Computer Communications 228 (2024) 107979

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Proactive Scheduling for mmWave Wireless LANs

Ang Deng[1], Douglas M. Blough[1,*]

School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0765, USA

ARTICLE INFO

Keywords:
Millimeter-wave
Proactive scheduling
Wireless LAN

ABSTRACT

To cope with growing wireless bandwidth demand, millimeter wave (mmWave) communication has been identified as a promising technology to deliver Gbps throughput. However, due to the susceptibility of mmWave signals to blockage, applications can experience significant performance variability as users move around due to rapid and significant variation in channel conditions. In this context, proactive schedulers that make use of future data rate prediction have potential to bring a significant performance improvement as compared to traditional schedulers. In this work, we explore the possibility of proactive scheduling that uses mobility prediction and some knowledge of the environment to predict future channel conditions. We present both an optimal proactive scheduler, which is based on an integer linear programming formulation and provides an upper bound on proactive scheduling performance, and a greedy heuristic proactive scheduler that is suitable for practical implementation. Extensive simulation results show that proactive scheduling has the potential to increase average user data rate by up to 35% over the classic proportional fair scheduler without any loss of fairness and incurring only a small increase in jitter. The results also show that the efficient proactive heuristic scheduler achieves from 60% to 75% of the performance gains of the optimal proactive scheduler. Finally, the results show that proactive scheduling performance is sensitive to the quality of mobility prediction and, thus, use of state-of-the-art mobility prediction techniques will be necessary to realize its full potential.

1. Introduction

For next-generation wireless networks, both in the cellular and wireless LAN domains, mmWave communications are considered a key technology due to the plentiful bandwidth available in mmWave bands. However, a major technical challenge with deploying mm-Wave technology in practical settings is the susceptibility of mm-Wave signals to blockage due to their short wavelengths. It is well established that mmWave signals are completely blocked by relatively small obstacles including even the human body [1–3].

As a result of the blockage issue, the link quality of a mmWave link can vary rapidly and widely as a user moves around within a space containing obstacles. Meanwhile, many next-generation applications such as ultra-high-definition real-time video and wireless virtual reality require continuous high-quality wireless links. One potential approach to help deal with this issue is to schedule packet transmissions when channel conditions are good and high date rates can be achieved. Such an approach has been referred to as *proactive scheduling*.

Prior work has considered proactive scheduling based on blockage predictions on the order of milliseconds by detecting when link quality is beginning to degrade [4]. It is the premise of our work that it is possible to predict blockages farther into the future by combining knowledge of the environment with user mobility prediction, thereby opening up greater potential performance improvements with proactive scheduling.

In this paper, we explore in detail the potential benefits achievable with proactive scheduling based on mobility and rate prediction. To be specific, we combine mobility prediction with some knowledge of the environment, e.g. an RF heat map, to predict channel conditions of users several seconds into the future. With predicted rates as input, we first formulate the problem of maximizing rate subject to a fairness constraint as a binary integer linear programming problem. Solution of this problem, although not practical in real network settings, provides an upper bound on the performance achievable with proactive scheduling. We then define a greedy heuristic algorithm that first goes time slot by time slot, ordered from harshest to most benign conditions, and then, within each considered time slot, orders users from most difficult to schedule to easiest to schedule. Finally, we compare, through extensive simulations, the performance of proactive scheduling with that of classic proportional fair (PF) scheduling under the same fairness constraints. The simulations also consider performance with both accurate and first-order mobility predictors and with only static obstacles and with both static and dynamic obstacles.

* Corresponding author.
E-mail addresses: adeng3@gatech.edu (A. Deng), doug.blough@ece.gatech.edu (D.M. Blough).
[1] Both authors contributed equally to the work included in this paper.

https://doi.org/10.1016/j.comcom.2024.107979
Received 8 March 2024; Received in revised form 13 August 2024; Accepted 14 October 2024
Available online 20 October 2024

The main contributions of the paper are:

- a framework for proactive scheduling in mmWave LAN that combines mobility prediction with link quality prediction to predict user performance several seconds into the future,
- formulation of a binary integer linear programming problem defining an optimal proactive schedule, given predicted rates and a fairness constraint,
- specification of a greedy heuristic proactive scheduler for the same problem, which considers time slots and users in a specific order to achieve both efficient running time and good performance, and
- extensive simulation results that demonstrate:
  - proactive schedulers can achieve up to a 35% increase in average user data rate compared to classic proportional fair scheduling without any loss of fairness and only a small increase in jitter,
  - our efficient heuristic proactive scheduler achieves between 60% and 75% of the performance gain of the optimal proactive scheduler, and
  - when trivial straight-line mobility prediction is employed, performance gains relative to non-proactive scheduling are still significant but drop to 5% to 15%, indicating that high-quality mobility prediction is important to achieve the full potential benefits of proactive scheduling.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 details the system model, while Section 4 presents the problem formulation. Section 5 describes our overall approach to rate prediction and proactive scheduling and also presents an efficient heuristic scheduling algorithm. Evaluations of proactive scheduling with static obstacles only and with both static and dynamic obstacles are reported in Sections 6 and 7, respectively. Finally, Section 8 discusses key findings and future work, concluding the paper.

2. Related work

The idea of proactive scheduling is not new to wireless networking. Previous works have studied proactive scheduling in non-mmWave wireless networking [5–7]. These works leverage the predictability of user channel state, and each develop their own extended form of the proportional fair scheduler which utilizes this future knowledge. In [5], the exact method of predicting future data rate is not introduced but assumed, whereas [6,7] present their own method for estimating the user data rate by leveraging the channel fading effect. Their results consistently agree that proactive scheduling can provide significant throughput gains while maintaining fairness levels. However, the channel state information (CSI) estimation techniques in these works would not apply in mmWave scenarios, due to mmWave's high susceptibility to blockage compared to sub-mmWave bands.

In recent years, several works have explored the use of blockage prediction to cope specifically with mmWave's highly dynamic channel conditions [8–13]. These works employ different techniques to achieve blockage prediction, and are designed for different network goals. [8] considers the optimization of handover performance in dense cellular networks. Blockage occurrence and duration are predicted using peripherals and geometry, enabling elimination of unnecessary handovers caused by transient blockage as well as early handover to avoid long blockage. In [9], a proactive path selection mechanism is proposed to improve resilience of multihop mmWave networks to blockages. In [10], camera images are used to predict blockages caused by human mobility. Learning based techniques are also used to predict blockage for mmWave transmission [11–13]. In [11], the blockage prediction decision is made by a trained classifier. On the other hand, [12,13] both utilize deep learning. The difference is that [12] uses in-band information to make the prediction, whereas [13] uses out-of-band information.

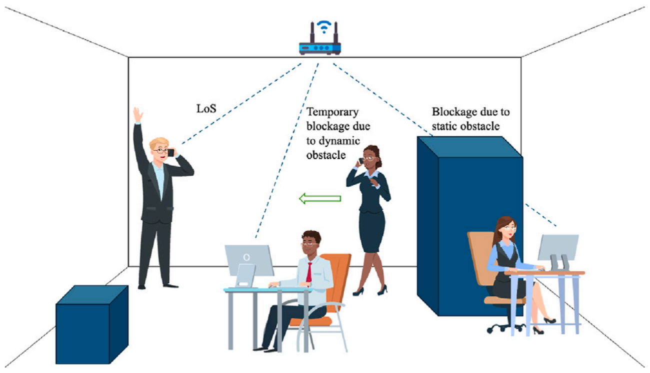
LoS
Temporary blockage due to dynamic obstacle
Blockage due to static obstacle

Fig. 1. mmWave wireless local area network with ceiling mounted AP.

There are only a few works that focus specifically on blockage-aware scheduling for mmWave networks [4,13]. Similar to the aforementioned proactive schedulers, [4] proposes a variation of the proportional fair scheduler. In this work, the scheduler reverses the order of time slot scheduling (going from the furthest future to the nearest future) in order to allocate time slots to users before the predicted blockage happens. This approach successfully boosts the link performance for users who experience the most severe blockage, however it actually slightly decreases the overall throughput of the system compared to proportional fair scheduling. In [13], a DNN scheduler is proposed, which combines blockage prediction with scheduling and beamforming. This work aims to maximize users' achievable rates but does not take fairness into consideration.

3. System model

We model the indoor mmWave LAN environment of a single room containing n_u mobile user devices communicating with a single ceiling-mounted access point and a number of 3-dimensional static obstacles, modeled as cuboids, on the floor. This creates several different communication conditions as illustrated in Fig. 1. Some users have direct line-of-sight (LoS) paths to the access point while others, who are blocked by static obstacles, must rely on non-line-of-sight (NLoS) paths. In most of our results and analysis, we focus on this static obstacle environment. However, in Section 7, we also consider the users to be mobile obstacles, who are capable of blocking other users, as illustrated in the figure by the woman whose path will eventually cause her to block the LoS path of the seated man.

