IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 25, 2020, accepted November 6, 2020, date of publication November 16, 2020,
date of current version December 15, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3038358

Multi-Channel Joint Forecasting-Scheduling for the Internet of Things

VOLKAN RODOPLU[1], (Member, IEEE), MERT NAKIP[2], ROOZBEH QORBANIAN[3], AND DENIZ TÜRSEL ELIIYI[4]

[1]Department of Electrical and Electronics Engineering, Yaşar University, 35100 İzmir, Turkey
[2]Institute of Theoretical and Applied Informatics, Polish Academy of Sciences (PAN), 44100 Gliwice, Poland
[3]Luxembourg Centre for Logistics and Supply Chain Management, University of Luxembourg, 1359 Luxembourg City, Luxembourg
[4]Department of Industrial Engineering, Izmir Bakırçay University, 35330 İzmir, Turkey

Corresponding author: Volkan Rodoplu (volkan.rodoplu@yasar.edu.tr)

This work was supported by the Project Support Commission of Yaşar University within the scope of the Scientific Research Project "Scheduling Algorithms for Wireless Communication" under Grant BAP060.

ABSTRACT We develop a methodology for Multi-Channel Joint Forecasting-Scheduling (MC-JFS) targeted at solving the Medium Access Control (MAC) layer Massive Access Problem of Machine-to-Machine (M2M) communication in the presence of multiple channels, as found in Orthogonal Frequency Division Multiple Access (OFDMA) systems. In contrast with the existing schemes that merely react to current traffic demand, Joint Forecasting-Scheduling (JFS) forecasts the traffic generation pattern of each Internet of Things (IoT) device in the coverage area of an IoT Gateway and schedules the uplink transmissions of the IoT devices over multiple channels in advance, thus obviating contention, collision and handshaking, which are found in reactive protocols. In this paper, we present the general form of a deterministic scheduling optimization program for MC-JFS that maximizes the total number of bits that are delivered over multiple channels by the delay deadlines of the IoT applications. In order to enable real-time operation of the MC-JFS system, first, we design a heuristic, called Multi-Channel Look Ahead Priority based on Average Load (MC-LAPAL), that solves the general form of the scheduling problem. Second, for the special case of identical channels, we develop a reduction technique by virtue of which an optimal solution of the scheduling problem is computed in real time. We compare the network performance of our MC-JFS scheme against Multi-Channel Reservation-based Access Barring (MC-RAB) and Multi-Channel Enhanced Reservation-based Access Barring (MC-ERAB), both of which serve as benchmark reactive protocols. Our results show that MC-JFS outperforms both MC-RAB and MC-ERAB with respect to uplink cross-layer throughput and transmit energy consumption, and that MC-LAPAL provides high performance as an MC-JFS heuristic. Furthermore, we show that the computation time of MC-LAPAL scales approximately linearly with the number of IoT devices. This work serves as a foundation for building scalable JFS schemes at IoT Gateways in the near future.

INDEX TERMS Forecasting, scheduling, massive access, IoT, M2M communication.

I. INTRODUCTION

It is expected that the majority of the connections on the Internet in the near future will be between machines that operate without human intervention [1]. This new paradigm, which is referred to as Machine-to-Machine (M2M) communication, typically originates at an Internet of Things (IoT) device that reports data and ends at an Artificial Intelligence (AI) algorithm at a server that makes intelligent decisions based on the report. While Fourth Generation (4G) cellular systems were designed to support Human-to-Human (H2H), Machine-to-Human (M2H) and Human-to-Machine (H2M) traffic [2], Fifth Generation (5G) and future wireless systems must effectively address the new challenges [3] brought by M2M communication.

A significant challenge of M2M communication is the Massive Access Problem, namely the problem of granting wireless access to a massive number of IoT devices.

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Feng.

