i$ is the extra transmission time incurred due to a lower data rate being used to combat the increased interference. We define S_m^- as the sum increased in transmission time after m links have been added into the schedule. Let S be a schedule with m links. Formally, we have,

$$S_m^- = \sum_{i \in S} (\phi_i - 1)\tau_i \tag{7}$$

We then have the following proposition.

Proposition 5: Assume Scheduler-0 and Scheduler-z select m links in the following order: $l_1, l_2, \ldots, l_m$. If Scheduler-z ensures $S_m^+ \geq S_m^-$, then we have $T_m^z \leq T_m^0$.

Proof: Initially, the schedule S of both schedulers only contains l_1. Hence, the inequality $T_1^z \leq T_1^0$ is true, where $S_1^+ = S_1^- = 0$. Assume $S_{m-1}^+ \geq S_{m-1}^-$ when Scheduler-z picks l_m. There are three cases to consider. *Case-1:* link l_m can co-exist with the links in S and the data rate of all links remains the same. Hence, we have $T_m^z < T_m^0$ because some links are scheduled concurrently with other links. Note, in this case, $S_m^+ = S_{m-1}^+ + \tau_m$ and $S^- = 0$. *Case-2:* links in S and l_m cannot co-exist with one another due to strong interference. Hence, link l_m must be scheduled to transmit independently. In this case, the inequality $T_m^z \leq T_m^0$ holds as there is no gain in saved time and scheduled links have the same data rate, meaning $S_m^+ = S_{m-1}^+$ and $S_m^- = S_{m-1}^-$. *Case-3:* in this case, all links in S suffer increased weak interference, meaning $\phi_i \geq 1$ for all $i \in S \cup l_m$. If $S_m^+ - S_m^- < 0$, then adding link l_m results in a total increase in transmission time that exceeds the saved time. Equivalently, the resulting schedule will exceed the one computed by Scheduler-0. So we must have $S_m^+ - S_m^- \geq 0$. This implies $T_m^z \leq T_m^0$, as desired. □

V. EVALUATION

Our evaluation methodology is to determine factors that influence the transmission completion time. In particular, our concerns are *not* protocol behaviours or channel errors. To this end, we have implemented all three algorithms in C#. All APs and clients are randomly placed on a square area of size 2500 m^2. Each AP and client pair consists of an up and down link. We initialize each link with a random amount of data drawn from a Gaussian distribution with a mean of 15 and variance of five at each iteration of our simulations. The unit of data length is MBytes. The input link set L contains all links sorted in an ascending order based on their data length. We study the impact of the following parameters:

1) *Node density*. This is the ratio between the number of APs and the number of clients, denoted as $\frac{|\mathcal{C}|}{|AP|}$, which ranges from one to 15 with an interval of one.
2) *Transmission power*. We study transmission power ranging from 1 to 25 mW with an interval of 1 mW.
3) *SINR threshold* β. The value of β is chosen from Table-1. This parameter is only of concern when we evaluate the performance of Algorithm 1 because Algorithm 2 and 3 choose a SINR threshold β automatically.

To benchmark our algorithms, we also created a reference algorithm, labelled as Algorithm-SDT, that models the algorithms in [23]–[25] and [26] where links are scheduled on a slot-by-slot basis. Moreover, no new links are added when a link completes its transmission. Links have the same start time. Also, these links have the same data rate. In the first two experiments, Algorithm-SDT uses a SINR threshold of $\beta = 4$ dB, meaning it is the lowest possible data rate. In the experiment reported in Section VI-C, Algorithm-SDT assigns the highest possible data rate from Table 1 that allows them to transmit simultaneously given the interference from other active links.

In the sequel, for each network topology k, for the schedule where links transmit one after another, its completion time is denoted as T_{ck}. On the other hand, for a given schedule S_i computed by Algorithm i, where $i = \{1, 2, 3, \text{SDT}\}$, its completion time is denoted by $T_c^{S_{ik}}$. In all our experiments, we collect the following metrics:

1) Average completion time reduction (Δ). That is,

$$\Delta = \frac{1}{N}\sum_{k=1}^{N}\left(1 - \frac{T_c^{S_{ik}}}{T_{ck}}\right) \tag{8}$$

The integer N is the number of tested network topologies. We set $N = 10000$.