Some information about the environment is assumed to be known to the proactive scheduler so that rate predictions for users at specified locations can be made. This could be just the line-of-sight (LoS) status of the user, which can be calculated from the user and AP positions and knowledge of the obstacles' sizes and locations. Alternatively, more granular rate data could be obtained from an RF heat map of the room, which gives signal strength data across the geometry of the room and can be acquired from a dedicated measurement campaign or from user measurements collected over time. In all simulations, static obstacles are modeled as cuboids with their locations distributed according to a Poisson process and their lengths, widths, and heights normally distributed. In the simulations of Section 7, for simplicity, we model the moving human obstacles as cuboids with fixed length and width and normally-distributed heights.

The path loss model is divided into two cases. The first case is the LoS case, where the user received power is calculated by the standard distance-based path loss equation:

$$p_{rx} = p_{tx} \cdot g_{tx} \cdot g_{rx} \cdot l_0 \cdot d_u^{-\upsilon} , \quad (1)$$

where p_{tx} is the transmit power, g_{tx} and g_{rx} are the transmit and receive antenna gains, l_0 is the free space path loss at reference distance of 1 m, d_u is the distance between transmitter and receiver, and υ is the attenuation exponent. The user data rate is then calculated with Shannon's capacity formula:

$$r = b \cdot log_2 \left(1 + \frac{p_{rx}}{p_n}\right) . \tag{2}$$

The second case is the non-line-of-sight (NLoS) case. In this case, the received rate has been shown to be highly variable, ranging from near zero to something close to the LoS rate, depending on the blockage conditions and the reflectivity properties of the surrounding materials [14]. To model this behavior, in the simulation results of Sections 6 and 7, we assume that the NLoS rate is uniformly distributed between zero and the LoS rate with an extra 10 dB of loss.

We assume the AP transmits an infinitely backlogged queue of downlink data to the users using the orthogonal frequency-division multiple access (OFDMA) scheme. The total duration of the transmission is T seconds, consisting of time slots of length Δt. The total duration is divided into scheduling sessions each containing n_{ts} time slots. All sub-carriers of a time slot are assigned to the same user, where the aggregate bandwidth is b.

4. Scheduling problem formulation

In this section, we present an overview of the specific proactive scheduling problem considered herein, give a simple example, and then formally define the problem as a type of integer linear programming optimization (ILP) problem. An efficient heuristic for this scheduling problem can be used within the overall proactive approach described in the next section as the basis for practical implementation of a proactive mmWave LAN.

4.1. Overview

We consider a scheduling problem where an input to the scheduler is the predicted data rate for all users at every time slot of the next scheduling session. For fairness considerations, the algorithm also takes as input an allotment scheme, which dictates how many time slots are to be allocated to each user.[2] These two pieces of information are all our proposed efficient heuristic scheduler needs to produce a schedule. The output schedule is represented by a matrix X of size n_u by n_{ts}. In this matrix, each row represents a user and each column represents a time slot, and a value of 1 means the user is assigned for communication in the time slot and 0 means it is not assigned. The problem constraints are that each time slot (column) must have exactly one user assigned and, for every user i, the sum of the values in row i must equal user i's allotment. Subject to these constraints, the goal is to maximize the aggregate data rate over the scheduling session under consideration.

4.2. Example

We first depict the scheduling problem in an assignment table to lay the foundation for understanding our formulation as a type of ILP problem. In Table 1, we show a sample assignment schedule for a system with 2 users and 4 time-slots. Each value of 1 in the table indicates that the corresponding time-slot has been assigned to a particular user. In this example, time-slots 1, 3, and 4 are assigned to user 1 and time-slot 2 is assigned to user 2. A value of zero means that the user is not active in the corresponding time-slot.

In parallel, we have Table 2 with the same structure but showing the predicted rate for each user in each time-slot based on the predicted channel conditions. Multiplying these two tables element-wise and taking the average over all time-slots yields the predicted average data rate during this period, which for the example shown is 1.5 Gbps.

Table 1
Example assignment table for a 2 user 4 time-slot system.

	TS 1	TS 2	TS 3	TS 4
User 1	1	0	1	1
User 2	0	1	0	0

Table 2
Example predicted rates for a 2 user 4 time-slot system (Gbps).

	TS 1	TS 2	TS 3	TS 4
User 1	1.2	0.8	1.2	2.1
User 2	0.4	1.5	1.2	0.8

It can be seen from Table 1 that in order for the schedule to be valid, there are two conditions that need to be met. First, the sum of each row indicates how many time-slots are assigned to this particular user, and these row sums need to add up to the number of total time-slots, which is 4 in this case. In our implementation, in order to guarantee comparable fairness with other scheduling algorithms as well as to prevent the algorithm from overly prioritizing users with favorable channel conditions, we take this requirement a step further and constrain each user to have a particular number of slots. As mentioned earlier, this approach allows our algorithm to achieve a certain fairness condition. Second, for each time-slot (column) there must be exactly one user to whom the time-slot is assigned. In other words, each column must sum to 1. If a table meets these two conditions, then it is a valid schedule.

4.3. Formulation as BILP problem

In this subsection, we formulate the scheduling problem as a binary ILP, or BILP, problem, with the goal being to maximize the sum rate over a single scheduling session with a given fairness constraint. The main constraints in our scheduling problem are:

1. there is exactly one user assigned in each time slot, and
2. a user u_i is assigned to exactly b_i time slots in the scheduling session under consideration.

The second constraint is what allows us to realize a specified fairness criterion. In the simulation results of Sections 6 and 7, we compare the optimal scheduler to the proportional fair scheduler, and there we allocate to each user the same number of time slots it is allocated in the corresponding proportional fair scheduler. Under that allocation, each user gets the same amount of time in the wireless channel as it gets in the corresponding proportional fair scheduler, which we consider to be achieving the same approximate fairness.[3]

We begin the problem formulation by defining a column vector $\mathbf{X}$ of size $n_U * n_{ts}$ which contains binary numbers denoting whether a particular time-slot is assigned to a specific user (0 for not assigned and 1 for assigned). This assignment vector can be considered as an unwinding of Table 1 going in ascending time order and row by row. For example, the sample schedule in Table 1 would be represented by the vector [1 0 1 1 0 1 0 0]. Each of the values in the vector represents if the time-slot is assigned to the corresponding user, and it is interpreted

[2] Different allotment schemes can be used to achieve different types of fairness. Examples of allotment schemes that produce different fairness conditions are discussed later.

[3] Note that the relative rates of different users will not be *exactly* the same for the optimal and proportional fair schedulers since the rates each user gets in its different time slots might differ across the two schedules. However, we will show in Section 6.2 that the fairness values of the optimal scheduler and a corresponding proportional fair scheduler are in fact very close in practice.

that the first n_{ts} elements represents the assignment for user 1, and the second n_{ts} elements for user 2, etc. An assignment vector of this type is the output of our optimized scheduler.

Next, we construct a matrix **A** of size $n_U + n_{ts}$ by $n_U * n_{ts}$. The first n_U rows denote the constraint that each user is only allowed a fixed number of time-slots. The last n_{ts} rows denote the constraint that only one user can be assigned in each time-slot. The exact construction of **A** is as follows: let k be the row number, then for the first n_U rows, for each value of k, $a_{kq} = 1$ for $(k-1) * n_U < q \leq k * n_U$, and $a_{kq} = 0$ otherwise. For the last n_{ts} rows, for each value of k, $a_{kq} = 1$ for every n_{ts}th element starting from the $(k - n_U)$th element in row, and $a_{kq} = 0$ otherwise. Matrix **A** is only dependent on n_U and n_{ts}.

Finally, we define a column vector **B** of length $n_U + n_{ts}$ that includes the b_i constraints. The first n_U rows equal the number of total time-slots to be allocated to each user and the last n_{ts} rows are all equal to 1.

To illustrate the matrix setup, if we have 2 users and 4 time-slots, and each user is allocated 2 time-slots, then the equation $\mathbf{AX} = \mathbf{B}$ is:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \tag{3}$$

This equation ensures that any solution **X** satisfies the aforementioned constraints and is therefore a valid schedule.

To implement the objective function, we construct a row vector **R** whose elements have a 1-to-1 correspondence to vector **X**, similar to how Table 2 is to Table 1. The element values of **R** are the predicted data rates as calculated in (2) for users in each time-slot. The product of **RX** and Δt is the total amount of data that is transmitted in the total duration of $n_{ts} * \Delta t$ scheduled time (one scheduling session). Putting everything together, the optimization problem to maximize the aggregate rate across n_U users and n_{ts} time-slots with fairness constraints is given by the following BILP problem:

$$max\ \mathbf{RX}$$

$$s.t.\ \mathbf{AX} = \mathbf{B} \tag{4}$$

whereby the solution vector **X** is an optimal schedule.

This BILP formulation allows us to use off-the-shelf optimization software to generate optimal proactive schedules. While this solution technique is not practical for implementation in a real network setting, it provides an upper bound on the performance achievable with proactive scheduling. This allows us to both evaluate the maximum potential benefit of proactive scheduling as compared to non-proactive scheduling and to determine how close to optimal is our heuristic proactive scheduler presented in Section 5.2.