It is estimated that in the near future, approximately 5000 - 35,000 smart utility meters will fall in the coverage of a single base station [4]. Since utility meters constitute merely a subclass of the wide range of IoT devices, the access attempts of the entire set of IoT devices to the wired infrastructure are expected to lead to Physical Random Access Channel (PRACH) overload [2], [3] in current cellular systems. As a result, novel approaches are required in order to provide wireless access to such a massive number of IoT devices while satisfying the diverse Quality of Service (QoS) requirements of IoT applications.

While all of the articles [5]–[20] on the Massive Access Problem have modeled the traffic generation pattern of each IoT device by random arrivals, recent work [21] has shown that machine and deep learning schemes are able to predict the future traffic generation patterns of IoT devices in a forecasting system that minimizes the symmetric Mean Absolute Percentage Error (sMAPE). In [22] and [23], this observation was utilized in the design of a Joint Forecasting-Scheduling (JFS) system, in which an IoT Gateway forecasts the future traffic generation pattern of each of the IoT devices in its coverage area and schedules the traffic of these devices in advance over a scheduling window. JFS has a significant advantage over schemes that react to current traffic demand in that it obviates contention, collision and handshaking.

The main contribution of this article is the design of a *multi-channel* JFS system (abbreviated as MC-JFS),[1] which is expected to be implemented at an IoT Gateway such that the system meets the delay constraints of a diverse set of IoT applications. (We note that both [22] and [23] develop *single-channel* JFS, whereas the development of a framework for JFS over *multiple* channels is the focus of this work.) Our MC-JFS system is comprised of a Multi-Layer Perceptron (MLP) forecaster and a multi-channel scheduler. Since many practical systems utilize multiple physical channels at the physical layer, e.g. as found in Orthogonal Frequency Division Multiple Access (OFDMA) systems, it is essential that real-time, multi-channel scheduling techniques that operate in tandem with real-time forecasting be developed in order to enable the practical deployment of JFS in next-generation IoT gateways. A major goal of this work is the design of deterministic multi-channel scheduling techniques that utilize point estimates of the traffic generation pattern of each IoT device in order to achieve a high system throughput as well as low energy consumption.

In this work, first, we formulate the uplink scheduling optimization program over multiple channels for the general case. Second, we develop a two-step algorithm that finds an optimal solution of this optimization program. Third, for the case of identical channels, we show that the two-step algorithm over multiple channels can be reduced such that its optimal solution can be computed in real time. Fourth, in order to solve the general form of the multi-channel scheduling optimization program in real time, we develop a heuristic, called Multi-Channel Look Ahead Priority based on Average Load (MC-LAPAL).

Based on the above mathematical framework, we compare the network performance of our MC-JFS system against Multi-Channel Reservation-based Access Barring (MC-RAB) and Multi-Channel Enhanced Reservation-based Access Barring (MC-ERAB), which serve as benchmark reactive protocols. Our results show that MC-JFS outperforms both MC-RAB and MC-ERAB with respect to uplink cross-layer throughput and transmit energy consumption, and that MC-LAPAL achieves high performance as an MC-JFS heuristic. Finally, we show that the computation time of MC-LAPAL scales approximately linearly with the number of IoT devices.

The rest of this paper is organized as follows: In Section II, we discuss the related work in this area. In Section III, we review the basic assumptions, the system design as well as the forecasting scheme for joint forecasting-scheduling. In Section IV, we present our mathematical framework for MC-JFS. In Section V, we present our results. In Section VI, we present our conclusions and directions for future work.

II. RELATIONSHIP TO THE STATE OF THE ART

In this section, we relate our work to the articles in the literature in six respects: (1) We contrast our approach with the reactive schemes that have been proposed to solve the massive access problem. (2) We explain the differences between this work and the few proactive MAC-layer techniques that have appeared in the literature, albeit for H2H communication. In addition, we highlight the differences between this work and the recent results in [21], [22] and [23]. (3) We contrast our work with those that have also used network traffic forecasting, albeit on *aggregate* traffic patterns. (4) We contrast the methodological contributions of our work with related, recently proposed scheduling techniques. (5) We state the relationship of this work to Multiple Input Multiple Output (MIMO) and to Non-Orthogonal Multiple Access (NOMA) networks. (6) We contrast our work with joint forecasting-scheduling schemes that have appeared outside the context of IoT and networking.