2) Maximum completion time reduction (Δ^+). This is the maximum reduction time over all tested network topologies. It is defined as

$$\Delta^+ = \max\left(1 - \frac{T_c^{S_i}}{T_c}\right) \tag{9}$$

3) Minimum completion time reduction (Δ^-). This is the minimum reduction in completion time over all tested topologies. Specifically,

$$\Delta^- = \min\left(1 - \frac{T_c^{S_i}}{T_c}\right) \tag{10}$$

4) Average computation time. We record the running time of Algorithms 1 to 3 on an Intel i7-6700 and 16 GB RAM.

VI. RESULTS

We now present results from our experiments in scenarios with different node densities, transmission power levels, and SINR thresholds.

A. NODE DENSITY

We set the number of APs to five. The transmission range is set to 15 meters. The SINR threshold for Algorithm 1 and Algorithm-SDT is 4 dB. In Figure 9 and 10, we observe that both Δ and Δ^+ decrease when the node density increases. The reason is that our algorithms schedule multiple links to transmit concurrently. However, the maximum number of concurrent transmissions is bounded by the number of APs because each AP can only support one up and one down link at a time. In addition, the total number of links increases with node density. Therefore, the quantities Δ and Δ^+ can only decrease with node density given the higher interference experienced by links.

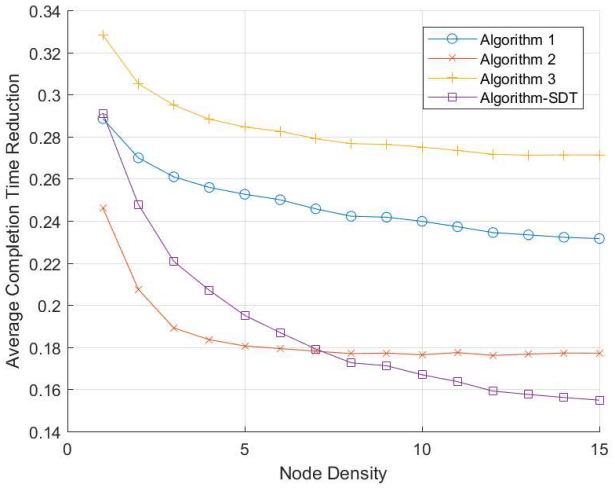


FIGURE 9. Average completion time reduction versus node density.

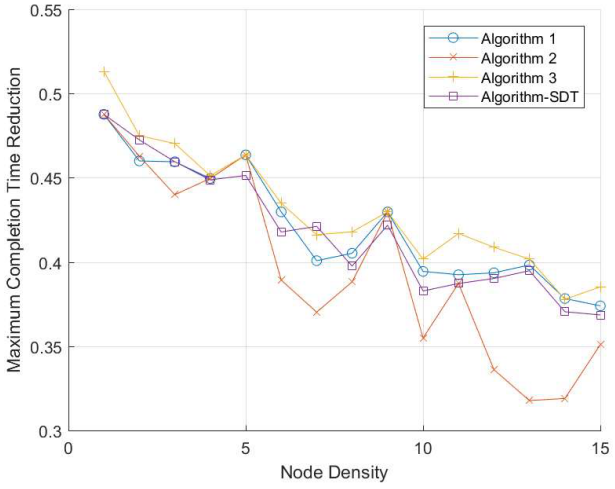


FIGURE 10. Maximum completion time reduction versus node density.

Algorithm 3 achieved the best Δ, Δ^+ and Δ^- value; see Figures 9, 10 and 11. The Δ value of Algorithm 3 is about 33% initially and reduces to about 28% when the node density is larger than five. The maximum reduction in completion time, i.e., Δ^+, of Algorithm 3 is about 51% initially and reduces to about 38% with increasing node density. The Δ^- value of Algorithm 3 fluctuates between 0.0006% and 16%. We see that Algorithm 3 has better performance than Algorithm 1 because it greedily searches through all SINR values. Algorithm 3 also only allows concurrent transmissions if the completion time of links is no longer than when they transmit one by one. Thus, we observe that the Δ^- value of Algorithm 3 does not contain any value below zero.