5. Proactive scheduling approach

As just discussed, an optimal proactive scheduler based on direct solution of the BILP problem is computationally expensive and, therefore, not suitable for real-time scheduling in practical network environments. In this section, we introduce a practical proactive approach that is based on mobility prediction and an efficient heuristic scheduling algorithm, which is capable of performing proactive scheduling in real time. Later, evaluations will show that the performance of this efficient scheduler is much better than the classic proportional fair (non-proactive) scheduler and is not far from the performance of the optimal proactive scheduler.

5.1. Overall process

The proactive scheduling process is assumed to take place in a room with known geometry, known obstacle sizes and locations, users moving around in an unknown fashion, and a mmWave access point at a known location. A schedule is to be determined for the next scheduling session of a certain length broken into a number of time slots of fixed duration. Given the current location of each node along with its current direction and speed, a mobility predictor predicts the path of the node during the next scheduling session.

The next step is to use the predicted path of a node to generate a predicted data rate for the node at each time slot of the next scheduling session. One approach to this is using an RF map [14–16], which is a heat map of SNR across a region of interest. If an RF map is available for the given space, the predicted data rate is a straightforward mapping of location to SNR to data rate. If no RF map is available, a predicted data rate can be calculated for a given location by determining the line-of-sight status between the location and the access point, applying the appropriate path loss model (either line-of-sight or non-line-of-sight), and performing an SNR to data rate conversion. The predicted data rate vectors for each user, denoted by $rate_u$ for user u, contain the user's predicted data rate at each time slot and are primary inputs to the proactive scheduler.

The other input to the scheduler is the number of slots that each user should be assigned in the next scheduling session. These allocations can be determined in many ways. An obvious one, corresponding to the notion of time fairness, is to assign each user the same number of time slots. For the simulations presented in Sections 6 and 7, we assume the time slot allocation resulting from the proportional fair (PF) scheduler in order to compare the performance of proactive schedulers to the classic PF scheduler under the same fairness conditions.

The only basic assumption of our scheduling approach is that there is a predicted data rate for every location within the region of movement of the users. In the simulation results presented later, we generate these values based on the geometry of the room, including sizes and locations of obstacles, and path loss models for LoS and NLoS conditions. However, as mentioned earlier, this information can also come from an RF map of the space, which could be learned from measurements taken as users move around. With such a measurement-based RP map approach, *there is no need to know the room geometry and obstacle information to generate the necessary predicted data rates.*

5.2. Heuristic scheduler

As mentioned earlier, solving the BILP formulation directly in real time is not practical. In order to achieve good data rate performance within a real-time scheduling constraint, we propose an efficient greedy heuristic algorithm that can schedule users to time slots in a fair allotment scheme. Greedy schedulers have been shown to perform remarkably well in many wireless contexts [17–19]. The main issue to consider with a greedy scheduler is in what order to consider the time slots and the users within each time slot, which is discussed next.

This algorithm utilizes prediction of future low data rate occurrences to prioritize the scheduling of scarce resources, thus securing good channel conditions for users that are the most likely to be affected by adverse channel conditions. This prioritization is achieved through ranking of the time slots by how many low-rate users they contain and ranking of users by a combination of the number of time slots they need to be allocated and in how many time slots they are predicted to have a low data rate. Intuitively, the user with the highest number of low-rate slots plus to-be-allocated slots is the most difficult to schedule. Although we do not present the results herein, we tried several other orderings of time slots and users and this method performed the best.

The detailed scheduling algorithm is shown in Algorithm 1. The inputs to the algorithm are as follows. The predicted data rate is represented by the matrix $rate_u$ of size $n_u \times n_{ts}$. This matrix gives the predicted data rate for every user at every time slot over the interval to be scheduled.[4]. Since not all blocked LoS conditions result in a low data rate, we do not simply categorize users by LoS or NLoS status. Instead, we apply a threshold rate to the predicted rate matrix to categorize users as either high rate or low rate within each time slot. The predicted channel conditions are represented by the matrix $status_u$, also of size $n_u \times n_{ts}$. Each row of the matrix represents a user and each column represents a time slot. A value of 1 means the respective user is predicted to experience a low data rate for the particular time slot and a value of 0 indicates good channel conditions are predicted. $status_u$ is directly generated from $rate_u$ by applying the aforementioned threshold on the predicted rates. Lastly, the fair allotment scheme is represented by matrix $allot_u$ which is an $n_u \times 1$ vector, where element i represents the designated number of time slots that user u_i is to be scheduled in for the coming scheduling interval. We refer to this as the allotment scheme.

Algorithm 1 Heuristic scheduler

```
Inputs:
  $rate_u$ (predicted user data rates at every time slot),
  $status_u$ (predicted user low-rate status at each timeslot,
        1 if $rate_u \le$ threshold, 0 if $rate_u >$ threshold),
  $allot_u$ (number of time slots to be assigned to each user)
Output: $X$ (user assignment)

1: $low\_rate_u[user] = \sum_{ts=1}^{n_{ts}} status_u[user][ts],\ 1 \le user \le n_u$
2: $low\_rate_{ts}[ts] = \sum_{user=1}^{n_u} status_u[user][ts],\ 1 \le ts \le n_{ts}$
3: $rank_{ts} = low\_rate_{ts}$ sorted from largest to smallest
4: for each $ts \in rank_{ts}$ do
5:     $assigned = 0$
6:     $metric_u = low\_rate_u + allot_u, \forall u$
7:     $rank_u$ = user IDs sorted from largest to smallest value of
              $metric_u, \forall u$ with $allot_u > 0$
8:     for each $user \in rank_u$ do
9:         if $status_u[user][ts] == 0$ and $allot_u[user] > 0$ then
10:            $X[user][ts] = 1$
11:            $allot_u[user] = allot_u[user] - 1$
12:            $low\_rate_u[v] = low\_rate_u[v] -$
13:                     $status_u[v][ts],\ 1 \le v \le n_u$
14:            $assigned = 1$
15:            break
16:        end if
17:    end for
18:    if $assigned == 0$ then
19:        for each $user \in rank_u$ do
20:            if $allot_u[user] > 0$ then
21:                $X[user][ts] = 1$
22:                $allot_u[user] = allot_u[user] - 1$
23:                $low\_rate_u[v] = low\_rate_u[v] -$
24:                         $status_u[v][ts],\ 1 \le v \le n_u$
25:                break
26:            end if
27:        end for
28:    end if
29: end for
30: return $X$
```

Algorithm 1 can be divided into two steps. In the first step, the algorithm computes a number of sums: the total number of low rate slots for each user and the total number of low-rate users for each time slot, in preparation for the ranking (Lines 1–2). The result is two vectors low_rate_u of length n_u and low_rate_{ts} of length n_{ts}. Then, the time slots are ranked from most to least low-rate users (Line 3) so that time slots with the most adverse channel conditions are prioritized. The second step is the outer for loop (Lines 4–27). In this loop, the algorithm sequentially walks through the time slots in the scheduling period based on the just computed ranking. Inside the loop, the $assigned$ flag keeps track of whether the current time slot has been assigned or not and is initially set to 0 at the beginning of each iteration (Line 5). Also, for each iteration, $metric_u$ is calculated for every user by summing low_rate_u and $allot_u$, and then the users are ranked by $metric_u$, from highest to lowest, to prioritize users that are predicted to experience the worst channel conditions and that need to be assigned the most time slots (Lines 6–7). Once a slot is assigned, the allotment for the assigned user is reduced by one and every user that had a low rate in the assigned slot has their low rate slot count reduced by one (Lines 11–12 and 22–23).

The details of user selection within the outer loop are as follows. There are two inner for loops. The first inner loop (Lines 8–17) walks through users from highest to lowest $metric_u$ and assigns the current time slot to the first user that has both available allotment and a high data-rate in the slot being considered (Line 10). Once a user is assigned, the available allotment for this user $allot_u[user]$ is decremented (Line 11). Each element of the low_rate_u vector is then updated according to the $status_u$ of each user in the just allocated time slot, so that the value reflects the number of low rate slots for each user in the remaining unassigned time slots (Lines 12–13). If a user is assigned in this step, the $assigned$ flag is set to 1 (Line 14).

If there is no user that meets the criteria to be assigned to the slot in the first inner for loop, the second inner for loop is executed. This means that there are no users with allotment remaining that have a high predicted rate in the current slot being assigned. In this case, the second for loop goes through the users again to assign this time slot to the highest ranked user that still has available allotment (Lines 17–26). If a user is assigned in this second loop, its allotment is updated and the low rate slots for all users are updated, just as when assignment happens in the first for loop (Lines 22–24).

The time complexity of this heuristic algorithm is $O(n_u \cdot n_{ts})$, where the n_{ts} accounts for the outer loop and n_u accounts for the sorting and iterated assignment which happens for every iteration of the outer loop. Therefore, the time complexity of the heuristic scheduling algorithm is polynomial. User sorting here is $O(n_u)$ because the update to $rank_u$ is very limited. The update to $allot_u$ only results in the adjustment of $rank_u$ by shifting one user, and the update to low_rate_u only results in a decrement by 1 for all affected users (users having a low rate in that time slot). Therefore, to handle the update to low_rate_u, we can just divide the sorted array into arrays of unaffected users and affected users, decrement each affected user, and merge the two divided arrays back, which is $O(n_u)$ complexity. By inspection, the remainder of the outer loop body is also $O(n_u)$.