First, there is a significant difference between this work and the reactive schemes [5]–[20] that have been proposed as solutions to the Massive Access Problem.[2] While these reactive schemes react to the current traffic demand, modeled via random arrivals, JFS forecasts the traffic generation pattern of each individual IoT device and schedules the uplink traffic of all of the IoT devices over multiple channels *in advance* based on these forecasts.

We shall now distinguish our work in this paper further from particular reactive schemes that have been proposed for both industrial IoT networks as well as for cellular systems. In [24], a ''slot stealing'' MAC-layer protocol was proposed

[1] We emphasize that this article constitutes fundamental work and our design does *not* conform to the LTE or IEEE 802.11 standards. Our goal is to present work that has the potential to impact the evolution of wireless systems beyond the existing standards.

[2] For a classification of these past reactive schemes, see [23].

for industrial wireless sensor networks, in which aperiodic, time-critical applications are allowed to steal slots on the fly from periodic, non-critical applications whose slots have been scheduled a priori. However, no forecasting takes place for aperiodic data. In contrast, in this paper, we predict the amount of periodically occurring traffic as well as the amount and timing of aperiodic traffic (albeit with laxer delay constraints than in [24]) via machine learning. Furthermore, we use these predictions to schedule the traffic of *all* IoT devices in the coverage area of an IoT Gateway in advance.[3]

In order to solve the massive access problem, in [25], the authors devised a scheme in which the base station probes the number of colliding devices, and based on the result, divides the colliding devices into groups and pushes the traffic of these devices onto an access queue. The performance is measured via the average access delay. Similarly, Reference [26] uses the average blocking probability as well as the average access delay to characterize performance based on a random access traffic model. Reference [27] uses probability of successful access as the main performance metric for M2M traffic in the presence of H2H traffic. Reference [28] proposes the use of virtual preambles to distinguish IoT devices and characterizes the access performance via the number of retrials required for successful access in addition to the above measures. In contrast with these articles that address *only* access, our focus in this paper is the design of *joint* access and scheduling schemes for massive IoT.

In [29] and [30], joint access control and resource allocation schemes were proposed for massive access of M2M devices. While both the access and data scheduling aspects have been handled in these works, the models and the simulation results therein are based on random access and do not utilize any forecasting of M2M traffic. In contrast, in this paper, our MC-JFS scheme utilizes forecasting of M2M traffic in order to enable joint access and resource allocation for all IoT devices in a coverage area. In [31], a comparison of collision-free and contention-based access protocols for IoT is presented; however, the treatment there is based on random generation of small packets by IoT devices and does not investigate the predictability of M2M traffic.

Second, prior to the development of JFS, only a few works [32]–[34] used predictions of the traffic from individual devices for MAC-layer scheduling. The main differences between these articles and JFS are as follows: (1) While these articles have addressed only Human-to-Human (H2H) applications, we focus entirely on M2M traffic for IoT in this paper. (2) Our methodology differs significantly from [32] and [33] in that we develop deterministic scheduling techniques while they use probabilistic scheduling. (3) While these papers focus on application-layer flows without any physical layer modeling, our emphasis is on the MAC-layer scheduling in the presence of multiple channels at the physical layer (as in OFDMA systems).

For M2M communication, the recent results in [21] have indicated that machine and deep learning schemes can be used to forecast the traffic generation pattern of an IoT device to achieve a relatively high accuracy in the sMAPE metric. However, the effects of these results on MAC-layer scheduling were not explored.