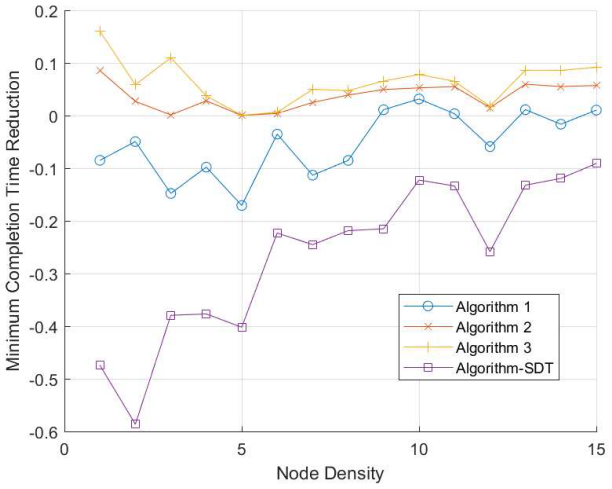


FIGURE 11. Minimum completion time reduction versus node density.

We remark that the fluctuations seen in Figure 10 and 11 are due to the use of *arbitrary* network topologies, which give rise to non-trivial interference relationships between links. As our experiments are conducted over 1000 arbitrary topologies, links may be placed far apart, meaning they do not interfere with one another significantly. Consequently, their high SINR allows them to use a high data rate. As a result, they complete their transmission quickly. On the other hand, links could be placed very closely together. Hence, they may interfere with each other significantly which means their data rate is likely to be low and they require a longer completion time.

Algorithm 1 achieves the second best average reduction in completion time or Δ value; it is about 29% initially and reduces to about 25% after the node density reaches five. Algorithm 1 has a worse Δ^+ than Algorithm 3. The maximum reduction, i.e., Δ^+, of Algorithm 1 is 49% initially and about 2.5% lower than Algorithm 3 when the node density is eleven. The reason is that Algorithm-1 uses the lowest SINR value level of 4 dB as a threshold and allows links to transmit concurrently whenever it is possible to do so. Consequently, Algorithm 1 schedules more concurrent links, which has a positive impact on both Δ and Δ^+. However, as Algorithm 1 allows links to transmit concurrently, doing so may cause a reduction in the data rate of some links. Therefore, the Δ^- value of Algorithm 1 is -8% when the node density is eight. We note that when using Algorithm-1, links scheduled to transmit together may experience significant interference. If their data rate is low, then the completion may be longer than the schedule where links transmit one by one and at the highest possible data rate. The maximum Δ^- value is 3%, which is thirteen percentage points lower than Algorithm 3.

Algorithm-SDT achieved the third best Δ when the node density is less than eight; its Δ value is 29% initially and reduces to 15% when the node density is fifteen. Algorithm-SDT achieved nearly identical Δ^+ as Algorithm 1, which recorded a reduction of 48% initially and reduces to about 36% when the node density is fifteen. The reason why Algorithm-SDT has worse performance in terms of Δ as compared to Algorithm 1 is because Algorithm-SDT assigns the same data rate and activation time to links belonging to the same subset. A link may have a high SINR but it is assigned a low data rate because other links in the subset have a low SINR. A link may also have to remain active longer than needed because other links in the same subset have not finished transmission. Both scenarios have a negative effect on reducing the overall transmission completion time. Thus, Algorithm-SDT performs worse than Algorithm 1 in terms of Δ. The same reason also causes Algorithm-SDT to have multiple Δ^- with negative values; i.e., their completion time is worse than the case where links transmit on their own. However, links can also have a similar SINR value and data rate in each subset because their parameters are generated randomly. In this situation, Algorithm-SDT can have a similar or even the same performance as Algorithm 1.

Algorithm 2 has the worst Δ when the node density is less than seven, which is 24% initially and reduces to 17% when the node density is seven. The reason is that Algorithm 2 uses the average SNR value of links in L as the SINR threshold β. The SINR threshold β chosen by Algorithm 2 will be higher than Algorithm 1 and Algorithm-SDT because they use the lowest SINR value in Table 1. Thus, Algorithm 2 allows fewer links to transmit concurrently as compared to other algorithms; this fact causes Algorithm 2 to have a longer completion time. However, the performance of Algorithm 2 is better than Algorithm-SDT in terms of Δ when the node density is larger than seven. The reason is that Algorithm-SDT may assign a link with a lower data rate than the one it can support because of other links with a low SINR in the same subset. A link may also need to have the same activation time as these links. These factors cause the overall completion time to increase and their impact becomes more pronounced with higher node densities due to the increased interference. Therefore, the performance of Algorithm-SDT is lower than Algorithm 2 when the node density reaches a high value, e.g., ten. In addition, Algorithm 2 allows concurrent transmissions when doing so reduces the overall transmission completion. Thus, we do not observe any negative Δ^- value for Algorithm 2. The Δ^- of Algorithm 2 fluctuates between 0.001% to 8%.