6. Performance study with static obstacles

This simulation study is set in rooms of size 20 m by 20 m where obstacles are randomly placed. The users' mobility is modeled with a hot spot mobility model with a number of randomly located hot spots where users pause for a certain duration of time, and then move to a different hot spot location. Users take the shortest path towards the next hot spot, except that they skirt around the edges of any obstacles encountered along the way.

For data rate prediction, we use a naive mobility predictor, which only predicts the user to continue at the current speed in the current direction. The predicted data rate is then calculated using the predicted location, the known room and obstacle geometries, and the line-of-sight and non-light-of-sight path loss models presented in Section 3. Although this results in perfect data rate prediction if the exact user

[4] Recall that the predicted data rates are generated by first estimating user location over the scheduling session interval through mobility prediction and then predicting data rate at those locations by using, for example, either an RF heat map of the region or ML-based rate prediction.

Table 3
Variable simulation parameters.

Parameter	Low	Medium	High
Scheduling session length (s)	1	3	5
User pause time (s)	2	4	6
Number of obstacles	50	65	80
Number of hot spots	4	6	8

location is known, the mobility prediction produces location errors that result in errors in data rate prediction.

Due to its very basic nature, the naive mobility predictor can be considered to yield worst case results for proactive scheduling. As a reference, we also ran all the experiments with a perfect predictor to show the best case performance for the optimal BILP scheduler and our heuristic proactive scheduler. If state-of-the-art mobility prediction techniques are employed, results will fall somewhere in between these two values.

In this section we compare the performance of 3 schedulers: (1) our proposed heuristic scheduler (denoted by HEUR in the following sections), (2) an optimal proactive scheduler (denoted by BILP), which solves the binary integer linear programming problem defined in Section 4.3 exactly and was implemented using the IBM ILOG CPLEX optimizer, and (3) the traditional proportional fair scheduler (denoted by PF), which is non-proactive and is defined as follows. The PF scheduler prioritizes users with the highest instantaneous-to-average rate ratio, promoting fairness and maximizing the sum of logarithmic average user rates. In the PF scheduler, the next user to be scheduled u^* is chosen by the following equation:

$$u^* = \underset{u}{\operatorname{argmax}} \frac{r_u(t)}{\bar{r}_u(t-1)}, \tag{5}$$

where $\bar{r}_u(t)$ is an exponential moving average defined by:

$$\bar{r}_u(t) = (1-w)\,\bar{r}_u(t-1) + w\, r_u(t), \tag{6}$$

and w is the weight of the current data rate, having a value between 0 and 1.

The proportional fair scheduler balances good performance with fairness and is included as a representative of what can be achieved with non-proactive scheduling. Since the BILP scheduler solves the proactive scheduling problem exactly, it represents the best possible proactive scheduling performance. To provide an equitable comparison, all experiments conducted hereafter use the same allotment scheme, which is the one produced by the PF scheduler. This means that the three algorithms will produce approximately the same fairness allowing us to focus on performance under different conditions: non-proactive scheduling (PF), heuristic proactive scheduling (HEUR), and optimal proactive scheduling (BILP).

6.1. Simulation settings

In this section, we study the influence of several variables on the performance of the schedulers, including: scheduling session length, user pause time, obstacle density and hot spot density, where for each of these factors we picked 3 values representing low, medium and high values. The full set of values used in the experiments are listed in Table 3. The default setting for experiments uses the medium values for all parameters, which is the base line case discussed in Section 6.2. The fixed variables are given by Table 4.

The values for scheduling session length were chosen based on the range of times over which it is reasonable to expect relatively accurate mobility prediction performance. The largest value corresponds to walking roughly halfway across the 20 m by 20 m room at typical human walking speed. User pause times are somewhat on the low side for what might be expected in practice. Longer pause times mean that users are paused most of the time resulting in trivial and highly

Table 4
Fixed simulation parameters.

Bandwidth	b	2 GHz
Noise power	p_n	−71.99 dBm
Scheduling time slot length	Δt	62.5 μs
Transmit power	p_{tx}	20 dBm
Transmitter gain	g_{tx}	3.16 dBi
Receiver gain	g_{rx}	0 dBi
Path loss reference	l_0	63.4 dB
Attenuation exponent	υ	1.72
Moving average weight	w	0.5

accurate mobility prediction. Therefore, we studied shorter pause times to be able to assess the impact of variable mobility prediction quality. The parameters in Table 4 and the mean obstacle dimensions specified below are typical values that were taken from several earlier published papers [4,14].

All experiments performed in the following subsections simulate a 120 s period in which 20 users are concurrently moving in the same 20 m by 20 m room. The ceiling mounted access point is located at the center of the room at a height of 3 m. The obstacles in the room are randomly allocated following a Poisson distribution. The obstacles are assumed to be cuboids sitting on the floor and their dimensions are given by the following distributions: $W \sim \mathcal{N}(0.56, 0.08, 0.25, 1.25)$, $L \sim \mathcal{N}(1.08, 0.18, 0.5, 1.75)$, and $H \sim \mathcal{N}(1.85, 0.2, 1.25, 2.4)$. The obstacles are randomly aligned to be parallel to one of the room walls. Finally, when a user arrives at a hot spot, they choose a new hot spot equiprobably at random from all other hot spots to be the next location.

The mobility model was implemented in the ns-3 enhanced mmWave LAN simulator [20]. This simulator includes obstacle modeling and code to determine the LoS/NLoS status of a user device. This information was then fed to an off-line module that calculated predicted data rates according to the LoS/NLoS models presented in Section 3.

6.2. Average data rate comparison on baseline case

We first analyze a baseline case performance, which is the case used for comparison in later subsections that vary one parameter at a time. In this baseline case, the parameter values correspond to the middle column of Table 3.

Let $rate_{PF}$, $rate_{BILP}$, and $rate_{HEUR}$ represent the average data rates of the PF, BILP, and HEUR schedulers, respectively. We define Q to be the increase over PF as in the following equation:

$$Q = \frac{rate_{BILP} - rate_{PF}}{rate_{PF}} \quad \text{or} \quad \frac{rate_{HEUR} - rate_{PF}}{rate_{PF}} \tag{7}$$

Thus, Q takes $rate_{PF}$ as the baseline non-proactive scheduling performance and calculates the improvement of proactive scheduling, via either BILP or HEUR. Also, we use ΔR to denote the data rate difference between one of the proactive schedulers and the PF scheduler, i.e. the numerator of Q, and P to denote the ratio between ΔR_{HEUR} and ΔR_{BILP}. Thus, P represents how much of the gap between the non-proactive baseline (PF) and the proactive upper bound (BILP), the heuristic proactive scheduler has closed. In the following results, we present the Q and P ratios as percentages.

In Table 5, we show how the proactive schedulers compare to PF scheduler in terms of average data rate performance. Recall that all schedulers use the PF allotment so they achieve very nearly the same proportional fairness. The rows labeled "accurate" show the performance of the proactive schedulers with perfect mobility prediction and the rows labeled "naive" are the results with the simple straight-line mobility predictor. We first note that ***proactive scheduling shows very strong potential compared to non-proactive scheduling*** with more than 20% improvement over PF when optimal proactive scheduling is performed with perfect mobility prediction. We also note that ***the heuristic scheduler does quite well compared to the optimal but impractical***

Table 5
Proactive scheduling improvement over PF: Baseline case.

Scheduler	ΔR	Q	P
HEUR (accurate)	73.35 Mbps	15.45%	72.36%
BILP (accurate)	101.4 Mbps	21.35%	N/A
HEUR (naive)	29.73 Mbps	6.26%	62.38%
BILP (naive)	47.65 Mbps	10.04%	N/A

Table 6
Sum of log rates for PF, HEUR, and BILP schedulers.

Scheduler	Log fairness (accu)	Log fairness (pred)
PF	173.467	N/A
HEUR	174.736	173.997
BILP	175.178	174.304

Table 7
Average jitter for PF, HEUR, and BILP schedulers.

Scheduler	Avg jitter (accu)	Avg jitter (pred)
PF	1.1874 ms	N/A
HEUR	1.7651 ms	1.6918 ms
BILP	4.0023 ms	3.0422 ms

BILP scheduler, closing about 72% or 62% of the gap between the non-proactive and optimal proactive scheduling results. This means that, even ***the very efficient HEUR proactive scheduler still achieves significant performance improvement compared to non-proactive scheduling*** (about 15% improvement with perfect mobility prediction). However, improvements with the naive mobility predictor are somewhat smaller, from 6% to 10% over non-proactive. This demonstrates that ***the quality of mobility prediction is very important to realizing the full benefits of proactive scheduling***.

6.3. Fairness and jitter comparison on baseline case

In this subsection, we discuss what the proactive schedulers lose, if anything, in return for improved data rate.