Reference [22] presented a comparison of the network throughput of JFS under three forecasting models: Long-Short Term Memory (LSTM), Multi-Layer Perceptron (MLP), and Autoregressive Integrated Moving Average (ARIMA). The key differences between this reference and the current work are as follows: (1) Whereas [22] assumes a single channel, the current work develops a novel optimization framework for the scheduling problem over *multiple* channels. (2) In [22], a simple scheduling heuristic, named Priority based on Average Load (PAL), was developed for the single-channel case. In contrast, in the current work, a novel scheduling heuristic, called MC-LAPAL, is developed in order to solve the scheduling problem over *multiple* channels. (3) Whereas [22] compares the network throughput of JFS across distinct forecasting schemes, the major focus of the current work is to evaluate both the network throughput and the energy consumption of multi-channel JFS under novel scheduling techniques while keeping the forecasting scheme fixed.[4]

Reference [23] developed a multi-scale algorithm for JFS. The key differences between this reference and our work are as follows: (1) Reference [23] assumes that there is a single channel for scheduling. In contrast, JFS over multiple channels is the main emphasis of our current work. (2) The multi-scale algorithm of Reference [23] represents the past traffic pattern of each IoT device at multiple time scales in order to enable accurate forecasting of its future traffic over successively longer time windows. In contrast, in this work, no such multi-scale approach is utilized; instead, our emphasis is on the development of JFS over multiple channels.[5] (3) While [23] uses load balancing (at all scales except the lowest time scale) and optimal scheduling (at the lowest time scale), the current work develops a novel heuristic, called MC-LAPAL, that is targeted at scheduling in the presence of multiple channels.

Third, we distinguish our work from those articles that have used network traffic forecasting. Reference [35] provides a systematic survey of the recent work [36]–[43] in this area. The key difference between all of these articles and our work is that while they focus on predicting *aggregate* traffic metrics (e.g. link load and traffic volume), we forecast the traffic generation pattern of each IoT device, which is required in order to be able to schedule in advance the uplink traffic of all IoT devices in a given coverage area.

[3]Our work does *not* handle emergency and event-triggered traffic types, which are addressed in [24].

[4]We shall compare the performance of distinct forecasting schemes for multi-channel JFS in our future work.

[5]We plan to develop a multi-scale, multi-channel JFS system that combines the multi-scale algorithm of [23] and the multi-channel techniques of the current paper in our future work.

We now focus on further articles on network traffic forecasting in order to contrast each one with our work: By utilizing Reinforcement Learning (RL) for M2M traffic forecasting, Reference [44] proposes to schedule the delay-tolerant IoT traffic to the off-peak hours of traditional (e.g. H2H) traffic. Similarly, Reference [45] devises a scheme to delay the uplink transmissions of IoT devices to minimize the peak data rate demand. In this reference, the traffic generation patterns of individual IoT devices are modeled as random, which stands in contrast with our work in which the traffic generation pattern of each IoT device is predicted. By utilizing Q-learning (a particular RL technique), Reference [46] proposes a smart congestion control scheme for delay-tolerant networks. Reference [47] separates traffic into delay-sensitive and delay-tolerant classes and devises an admission control scheme for M2M communication. Their model aggregates all delay-tolerant requests by routing them to a single low-priority queue, aiming to reduce the number of access requests from individual IoT devices. The main difference between all of these past articles and our work is that while their emphasis is on minimizing network congestion, we focus on the access and MAC-layer scheduling problem of individual IoT devices. While these past works take the traffic generation patterns of individual IoT devices as random and focus on their *aggregate* impact on network congestion, our work focuses on predicting the traffic from *individual* IoT devices and enabling their uplink access in a collision-free manner.

Fourth, we contrast the methodological contributions of our work with those in the recent literature on scheduling techniques. Reference [48] develops a resource management technique based on deep reinforcement learning. Associating the resources with the channels in our work, we note that the experiment presented in [48] uses only two resources and takes 80 s per iteration to converge in a multi-threaded implementation on a 24-core CPU server. In contrast, our application is targeted at OFDMA, for which the number of channels (i.e. resources) in practical systems is typically much larger than 2. In order to focus on scalable designs with manageable time complexity, we have chosen to focus on the development of scheduling techniques that do not utilize reinforcement learning in this paper.