With increasing node density, from Figure 12, we observe that Algorithm 1 has a faster run time than others at approximately 0.02 ms initially and increases to 12 ms when the node density is fifteen. The reason is that it has the lowest run time complexity of $O(|AP||L|^2)$. Algorithm-2 has higher run time complexity and thus its run time increases faster than Algorithm-1. The run time of Algorithm-2 is about 0.11 ms initially and increases to about 40 ms when the node density is fifteen. The reason is because Algorithm-2 has a higher run time complexity of $O(|AP||L|^3)$. Algorithm-3 has the worst run time complexity, i.e., $\mathcal{O}((|AP| + |\Omega|)|L^3|)$, where its run time is recorded to be at 0.10 ms initially but increases

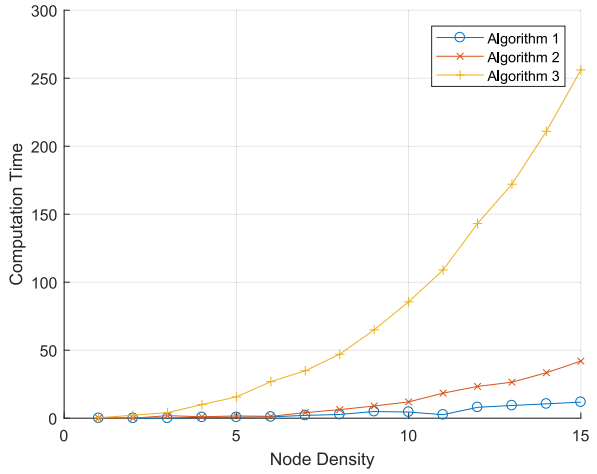


FIGURE 12. Computation time versus node density.

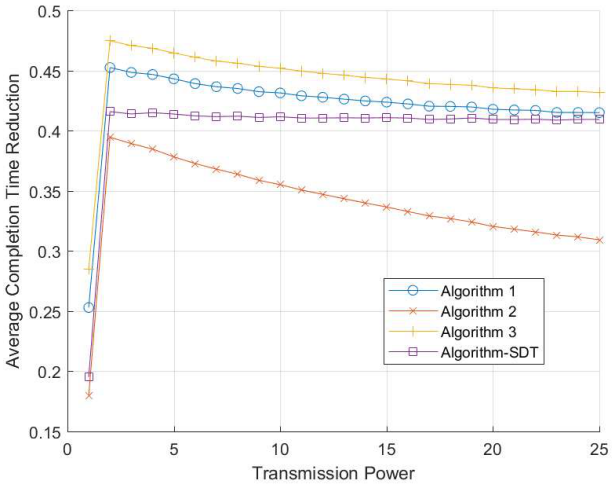


FIGURE 13. Average completion time reduction versus transmission power.

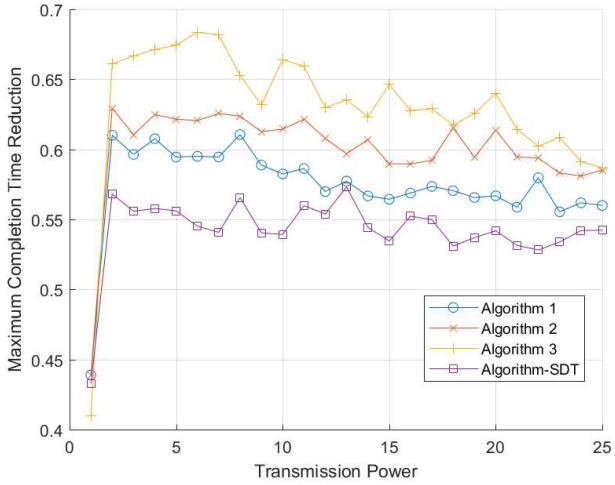


FIGURE 14. Maximum completion time reduction versus transmission power.