We reiterate that, ***by design, the improved performance of the proactive schedulers reported in the prior subsection is achieved with the same proportional fairness constraint***. This is because the user time slot allotment produced by the PF scheduler was provided as an input to the proactive schedulers. To demonstrate this, we also calculated the proportional fairness by calculating the sum of log rates values for all three schedulers. The results are presented in Table 6, and show that all three schedulers have nearly identical values of the proportional fairness metric.

Another important metric is delay, usually quantified by latency or jitter. Since we are focused on wireless LAN, which is the “last hop” and latency is best measured end-to-end, we focus herein on jitter. Table 7 shows the average jitter produced by each of the different algorithms on the baseline case. The ***proactive schedulers do have increased jitter relative to the PF scheduler***. This is because some packet transmissions are delayed to push them to higher rate slots that occur after a blockage is resolved. Interestingly, this effect is more severe with the highly optimized BILP algorithm than with the heuristic proactive scheduler. Jitter values for the heuristic are increased by only about 0.5 ms compared to PF while, for BILP, they are increased by 2–3 ms. ***Nevertheless, all jitter values are in an acceptable range for real-time applications***. For example, jitter for audio and video conferencing should be less than 30 ms, while in the extreme case of on-line gaming, jitter of less than 10 ms is considered acceptable.

6.4. Impact of parameters on mobility prediction accuracy

As we have seen that the quality of mobility prediction has a significant impact on proactive scheduling performance, we now evaluate the accuracy of the naive mobility prediction algorithm. Accuracy is defined as the number of time slots where the user location is correctly predicted divided by the total number of time slots. Although this result is not directly part of the scheduler performance, it is still an important factor that affects the performance of the scheduler like other parameters. Understanding its performance can help explain how this and other factors affect the overall system performance.

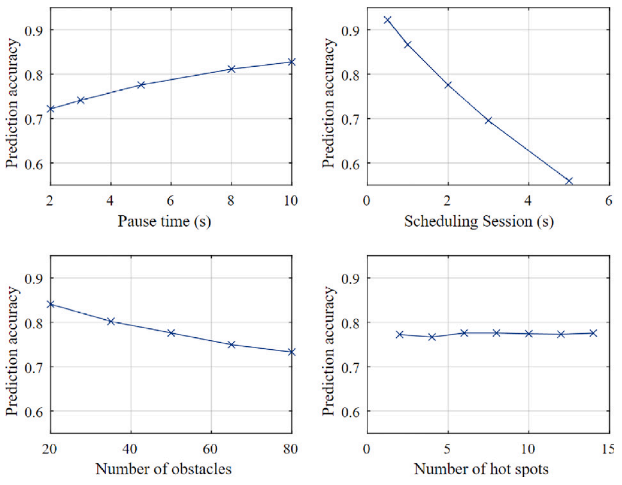

Fig. 2. Prediction accuracy vs. different parameters.

Fig. 2 depicts the impact of the pause time, scheduling session duration, number of hot spots, and number of obstacles on mobility prediction accuracy. The trends in the results agree with intuition. First, the longer the pause time, the better the accuracy, since the prediction is more likely to be correct when the user is not moving. Next, the scheduling session duration affects the prediction accuracy in an adverse fashion, with a slope that is steeper than the other variables. This is a result of it being harder to correctly predict location the longer into the future we attempt to predict. The number of obstacles also adversely affects prediction accuracy because more obstacles result in more path obstruction and thus more path diversions, which the naive mobility predictor does not attempt to predict. Lastly, the number of hot spots does not significantly affect prediction accuracy.

6.5. Impact of pause time on data rate

In this subsection, we study how different pause time durations affect the scheduler performance. Fig. 3 plots the average user data rate vs. pause time. As the pause time increases, we see that the performance of the proactive schedulers drop somewhat, despite the mobility prediction accuracy getting higher (see Fig. 2). This is due to the pause time becoming longer than the scheduling session duration, which means that if a user is paused at a hot spot with low data rate, the scheduler cannot assign them to higher performing slots within the scheduling session. Despite this, the proactive schedulers significantly outperform the non-proactive scheduler in all cases, particularly when the mobility prediction is perfectly accurate.

Table 8 shows more detailed data on scheduling performance. We again see that the heuristic scheduler does quite well compared to the optimal proactive scheduler, recovering between 62% and 77% of the benefit of the optimal scheduler. The data also confirm the results of Fig. 3 in that the benefits of proactive scheduling are reduced for longer pause times. Finally, the importance of accurate mobility prediction is clearly shown — proactive performance improvement ranges from 13% to 24% with perfect prediction while the improvement is only between 5% and 12% with naive prediction.

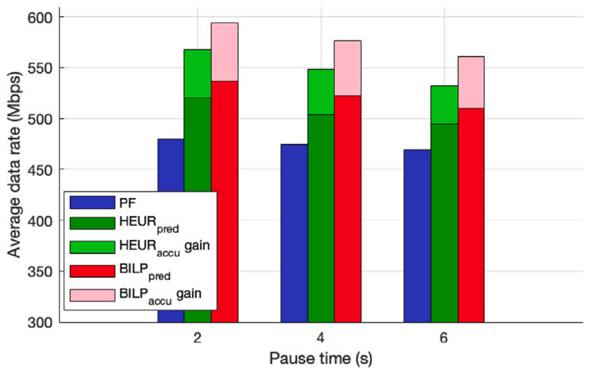

Fig. 3. Average user data rate vs. pause time.

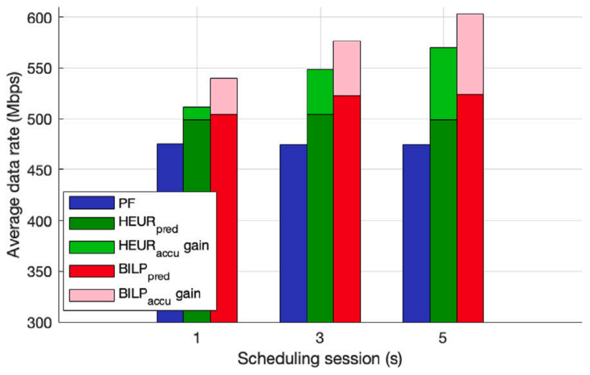

Fig. 4. Average user data rate vs. scheduling session duration.

Table 8
Proactive scheduler improvement against PF by pause time.

Pause (s)	ΔR_{BILP}	Q_{BILP}	ΔR_{HEUR}	Q_{HEUR}	P
2 (accu)	114 Mbps	23.9%	87.7 Mbps	18.27%	76.63%
4 (accu)	101 Mbps	21.4%	73.3 Mbps	15.45%	72.36%
6 (accu)	91.9 Mbps	19.6%	62.7 Mbps	13.38%	68.28%
2 (naive)	56.3 Mbps	11.8%	39.5 Mbps	8.24%	70.16%
4 (naive)	47.6 Mbps	10.0%	29.7 Mbps	6.26%	62.38%
6 (naive)	40.6 Mbps	8.66%	25.5 Mbps	5.44%	62.84%

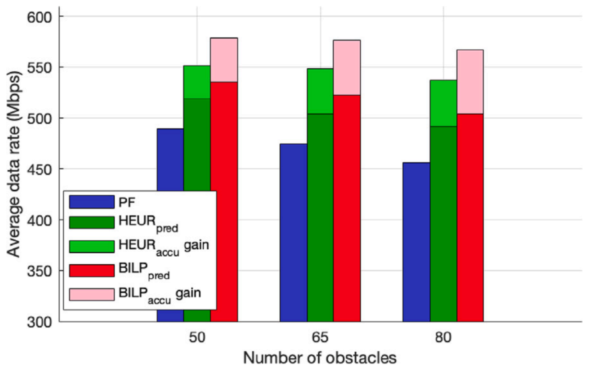

Fig. 5. Average user data rate vs. number of obstacles.

6.6. Impact of scheduling session duration on data rate

Fig. 4 shows the average user data rate plotted against different scheduling session lengths. We can see from the plot that with accurate mobility prediction, the performance of both proactive schedulers significantly increases with longer scheduling sessions. However, note that, with the naive predictor, the proactive gain is much smaller. This is explained by Fig. 2, where the slope for the scheduling session subplot is significantly steeper than the other parameters, meaning that longer scheduling sessions heavily degrade prediction accuracy with such a simple predictor.

Increasing the scheduling session length has two competing effects. Longer sessions allow for more optimized scheduling around blockage events, which tends to improve performance. However, as indicated above, prediction accuracy decreases over time, which tends to degrade performance as scheduling session length increases. A detailed study of state-of-the-art mobility predictors could reveal whether there is an optimum scheduling length that balances these competing effects. For the naive mobility predictor studied herein, that optimum session length appears to be around 3 s. However, with more sophisticated mobility predictors, we believe even longer scheduling sessions could continue to yield performance improvements. Such a detailed study is beyond the scope of this work.

Table 9 shows the detailed data for different scheduling session lengths. ***With accurate prediction and a 5 s scheduling session, optimal proactive scheduling achieves 27% higher data rate than non-proactive PF scheduling with the same fairness***. The heuristic scheduler achieves 20% higher data rate than the PF scheduler and is within 74% of the optimal proactive scheduling gain with perfect prediction. However, with naive prediction, its performance drops to only 5% better than PF and only 49% of optimal, again demonstrating the need for high-quality mobility prediction.