Reference [49] presents a dynamic inter-channel resource allocation technique for massive M2M control signaling storm mitigation. Their model calculates the demand on the control and data channels and re-allocates the resources among these two types of channels. This demand is calculated based on the prediction of the number of devices whose data packet size is less than 1 kB; however, the traffic generation pattern of each individual device is *not* predicted. In contrast, in our model, the IoT Gateway allocates the wireless channel resources based on the prediction of the traffic generation pattern of each device in its coverage area.

Fifth, we note that we focus on MAC-layer scheduling over a set of parallel channels in this paper and present our simulation results for OFDMA as an example of a system that utilizes parallel channels over the frequency domain. However, since our scheduling techniques are general, they can potentially be applied to MIMO systems [50] in which parallel spatial channels are formed, and to Non-Orthogonal Multiple Access (NOMA) systems [51]–[54] in which users are placed at distinct power levels and separated via successive interference cancellation at the receiver (in power-domain NOMA) or via placement onto different codes (in code-domain NOMA). We emphasize that the general MAC-layer scheduling techniques in this paper are applicable to the extent that in each of these systems, a set of parallel channels can be established at the Physical Layer. The main differences between the techniques presented in this paper and the above recent work on NOMA are as follows: (1) Our emphasis is on MAC-layer scheduling while the above works utilize a cross-layer design for optimizing the usage of resources at the physical layer, and (2) the above works do not forecast the future traffic generation patterns of IoT devices, whereas the performance of joint forecasting-scheduling is our focus in this paper.

Sixth, outside the context of IoT and networking, Reference [55] employed parametric forecasting for call-center workforce scheduling. Furthermore, Reference [56] used probabilistic forecasts to schedule renewable energy sources. Besides the respective contexts in which the techniques were developed, each of which is significantly different from ours, the key methodological difference between these works and ours is that they use probabilistic forecasting models, whereas we use machine learning based forecasting that produces point estimates. In addition, in contrast with [55], which develops a stochastic programming model for scheduling, we use fast, deterministic scheduling based on the point estimates.

III. REVIEW OF THE BASIC ASSUMPTIONS, SYSTEM DESIGN AND FORECASTING SCHEME FOR JOINT FORECASTING-SCHEDULING

In this section, we review the basic assumptions, the system design as well as the forecasting scheme for JFS, while specializing the discussion to MC-JFS, wherever applicable. The contents of this section constitute the preliminaries that are required for the development of multi-channel scheduling techniques, which will be described in the next section. A list of the mathematical symbols in the order in which they appear in this paper is shown in Table 1.

A. ASSUMPTIONS

In our wireless architecture model, a set of IoT devices are assumed to be located in the coverage area of a single IoT Gateway. The IoT Gateway is denoted by G, and the set of IoT devices is denoted by $\mathcal{N}$. We let N denote the cardinality of $\mathcal{N}$. We shall denote the coverage area of G by $\mathcal{C}_G$. We assume that each IoT device (called ''device'' for short) remains in $\mathcal{C}_G$ at all times. (This includes both the case of static (that is, non-moving) devices in $\mathcal{C}_G$ as well as the case

TABLE 1. List of symbols.