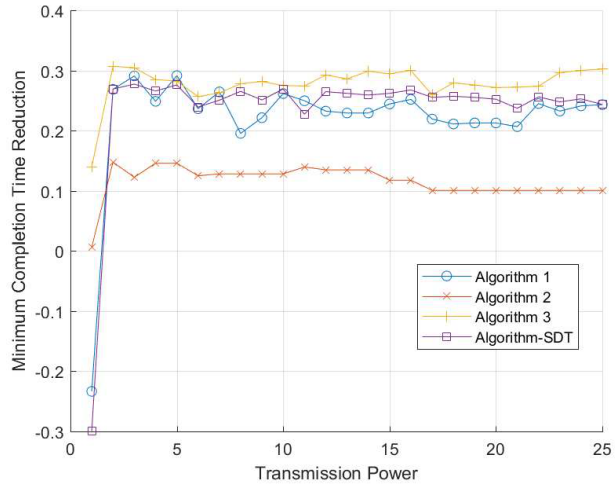


FIGURE 15. Minimum completion time reduction versus transmission power.

to 256 ms when the node density is fifteen. These results confirm our theoretical analysis in Section IV-D.

B. TRANSMISSION POWER

In this experiment we study the impact of different transmission powers. We vary the transmission power from 1 to 25 mW, with an interval of 1 mW. There are five APs and 25 clients. The SINR threshold for Algorithm-1 and Algorithm-SDT is 4 dB.

From Figure 13, 14 and 15, we observe that the Δ and Δ^+ value of all algorithms have the same trend. The reason is that a higher transmission power means clients experience a stronger received signal. Consequently, all links are able to use a higher data rate. However, when the transmission power continues to increase, the interference between links also increases. Therefore, the value of Δ and Δ^+ has a decreasing trend after 2mW. The Δ of Algorithm 1, Algorithm 2, Algorithm 3, and Algorithm-SDT starts from 25%, 17%, 28% and 19%, respectively. After that, all algorithms experience a significant jump in their Δ value. In particular, the Δ value of Algorithm-1, Algorithm-2, Algorithm-3 and Algorithm-SDT reaches 45%, 39%, 47% and 41%, respectively.

In Figure 14, we observe that the Δ^+ value of Algorithm-1, Algorithm-2, Algorithm-3 and Algorithm-SDT starts from 43%, 43%, 40% and 43%, respectively. Then, the Δ^+ value of Algorithm-1, Algorithm-2, Algorithm-3 and Algorithm-SDT respectively reaches 61%, 62%, 66% and 56% when the transmission power increased to 2 mW. After that, the Δ^+ of all algorithms has a decreasing trend.

From Figure 15, we can observe that the Δ^- value of all tested algorithms starts from -23%, 0.06%, 13% and -29%, respectively. Then, the Δ^- value of Algorithm 1, Algorithm 2, Algorithm 3 and Algorithm-SDT increases to 26%, 15%, 30% and 26%, thanks to the increased in transmission power. The Δ^- value of Algorithm 2 is the worst among all algorithms. The reason is because Algorithm 2

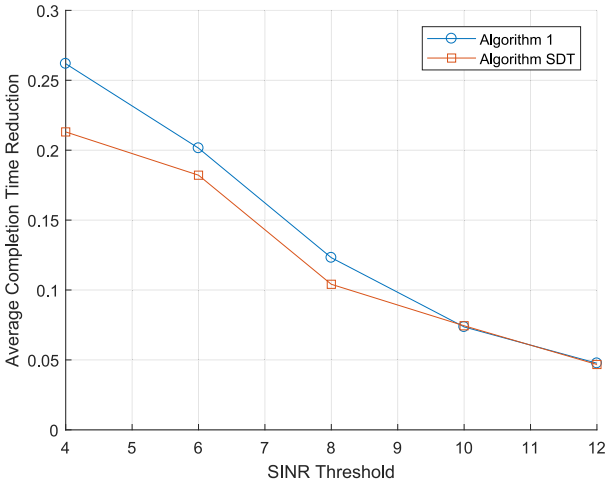


FIGURE 16. Average completion time reduction versus SINR threshold β.