6.7. Impact of obstacle and hot spot density on data rate

Fig. 5 shows the average user data rate vs. the number of obstacles in the room. It can be observed that the benefit of proactive scheduling becomes greater as obstacle density increases. This is due to the users being more likely to experience blockages when more obstacles are present. While the proactive schedulers are almost able to maintain their performance with increasing obstacle density and accurate prediction, the performance of the non-proactive PF scheduler decreases fairly significantly. However, the benefits do decrease when the naive mobility predictor is used. With more obstacles, users tend to detour more often around them, which is not accounted for in the naive prediction scheme. While we could have significantly improved the naive predictor by predicting these detours, this would have made the predictor quite close to the actual mobility model. We chose not to do this, because we wanted to evaluate the range of proactive scheduling performance from a basic (and not very accurate) prediction scheme to a perfect one.

Table 9
Proactive scheduler improvement against PF by scheduling session duration.

Sched (s)	ΔR_{BILP}	Q_{BILP}	ΔR_{HEUR}	Q_{HEUR}	P
1 (accu)	63.9 Mbps	13.5%	36.3 Mbps	7.64%	56.81%
3 (accu)	101 Mbps	21.4%	73.4 Mbps	15.45%	72.36%
5 (accu)	128 Mbps	27.0%	95.1 Mbps	20.02%	74.20%
1 (naive)	29.1 Mbps	6.12%	23.9 Mbps	5.03%	82.20%
3 (naive)	47.7 Mbps	10.0%	29.7 Mbps	6.26%	62.38%
5 (naive)	49.3 Mbps	10.4%	24.2 Mbps	5.09%	49.03%

From column P in Table 10 we see, once again, that the heuristic scheduler does quite well compared to the optimal BILP scheduler, closing between 62% and 74% of the performance gap between PF and the optimal proactive result.

As the number of hot spots increases, there is a more even distribution of user locations in the room, which helps with users getting better performance on average across all three schedulers. With more hot spots, the average distance between hot spots also decreases, thereby shortening the average user movement time. Therefore, as shown in Fig. 6, the performance of all three schedulers increases with the number of hot spots. However, the increase between 6 and 8 hot spots is

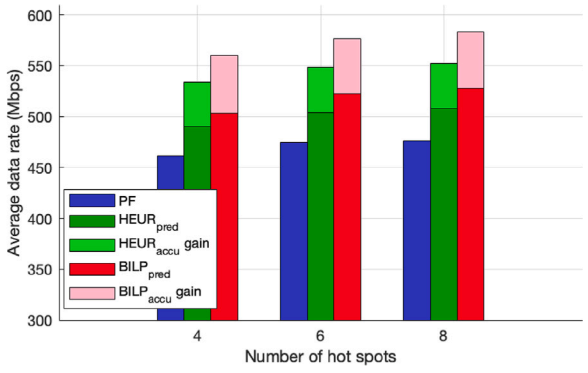

Fig. 6. Average user data rate vs. number of hot spots.

Table 10
Proactive scheduler improvement against PF by obstacle number.

Obs #	ΔR_{BILP}	Q_{BILP}	ΔR_{HEUR}	Q_{HEUR}	P
50 (accu)	89.1 Mbps	18.2%	62.0 Mbps	12.7%	69.52%
65 (accu)	101 Mbps	21.4%	73.4 Mbps	15.5%	72.36%
80 (accu)	111 Mbps	24.3%	81.5 Mbps	17.9%	73.67%
50 (naive)	45.7 Mbps	9.4%	29.4 Mbps	6.0%	64.23%
65 (naive)	47.6 Mbps	10.0%	29.7 Mbps	6.3%	62.38%
80 (naive)	47.7 Mbps	10.5%	35.2 Mbps	7.7%	73.90%

Table 11
Proactive scheduler improvement against PF by hot spot number.

Hot Spot #	ΔR_{BILP}	Q_{BILP}	ΔR_{HEUR}	Q_{HEUR}	P
4 (accu)	98.8 Mbps	21.41%	72.5 Mbps	15.73%	73.44%
6 (accu)	101 Mbps	21.35%	73.4 Mbps	15.45%	72.36%
8 (accu)	107 Mbps	22.47%	76.3 Mbps	16.03%	71.38%
4 (naive)	41.8 Mbps	9.07%	29.6 Mbps	6.42%	70.82%
6 (naive)	47.6 Mbps	10.04%	29.7 Mbps	6.26%	62.38%
8 (naive)	51.7 Mbps	10.87%	32.4 Mbps	6.81%	62.66%

smaller than between 4 and 6 hot spots, indicating that the performance is likely to converge to a peak value instead of continuing to increase with the number of hot spots.

As far as the heuristic algorithm is concerned, as the hot spots get denser, HEUR's improvement compared to PF also slightly grows, as shown in column ΔR_{HEUR} of Table 11. With perfect prediction, HEUR can achieve more than 70% of the optimal proactive scheduling gain, whereas with naive prediction, performance drops to around 62% of optimal with higher hot spot density.

6.8. *Running time of heuristic algorithm*

In Section 5.2, the running time of our heuristic scheduling algorithm was evaluated as $O(n_u \cdot n_{ts})$, where n_u is the number of users and n_{ts} is the number of time slots in a scheduling session. Since we showed in Section 6.6 that the performance of proactive scheduling with accurate mobility prediction improves as the scheduling session duration increases, we would like to ensure that the scheduler can run in real time for long scheduling sessions. To test this, we evaluated the algorithm's running time on the baseline case from Section 6.2. In that scenario, there are 20 users and 48,000 time slots in the 3 s scheduling session. With those values ($n_u = 20$, $n_{ts} = 30{,}000$), the heuristic scheduler took only 0.02126 s to compute the entire schedule, which is well within the time limit for real-time scheduling for a 3 s scheduling period.

7. Performance study with static and dynamic obstacles

In this section, we study the influence of dynamic obstacles on the performance of the BILP and HEUR schedulers. Note that there is no change needed to the proactive scheduling algorithms to handle dynamic obstacles. The difference lies in how the predicted rates, which are inputs to the algorithms, are generated, If mobility prediction is done for *all* users, then we can predict when one user will block another user in addition to predicting the blockages from static obstacles. This leads to lower predicted rates at times when user–user blockage is predicted and this data is then fed to the proactive scheduler for processing.

Table 12
Dynamic performance compared to static for baseline case.

Scheduler	Accurate prediction		Naive prediction	
	Absolute	%	Absolute	%
PF	−23.38 Mbps	−4.99%	N/A	N/A
HEUR	−5.21 Mbps	−0.95%	−15.86 Mbps	−3.19%
BILP	−4.73 Mbps	−0.82%	−16.48 Mbps	−3.21%

7.1. *Simulation settings*

In this section, since we take into consideration the size and shape of users as dynamic obstacles, we adjust the hot spot mobility model so that each hot spot is given a size of 4 m by 4 m square. When a user moves to a new hot spot, it picks a destination location following a random uniform distribution within the square. The users are modeled as cuboids of cross section size 0.3 m by 0.6 m, and a height that follows a normal distribution of 1.71 m mean and 0.1 m variance, with orientation randomly aligned with either the room length or the room width. These parameter values were chosen to model the sizes of human beings, which form the dynamic obstacles in our scenario. All other fixed variable settings follow that of Section 6.1. In this section, the term "dynamic results" refers to results obtained with both static and dynamic obstacles, which are all other users in the scenario. We analyze how dynamic obstacles affect network performance when the following parameters change: scheduling session length, user pause time, hot spot density, and user walking speed. The exact values are given by Table 3, except for the user walking speed, whose low, medium and high levels are set at 0.97, 1.34, and 1.71 m per second to reflect the range of human walking speeds.

Since the mobility model is adjusted to fit the dynamic obstacle scenarios, we cannot directly compare dynamic results to the results in Section 6. To produce comparable static-obstacle-only results, we let the users follow the same paths as with the dynamic results but we ignore user–user blockages. We refer to these as "static results" in what follows.

7.2. *Baseline case*

Here, we present results for the baseline case. Table 12 lists the absolute and percentage decrease when we take the dynamic obstacles into consideration relative to the static results' average data rates, with the rows representing the three different schedulers. Like in Section 6, we compare the rates calculated with both perfect and naive mobility prediction. It is evident from the table that the impact of dynamic obstacles on the proportional fair (PF) scheduler is greater than on the proactive schedulers. In fact, with perfect prediction, the impact of dynamic obstacles is less than 1% with proactive scheduling. Although blockages are predicted perfectly in that case, there are some fairly long user to user blockages that occur in hot spot locations when users are paused, which cannot be handled with scheduling over a few seconds scheduling session, and this results in the slight decrease for proactive scheduling even with perfect prediction. Even with naive prediction, the performance of the proactive schedulers drops less than for the PF scheduler due to the higher blockage rate with both static and dynamic obstacles and PF's inability to handle the blockages.