Symbol	Meaning
G	Gateway
$\mathcal{N}$	Set of IoT devices
N	Cardinality of $\mathcal{N}$
$\mathcal{C}_G$	Coverage area of G
Δ_j	Duration of the uplink MAC-layer delay constraint of burst j
$x_i[t]$	Number of bits generated by device i at discrete time t
η	Uplink cross-layer throughput
T_{sch}	Duration of the scheduling window
K_i	Maximum value of k for k-step ahead prediction of the traffic generation pattern of device i
n_0^i	Number of past inputs into the forecaster of device i
E_i	Number of hidden layers in the Multi-Layer Perceptron forecasting architecture for device i
n_e^i	Number of neurons in hidden layer e of the Multi-Layer Perceptron forecasting architecture for device i
r_j	Generation time of burst j
d_j	Deadline of burst j
τ_{MAC}	Duration of a single MAC-layer slot
$\tilde{R}^{mi}$	Data rate at which device i can transmit a burst on channel m
R_{mj}	Data rate at which burst j can be transmitted on channel m
$\mathcal{J}$	Set of all of the bursts generated by the devices in $\mathcal{N}$ on the scheduling window
$\mathcal{M}$	Set of all of the channels that are available for scheduling
a_j	Total number of bits in burst j
C_{mj}	Capacity of channel m for burst j
R_j	Common data rate for each burst j across all of the channels (for the case of identical channels)
C_j	Common MAC-layer slot capacity for each burst j across all of the channels (for the case of identical channels)
$\tilde{a}_j[t]$	Number of bits of burst j that have not yet been scheduled by the beginning of the current slot t
$\tilde{\Delta}_j[t]$	Duration that remains, measured at the beginning of the current slot t, until the beginning of slot d_j
$\langle \mathcal{M}_j \rangle$	Sequence of channels that have been sorted in the descending order with respect to R_{mj}
M	Total number of channels
$\mathbf{C}$	Capacity matrix
p_{RAB}	Access probability in the MC-RAB protocol
T_i	Traffic generation period of device i
$h_i[l]$	Complex channel gain of tap l for the uplink of device i
$\bar{a}_i[l]$	Real part of $h_i[l]$
$\bar{b}_i[l]$	Imaginary part of $h_i[l]$
κ	Coefficient that multiplies the normalized variance of each of the real and imaginary components of a single tap in the wireless channel model
L	Total number of channel taps in the wireless channel model
$\langle \tilde{h}_{mi} \rangle$	Sequence of complex gains across all of the channels (each indexed by m) of device i
P	Total transmit power of each of the devices in $\mathcal{N}$ over the entire set of M channels
$\tilde{R}_{mi}$	Data rate on channel m for device i
$\tilde{W}_m$	Bandwidth of channel m
N_o	Value of the single-sided power spectral density of the white noise at the receiver of Gateway G
$\tilde{C}^{mi}$	MAC-layer capacity of channel m for device i
$\mathcal{E}$	Ratio of the total transmit energy consumption of all devices in $\mathcal{N}$ to the total number of bits in successfully delivered bursts on the entire slot-channel grid
J	Total number of bursts over the entire scheduling window ($J \equiv \lvert\mathcal{J}\rvert$)
V	Ratio of T_{sch} to τ_{MAC}

of mobile devices that move within $\mathcal{C}_G$ at all times.[6]) The goal of each device *i* in $\mathcal{N}$ is to send its traffic directly to *G* on the wireless link that exists from *i* to *G*.

[6]The latter case may correspond, e.g. to devices in the coverage area of a single IoT Gateway in a smart factory. An extension of JFS to the case in which mobile devices move into or out of $\mathcal{C}_G$ is addressed in Section VII of [23].

Each IoT device is assumed to generate its traffic in bursts, where a "burst" *j* is defined, following [23], to be the minimal set of bits that correspond to *j* such that the failure to deliver any one of the bits of burst *j* to *G* causes the delivery of the remaining bits of the burst to be useless from the perspective of the application layer. The bit representation at the physical layer of each of the following application-layer

data constitutes an example of a burst: A single reading from a smart meter, a single GPS coordinate of a truck in fleet management applications, and a single bit that indicates that a smart bin is full in smart cities.

This paper is based on the assumption that a cross-layer design is used in which the beginning of each burst is marked by a preamble at the MAC layer. While this would result in unacceptable overhead for traditional H2H applications, since IoT devices typically send small bursts at sparse intervals, communication of this high-level information to the MAC layer is viable for IoT. A key advantage of this cross-layer design, which we shall exploit, is that no resources will be spent on sending parts of bursts in scheduling; either all of the bits in a burst are scheduled to be transmitted, or no bits of that bursts are transmitted at all.

We assume that for each IoT application, there exists an end-to-end delay budget.[7] As part of this delay budget, each burst j, generated by an IoT device, has an uplink MAC-layer delay constraint from the device to the IoT Gateway. The duration of this uplink MAC-layer delay constraint shall be denoted by Δ_j.