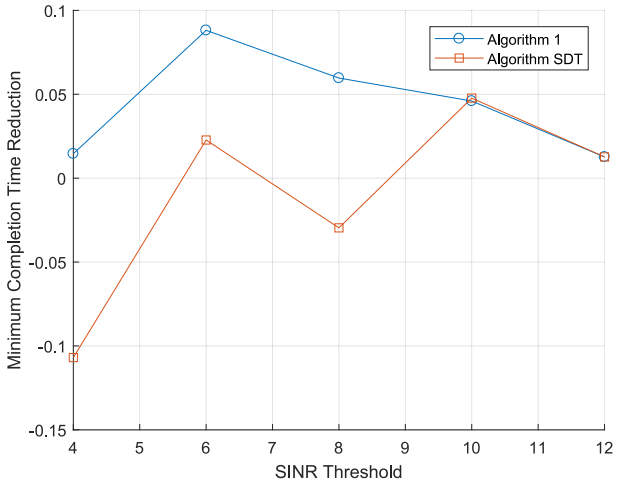


FIGURE 18. Minimum completion time reduction versus SINR threshold β.

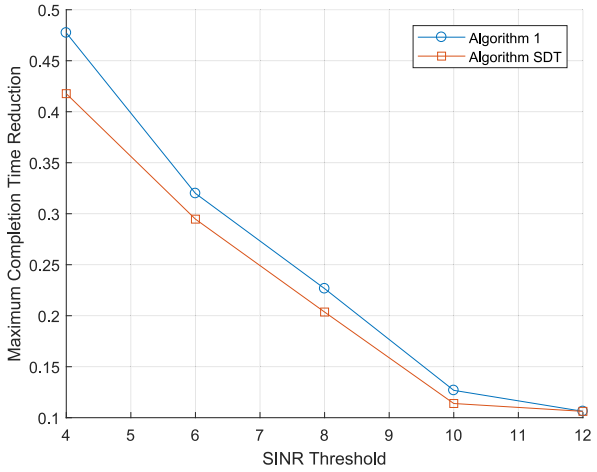


FIGURE 17. Maximum completion time reduction versus SINR threshold β.

uses the average SNR value of links as the SINR threshold. When the transmission power is high, the threshold chosen by Algorithm 2 can be high, which leads to fewer links being scheduled concurrently. This explains why Algorithm 2 has the worst performance.

C. SINR THRESHOLD β

In this experiment, we study the impact of different SINR thresholds β on completion time reduction. The number of APs is five and the number of clients is 25. The transmission power is 10 mW. We only study Algorithm 1 and Algorithm-SDT because the other two algorithms choose a SINR threshold by themselves. From Figure 16, we observe that the Δ value of Algorithm-1 is about 26% initially. The Δ value of Algorithm-SDT is about five percentage points lower than that of Algorithm 1. The Δ value of both algorithms decreases with the SINR threshold β. The difference in Δ value between Algorithm 1 and Algorithm-SDT also decreases. When the SINR threshold reaches 10 dB, the Δ value of both algorithms is the same, which is about 7.5%. The Δ value of both algorithm reduces to only 5% when the SINR threshold β is 12 dB. From Figure 17, we also observe a similar situation in terms of the Δ^+ value of Algorithm 1 and Algorithm-SDT. The Δ^+ value of Algorithm 1 is about 47% and the Δ^+ of Algorithm-SDT is about five percentage points lower. The Δ^+ of both algorithm reduces with SINR threshold β. When β is 12 dB, both algorithms have a Δ^+ value of 10%. The reason is that when the SINR threshold β increases, fewer links will be chosen to transmit concurrently due to higher interference. Algorithm 1 and Algorithm-SDT will only allow a small number of links to concurrently transmit when β is high, e.g., 12 dB. Therefore, the performance of both algorithms decreases. The value of Δ^+ also reduces to Δ because only a few links are allowed to transmit concurrently. However, the SINR threshold β has no impact on the minimum number of concurrently transmitting links.

In Figure 18, the Δ^- of Algorithm 1 fluctuates between 2% to 8%. The Δ^- value of Algorithm-SDT fluctuates between -11% to 5%. When the SINR threshold β is high, e.g., 10 dB, both algorithms only can find a small number of links that can transmit concurrently. Thus, the schedule obtained by both algorithms is similar to each other. The difference in Δ^- value between Algorithm 1 and Algorithm-SDT also decreases.

VII. CONCLUSION

This paper has addressed an important problem in dense WLANs comprising of APs equipped with an IBFD radio: deriving a schedule that allows nodes to complete the transmission of a given set of packets in minimum time. We propose three novel algorithms to maximize the number of concurrent transmissions and also to determine the best data rate for use by each transmitting link. Our results indicate that the proposed algorithms are able to reduce completion time by up to 68%. Moreover, the results show that our algorithms are superior to prior algorithms that schedule links on a

slot-by-slot basis. A key future work is to design distributed algorithms that allow APs to complete their transmissions without the help of a controller. Another immediate work is to consider random channel gains, which affect the level of interference over time.