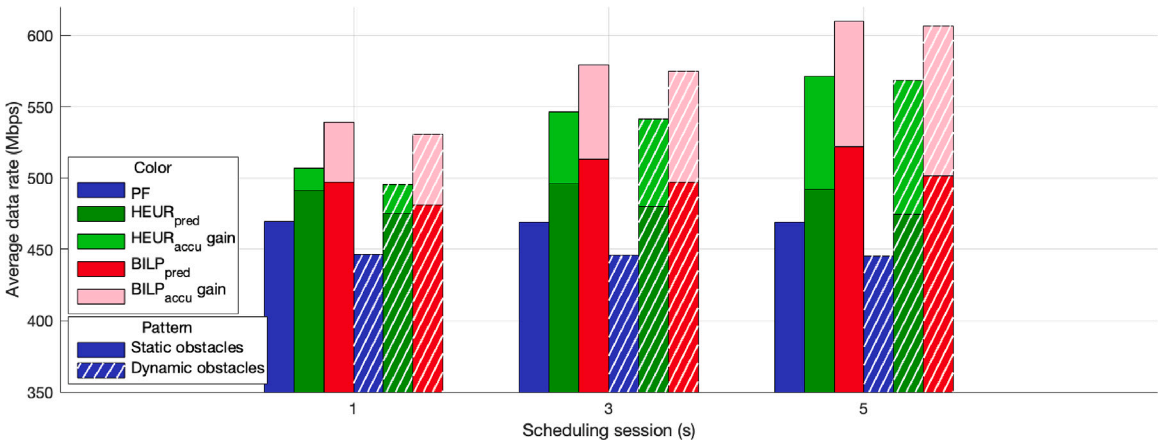

Fig. 7. Average user data rate vs. scheduling session duration.

7.3. Impact of scheduling session duration

Fig. 7 plots the average user data rate in Mbps against different scheduling session lengths. Solid colored bars denote the results of each scheduler considering only the static obstacles and the bars with white hatched lines denote the results considering both the static and dynamic obstacles. From the figure, we can see that, for both static and dynamic results, as the scheduling session lengthens, average data rate increases for both proactive schedulers with perfect mobility prediction. However, with naive prediction, when increasing the scheduling session from 3 s to 5 s, there is a noticeable decline in the average data rate. It can also be observed that the gap between naive and perfect prediction becomes larger as scheduling session duration increases, suggesting that simplistic mobility prediction is more severely affected by dynamic obstacles. Thus, having high-quality mobility prediction is even more important when taking dynamic obstacles into consideration.

Finally, we note that, over all of our different simulation scenarios, the maximum difference between proactive and non-proactive scheduling occurs with both static and dynamic obstacles and a scheduling session duration of 5 s. ***With a 5-s scheduling session and dynamic obstacles, the optimal proactive scheduler with perfect mobility prediction achieves more than 35% higher average data rate than the non-proactive PF scheduler while achieving the same fairness.*** For this case with naive mobility prediction, however, the performance improvement over PF drops to about 14%, which is still significant but not nearly as good. We believe that these results show great promise for proactive scheduling if it can leverage state-of-the-art mobility prediction techniques, particularly in the presence of dynamic obstacles.

7.4. Impact of pause time

Fig. 8 plots the average user data rate against increasing pause time. For all three algorithms, across different obstacle scenarios and prediction accuracies, average user data rate declines as pause time increases. This is due to the pause time exceeding the default scheduling session duration which is 3 s, preventing the schedulers from assigning users that pause at low rate locations to higher rate slots. Once again, we see that the PF scheduler's performance drop with dynamic obstacles is greater than those of the proactive schedulers since they can handle some of the increased blockages while PF cannot.

7.5. Impact of hot spot density

From Fig. 9, we can see that the number of hots spots does not have a large impact on the results. There is a small increase in performance for all schedulers when going from 4 to 6 hot spots but almost no change when going from 6 to 8 hot spots. The reason for lower performance with 4 hot spots and 20 users is that there can be a large number of users paused at the same hot spot at the same time, which results in a high blockage probability while users are paused. With a larger number of hot spots, user density at each hot spot is reduced, thereby lowering the probability of blockage at the hot spots.

7.6. Impact of user walking speed

Here, we modify a parameter that should have a clear effect on dynamic obstacle behavior, namely the user walking speed. There are two main effects from increasing the walking speed. First, users spend a higher percentage of time paused and, since blockages often occur at hot spots when users are paused, this increases the overall blockage rate. Second, blockages that occur when a user is moving are shorter in duration, because time spent behind obstacles is reduced.

The two effects just described explain the data shown in Fig. 10. The proactive schedulers benefit from shorter blockages along the paths, because that provides more opportunity for scheduling users in non-blocked higher-rate conditions. However, they also have to deal with the higher overall blockage rate due to a higher percentage of time spent in a paused state. This results in a slight increase when going from slow to medium walking speed but the two effects balance out when going from medium to high speed. Since the PF scheduler does not handle blockages, it benefits only slightly from the shorter path blockages but is affected significantly by the higher overall blockage rate. This causes its performance to drop noticeably for the PF scheduler at the highest walking speed.

8. Discussion and conclusions

In this paper, we presented both an optimal proactive scheduling algorithm based on integer linear programming and an efficient greedy heuristic proactive scheduling algorithm. With accurate mobility prediction, both algorithms were shown to be capable of increasing average user data rate substantially compared to non-proactive scheduling with no loss of fairness and only a small increase in jitter, which was still well within an acceptable range. Proactive performance with an extremely simplistic mobility prediction scheme was shown to still be better than with non-proactive scheduling but the performance difference in that case was not as large as with accurate mobility prediction.

The results in the paper with extremely simplistic mobility prediction and with perfect mobility prediction define the range of performance that can be expected with proactive scheduling. It can be observed from the results presented in this paper that accurate mobility

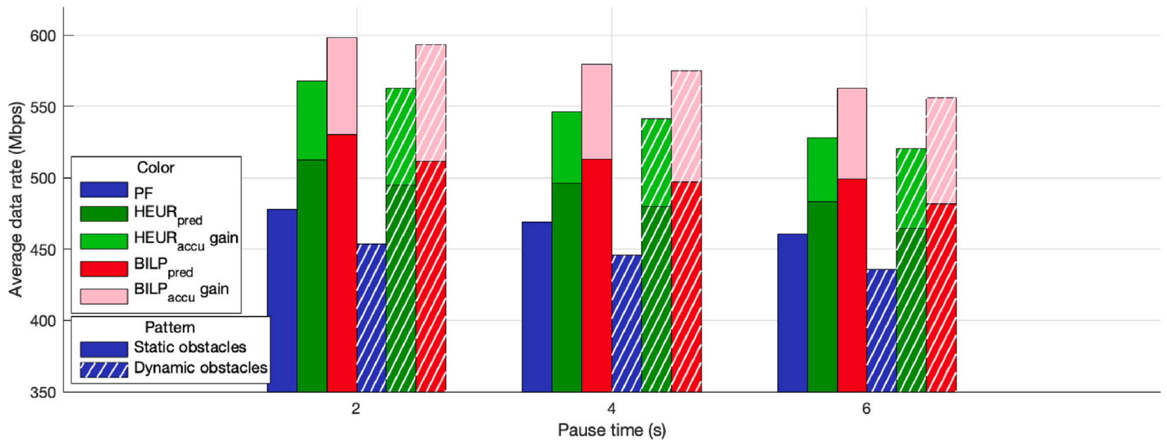

Fig. 8. Average user data rate vs. pause time.

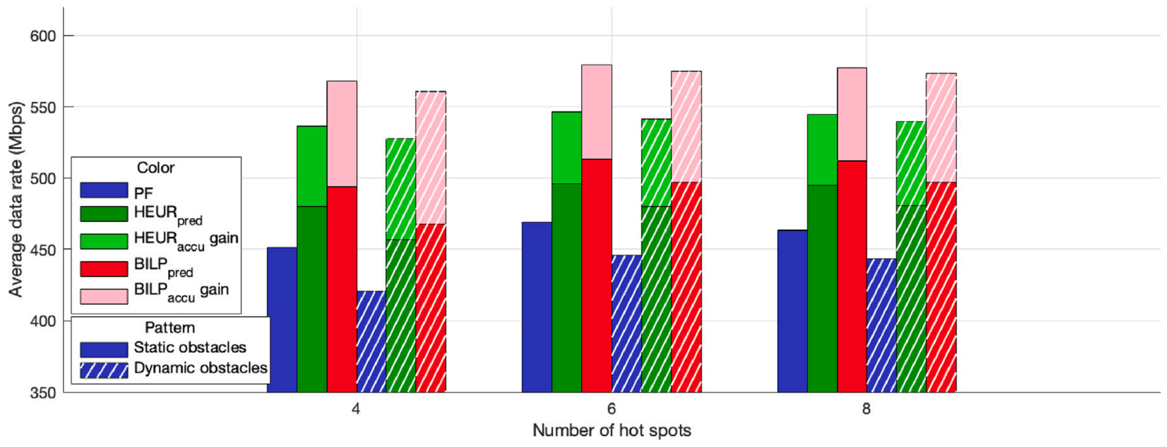

Fig. 9. Average user data rate vs. number of hot spots.