In this work, each uplink MAC-layer slot (which shall be referred to as a "slot") on the wireless link from the IoT device to the IoT Gateway G shall be denoted by t. Furthermore, the uplink channels of all of the devices to G are assumed to have been synchronized at G at the resolution of a slot. As in [22] and [23], the "traffic generation pattern" of device i is defined to be the collection each element of which is the number of bits generated by device i during each slot t. The traffic generation pattern of device i will be denoted by $\{x_i[t]\}$, which is a set indexed by t.[8]

If all of the bits of burst j have been transmitted to G in a collision-free manner such that the Δ_j delay constraint is satisfied, that burst is said to have been "successfully delivered." As in [22] and [23], the uplink cross-layer throughput η from $\mathcal{N}$ to G is defined to be the ratio of the total number of bits in successfully delivered bursts to the total number of bits of traffic generated by the devices in $\mathcal{N}$ over a single scheduling window.

We now state our assumptions regarding the Physical (PHY) layer. We emphasize that throughout this paper, only the coverage area of a single IoT Gateway is considered. Inter-cell interference [57] that would result, in our case, from adjacent IoT Gateways is not modeled. Furthermore, we assume a collision-based multi-channel model in which a collision may be experienced at the IoT Gateway in any slot-channel pair. (As shall be detailed in the next section, our MC-JFS system will schedule the transmissions free of collisions in this slot-synchronized system; hence, there will be no intra-cell interference for MC-JFS. However, the reactive protocols against which we compare MC-JFS are susceptible to collisions.) For our simulations, we shall specify the detailed PHY layer channel model, based on OFDM, for the uplink transmission from each device to the IoT Gateway in Section V-A2.

B. SYSTEM DESIGN

In our system, a multi-channel joint forecaster-scheduler, which resides at the IoT Gateway G, is comprised of a Forecasting Module and a Scheduling Module, as shown in Fig. 1. In this figure, the Forecasting Module forecasts the traffic generation pattern $\{x_i[t]\}$ of each IoT device $i \in \mathcal{N}$ over the scheduling window of duration T_{sch}. Then, the Scheduling Module schedules the devices' uplink transmissions in a collision-free manner *in advance* over multiple channels on this window based on the output of the Forecasting Module. (This process is repeated over successive scheduling windows.[9])

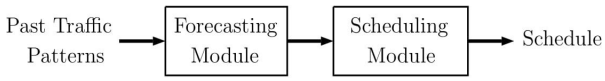


FIGURE 1. Main idea behind the joint forecasting-scheduling system that resides at an IoT Gateway: The past traffic generation patterns of all the IoT devices in the coverage area of the IoT Gateway are fed into the Forecasting Module. The forecasts are subsequently used to schedule the future traffic of all of the devices in advance over a scheduling window. This paper focuses on the development of multi-channel scheduling techniques for the Scheduling Module that appears in this figure.

Throughout this paper, Frequency Division Duplexing (FDD) is assumed to be utilized at the physical layer. In our design, device i is informed by G on a broadcast channel on the downlink as to what uplink resources have been allocated to i for the next burst of i that has been forecast by the Forecasting Module. Furthermore, each device i compresses its actual traffic generation pattern since the last time it communicated to G and appends the result whenever it transmits to G on the uplink. Subsequently, G utilizes $\{x_i[t]\}$ in forecasting the future values of the traffic generation pattern of device i.

Although the emphasis of this paper is on the MAC layer, we distinguish two possible PHY-layer designs that may accompany the MAC layer algorithms developed in this paper. First, in a power-adaptive design, the data rate from each device i to G is kept constant as a function of time by continuously varying the uplink transmit power of i based on the PHY-layer channel feedback from G on the downlink. In this case, MC-JFS requires no estimation of the future channel state, since the uplink data rate, which is held constant for each device i, is sufficient to compute the MAC-layer slot capacities, which will be discussed in Section IV. Second, in a rate-adaptive design, the uplink transmit power of each

[7] Even for applications such as residential smart utility meters, there exists a delay constraint, which may extend to 1 hour. All IoT applications, whose delay constraints may potentially span a wide range, are included in our framework.