REFERENCES

[1] Y. Kim, G. Hwang, J. Um, S. Yoo, H. Jung, and S. Park, "Throughput performance optimization of super dense wireless networks with the renewal access protocol," *IEEE Trans. Wireless Commun.*, vol. 15, no. 5, pp. 3440–3452, May 2016.

[2] D. J. Deng, K. C. Chen, and R. S. Cheng, "IEEE 802.11ax: Next generation wireless local area networks," in *Proc. 10th Int. Conf. Heterogeneous Netw. Quality, Rel., Secur. Robustness (QShine)*, Aug. 2014, pp. 77–82.

[3] E. Obregon, K. W. Sung, and J. Zander, "On the sharing opportunities for ultra-dense networks in the radar bands," in *Proc. IEEE Int. Symp. Dynamic Spectr. Access Netw. (DYSPAN)*, Apr. 2014, pp. 215–223.

[4] Cisco. (Mar. 2017). *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update 2016–2021*. [Online]. Available: https://goo.gl/hrTGF5

[5] I. Hwang, B. Song, and S. S. Soliman, "A holistic view on hyper-dense heterogeneous and small cell networks," *IEEE Commun. Mag.*, vol. 51, no. 6, pp. 20–27, Jun. 2013.

[6] C.-H. Lee, S.-H. Lee, K. Go, S.-M. Oh, J. S. Shin, and J.-H. Kim, "Mobile small cells for further enhanced 5g heterogeneous networks," *ETRI J.*, vol. 37, no. 5, pp. 856–866, 2015.

[7] Q. C. Li, H. Niu, A. T. Papathanassiou, and G. Wu, "5G network capacity: Key elements and technologies," *IEEE Veh. Technol. Mag.*, vol. 9, no. 1, pp. 71–78, Mar. 2014.

[8] L. Suresh, J. Schulz-Zander, R. Merz, A. Feldmann, and T. Vazao, "Towards programmable enterprise WLANs with odin," in *Proc. ACM HotSDN*, Helsinki, Finland, 2012, pp. 115–120.

[9] V. Shrivastava *et al.*, "CENTAUR: Realizing the full potential of centralized wlans through a hybrid data path," in *Proc. ACM Mobicom*, Beijing, China, 2009, pp. 297–308.

[10] *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Wireless Network Management*, IEEE Standard 802.11v WG, Nov. 2008

[11] J. I. Choi, M. Jain, K. Srinivasan, P. Levis, and S. Katti, "Achieving single channel, full duplex wireless communication," in *Proc. ACM Mobicom*, Chicago, IL, USA, 2010, pp. 1–12.

[12] E. Everett, A. Sahai, and A. Sabharwal, "Passive self-interference suppression for full-duplex infrastructure nodes," *IEEE Trans. Wireless Commun.*, vol. 13, no. 2, pp. 680–694, Jan. 2014.

[13] G. Liu, F. R. Yu, H. Ji, V. C. M. Leung, and X. Li, "In-band full-duplex relaying: A survey, research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 2, pp. 500–524, 2nd Quart., 2015.

[14] M. Jain *et al.*, "Practical, real-time, full duplex wireless," in *Proc. ACM Mobicom*, Las Vegas, NV, USA, 2011, pp. 301–312.

[15] X. Xie and X. Zhang, "Does full-duplex double the capacity of wireless networks?" in *Proc. IEEE INFOCOM*, Toronto, ON, Canada, Apr. 2014, pp. 253–261.

[16] P. C. Weeraddana, M. Codreanu, M. Latva-aho, and A. Ephremides, "On the effect of self-interference cancellation in multihop wireless networks," *EURASIP J. Wireless Commun. Netw.*, vol. 2010, no. 1, pp. 1–12, Oct. 2010.

[17] S. Goyal, P. Liu, O. Gurbuz, E. Erkip, and S. Panwar, "A distributed MAC protocol for full duplex radio," in *Proc. Asilomar Conf. Signals, Syst., Comput.*, Nov. 2013, pp. 788–792.