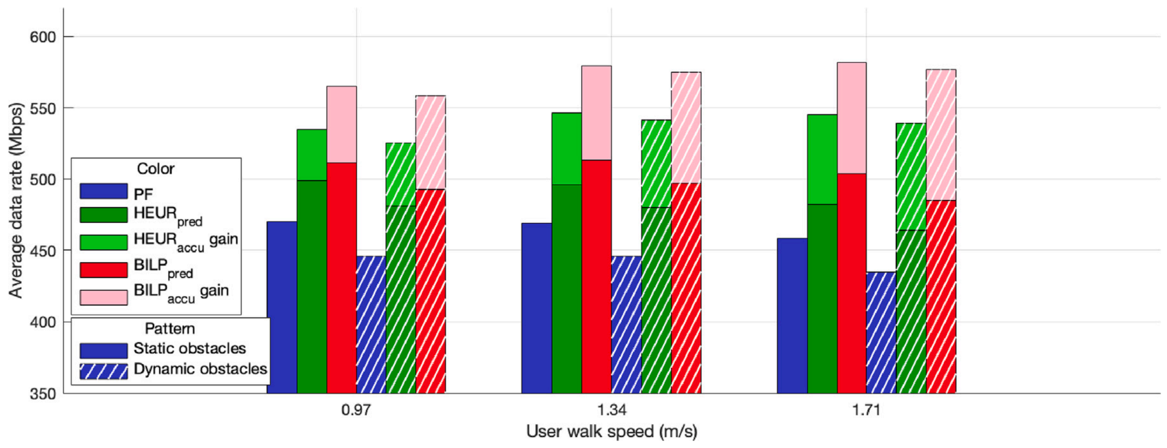

Fig. 10. Average user data rate vs. user walking speed.

prediction significantly enhances the performance gain of proactive scheduling. We believe this is well within the capability of state-of-the-art mobility predictors. One study showed that there is a 93% potential predictability in human mobility through analyzing mobile phone users' mobility patterns [21]. Trajectory-based mobility prediction has been extensively studied and can be achieved through traditional and machine learning methods [22]. Common traditional approaches include Kalman filters, Markov models and Hidden Markov Models. Representative machine learning methods include convolutional neural networks (CNN), recurrent neural networks (RNN), and long short-term memory networks (LSTM). Integrating any of these prediction methods into our approach is straightforward as the mobility prediction piece is a preprocessing step that is used in data rate prediction, and the predicted data rates then become an input to our heuristic scheduling algorithm.

An open problem is to evaluate proactive scheduling performance with state-of-the-art (but not perfect) mobility prediction. This is a challenge for the following reasons. With a synthetic mobility model such as the hot spot mobility model with obstacles used herein, one can improve mobility prediction with knowledge of the model. However, real users' movements are not dictated by a synthetic model so this is not a realistic scenario. A better approach would be to use mobility traces gathered from a real environment. However, to our knowledge, real mobility traces from an indoor WLAN environment covering a single room do not exist in the public domain. The next step for research on this topic is to gather realistic mobility traces from a single-room WLAN scenario and use them to evaluate proactive scheduling with state-of-the-art mobility predictors.

CRediT authorship contribution statement

Ang Deng: Writing – original draft, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Douglas M. Blough:** Writing – review & editing, Validation, Supervision, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was sponsored in part by the U.S. National Science Foundation under Award CCRI-2016381.

Data availability

No data was used for the research described in the article.

References

[1] R.J. Weiler, M. Peter, W. Keusgen, M. Wisotzki, Measuring the busy urban 60 GHz outdoor access radio channel, in: 2014 IEEE International Conference on Ultra-WideBand, ICUWB, 2014, pp. 166–170, http://dx.doi.org/10.1109/ICUWB.2014.6958971.

[2] G.R. MacCartney, T.S. Rappaport, S. Rangan, Rapid fading due to human blockage in pedestrian crowds at 5G millimeter-wave frequencies, in: GLOBECOM 2017 - 2017 IEEE Global Communications Conference, 2017, pp. 1–7, http://dx.doi.org/10.1109/GLOCOM.2017.8254900.

[3] G.R. MacCartney, S. Deng, S. Sun, T.S. Rappaport, Millimeter-wave human blockage at 73 GHz with a simple double knife-edge diffraction model and extension for directional antennas, in: 2016 IEEE 84th Vehicular Technology Conference, VTC-Fall, 2016, pp. 1–6, http://dx.doi.org/10.1109/VTCFall.2016.7881087.

[4] F. Firyaguna, A. Bonfante, J. Kibilda, N. Marchetti, Performance evaluation of scheduling in 5G-mmWave networks under human blockage, 2020, CoRR abs/2007.13112, arXiv:2007.13112, URL https://arxiv.org/abs/2007.13112.

[5] H.J. Bang, T. Ekman, D. Gesbert, Channel predictive proportional fair scheduling, IEEE Trans. Wireless Commun. 7 (2) (2008) 482–487, http://dx.doi.org/10.1109/TWC.2008.060729.

[6] R. Margolies, A. Sridharan, V. Aggarwal, R. Jana, N.K. Shankaranarayanan, V.A. Vaishampayan, G. Zussman, Exploiting mobility in proportional fair cellular scheduling: Measurements and algorithms, IEEE/ACM Trans. Netw. 24 (1) (2016) 355–367, http://dx.doi.org/10.1109/TNET.2014.2362928.

[7] L. Shen, T. Wang, S. Wang, Proactive proportional fair: A novel scheduling algorithm based on future channel information in OFDMA systems, in: 2019 IEEE/CIC International Conference on Communications in China, ICCC, 2019, pp. 925–930, http://dx.doi.org/10.1109/ICCChina.2019.8855928.

[8] J. Bao, T. Shu, H. Li, Handover prediction based on geometry method in mmWave communications: A sensing approach, in: 2018 IEEE International Conference on Communications Workshops, ICC Workshops, 2018, pp. 1–6.

[9] M.G. Dogan, M. Cardone, C. Fragouli, Proactive resilient transmission and scheduling mechanisms for mmWave networks, 2022, arXiv:2211.09307.

[10] Y. Oguma, T. Nishio, K. Yamamoto, M. Morikura, Proactive handover based on human blockage prediction using RGB-D cameras for mmWave communications, IEEE Trans. Mob. Comput. E99-B (6) (2016) 1734–1744.

[11] M. Zarifneshat, L. Xiao, J. Tang, Learning-based blockage prediction for robust links in dynamic millimeter wave networks, in: 2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking, SECON, 2019, pp. 1–9, http://dx.doi.org/10.1109/SAHCN.2019.8824987.

[12] M. Alrabeiah, A. Alkhateeb, Deep learning for mmWave beam and blockage prediction using sub-6 GHz channels, IEEE Trans. Commun. 68 (9) (2020) 5504–5518, http://dx.doi.org/10.1109/TCOMM.2020.3003670.

[13] F. Göttsch, M. Kaneko, Deep learning-based beamforming and blockage prediction for sub-6-GHz/mmWave mobile networks, in: GLOBECOM 2020 - 2020 IEEE Global Communications Conference, 2020, pp. 1–6, http://dx.doi.org/10.1109/GLOBECOM42002.2020.9322404.

[14] Y. Liu, D.M. Blough, Environment-aware link quality prediction for millimeter-wave wireless LANs, in: Proc. ACM International Symposium on Mobility Management and Wireless Access, 2022, pp. 1–10.

[15] R. Amiri, S. Yerramalli, T. Yoo, M. Hirzallah, M. Zorgui, R. Prakash, X. Zhang, Indoor environment learning via RF-mapping, IEEE J. Sel. Areas Commun. 41 (2023) 1859–1872.

[16] K. Sato, K. Suto, K. Inage, K. Adachi, T. Fujii, Space-frequency-interpolated radio map, IEEE Trans. Veh. Technol. 70 (2021) 714–725.

[17] G. Brar, D.M. Blough, P. Santi, Computationally efficient scheduling with the physical interference model for throughput improvement in wireless mesh networks, in: Proc. ACM International Conference on Mobile Computing and Networking, 2006, pp. 2–13.

[18] R. Chackochan, S. Dhanasekaran, A. Sunny, Asynchronous distributed greedy link scheduling in multihop wireless networks, IEEE Trans. Veh. Technol. 67 (2018) 10166–10170.

[19] Y. Shao, Q. Cao, S.C. Liew, H. Chen, Partially observable minimum-age scheduling: The greedy policy, IEEE Trans. Commun. 70 (2022) 404–418.

[20] Y. Liu, Enhanced mmWave LAN ns-3 simulator, 2022, https://github.com/yuchen-sh/mmWave-WLAN-802.11ad/tree/master.

[21] C. Song, Z. Qu, N. Blumm, A.-L. Barabási, Limits of predictability in human mobility, Science 327 (5968) (2010) 1018–1021.

[22] A. Rudenko, L. Palmieri, M. Herman, K.M. Kitani, D.M. Gavrila, K.O. Arras, Human motion trajectory prediction: A survey, Int. J. Robot. Res. 39 (8) (2020) 895–935.