[18] W. Cheng, X. Zhang, and H. Zhang, "RTS/FCTS mechanism based full-duplex MAC protocol for wireless networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2013, pp. 5017–5022.

[19] S. Y. Chen, T. F. Huang, K. C. J. Lin, Y. W. P. Hong, and A. Sabharwal, "Probabilistic medium access control for full-duplex networks with half-duplex clients," *IEEE Trans. Wireless Commun.*, vol. 16, no. 4, pp. 2627–2640, Apr. 2017.

[20] W. Choi, H. Lim, and A. Sabharwal, "Power-controlled medium access control protocol for full-duplex WiFi networks," *IEEE Trans. Wireless Commun.*, vol. 14, no. 7, pp. 3601–3613, Jul. 2015.

[21] S. Kim, M. S. Sim, C. B. Chae, and S. Choi, "Asymmetric simultaneous transmit and receive in WiFi networks," *IEEE Access*, vol. 5, pp. 14079–14094, Jul. 2017.

[22] W. Zhou, K. Srinivasan, and P. Sinha, "RCTC: Rapid concurrent transmission coordination in full duplex wireless networks," in *Proc. IEEE ICNP*, Oct. 2013, pp. 1–10.

[23] O. Goussevskaia, R. Wattenhofer, M. M. Halldorsson, and E. Welzl, "Capacity of arbitrary wireless networks," in *Proc. IEEE INFOCOM*, Apr. 2009, pp. 1872–1880.

[24] O. Goussevskaia, M. M. Halldórsson, and R. Wattenhofer, "Algorithms for wireless capacity," *IEEE/ACM Trans. Netw.*, vol. 22, no. 3, pp. 745–755, Jun. 2014.

[25] V. Angelakis, A. Ephremides, Q. He, and D. Yuan, "Minimum-time link scheduling for emptying wireless systems: Solution characterization and algorithmic framework," *IEEE Trans. Inf. Theory*, vol. 60, no. 2, pp. 1083–1100, Feb. 2014.

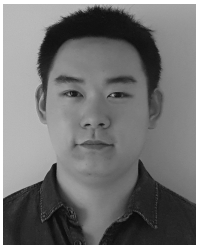
[26] Q. He, V. Angelakis, A. Ephremides, and D. Yuan, "Polynomial complexity minimum-time scheduling in a class of wireless networks," *IEEE Trans. Control Netw. Syst.*, vol. 3, no. 3, pp. 322–331, Sep. 2016.

[27] K. Chi, Y. Zhu, Y. Li, L. Huang, and M. Xia, "Minimization of transmission completion time in wireless powered communication networks," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1671–1683, May 2017.

[28] Y. Xu, K.-W. Chin, S. Soh, and R. Raad, "Scheduling links with air-time in multi transmit/receive wireless mesh networks," *Wireless Netw.*, vol. 22, no. 6, pp. 1999–2012, Aug. 2016.

[29] J. Y. Kim, O. Mashayekhi, H. Qu, M. Kazandjieva, and P. Levis, "Janus: A novel MAC protocol for full duplex radio," Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, Tech. Rep. CSTR-2013-02, Jul. 2013.

[30] M. S. Gast, *802.11ac: A Survival Guide: Wi-Fi at Gigabit and Beyond*. Newton, MA, USA: O'Reilly Media Inc., 2013.



YIFEI REN received the B.E. degree from the University of Wollongong, Australia, and Zhengzhou University, China, in 2016. He is currently pursuing the Ph.D. degree at the University of Wollongong. His research areas are resource allocation approaches for full-duplex wireless networks.

KWAN-WU CHIN received the B.Sc. (Hons.) and Ph.D. degrees from Curtin University, Australia, in 1997 and 2000, respectively. He was a Senior Research Engineer at Motorola, 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 2004, he joined the University of Wollongong as a Senior Lecturer before being promoted to the rank of Associate Professor in 2011. He holds four U.S. patents. He has published more than 100 conference and journal articles. His current research areas include medium access control protocols for wireless networks, and resource allocation algorithms/policies for communications networks.

SIETENG SOH (M'98) received the B.S. degree in electrical engineering from the University of Wisconsin–Madison and the M.S. and Ph.D. degrees in electrical engineering from Louisiana State University, Baton Rouge. From 1993 to 2000, he was a Faculty Member at 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, Perth, Australia. His research interests include network reliability, and parallel and distributed processing.