[8] Throughout this paper, t will be used a discrete-time index for MAC-layer slots.

[9] Whereas forecasting is performed for each device without regard to what channel resources are available, scheduling is performed over multiple channels. Hence, the phrase "multi-channel" is in reference to only the resources available for scheduling; it does not refer to the forecasting schemes employed.

device i is held constant. In this case, as the channel condition changes, the data rate as well as the slot capacities used by MC-JFS will vary. Thus, for MC-JFS, large-scale Channel State Information (CSI) [58], [59] must be obtained for the uplink of each device i to G at the resolution of successive MAC-layer slots[10] over the duration of the upcoming scheduling window. For static IoT devices, the large-scale CSI can be obtained since the position of the IoT device does not change in this case. For mobile IoT devices, channel sounding data based on the positions of mobile IoT devices relative to the IoT Gateway G may be used as in [58], [59] to obtain the required large-scale CSI.[11]

C. FORECASTING SCHEME

In [21], IoT traffic for M2M communication was classified into device classes that are distinguished by two features: First, if the number of bits generated by a device at each of its traffic generation instances is equal, then the device is said to be "Fixed Bit," and a device is said to be "Variable Bit" otherwise. Second, a device that generates at regular time intervals is said to be "Periodic," and a device is said to be "Aperiodic" otherwise.[12] Accordingly, all IoT devices may be classified based on these two features into one of these 4 classes: Fixed Bit Periodic (FBP); Fixed Bit Aperiodic (FBA); Variable Bit Periodic (VBP); and Variable Bit Aperiodic (VBA). We adopt the same terminology for these device classes in this paper.

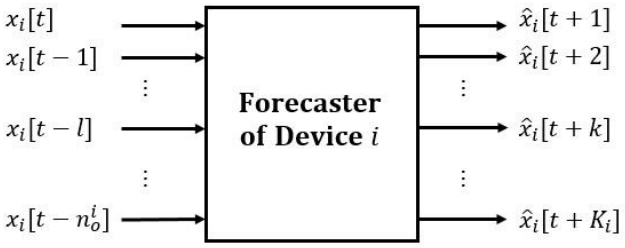


FIGURE 2. The forecaster for device i at Gateway G. Note that the Forecasting Module of Fig. 1 is comprised of a bank of such forecasters for individual IoT devices in the coverage area of G, as i ranges from 1 to N.

Within the forecasting module of the JFS system, we employ for each IoT device i a separate forecaster, as shown in Fig. 2. The forecaster takes as input the last n_0^i values as well as the current value of the traffic generation pattern of device i, namely $\{x_i[t-l]\}_{l\in\{0,\ldots,n_0^i\}}$. We define K_i as the maximum value of k for k-step ahead prediction of the traffic generation pattern of device i. In the figure, the forecaster outputs the k-step ahead prediction for the traffic generation pattern of device i, where k ranges from 1 to K_i. The predicted values are shown as $\{\hat{x}_i[t+k]\}_{k\in\{1,\ldots,K_i\}}$ in Fig. 2.

In this paper, for forecasting, we follow the same methodology as in [23]: First, we use an MLP architecture for forecasting, since such an architecture was demonstrated to provide an effective trade-off between network performance and execution time. Second, we exhaustively search over all of the MLP architectures parameterized by the number of past inputs n_0^i in the range $(n_0^i)_{\min}$ to $(n_0^i)_{\max}$. (The values of these two parameters shall be specified in Section V-A.) For each n_0^i in the range $(n_0^i)_{\min} \le n_0^i \le (n_0^i)_{\max}$, we search for a local optimum of the remaining architectural parameters of the MLP for device i, which are the number of hidden layers, denoted by E_i, and the number of neurons n_e^i in each hidden layer e such that $1 \le e \le E_i$. We select the local optimal values of E_i and n_e^i for e, where $e \in \{1, \ldots, E_i\}$, via the Neural Network Selection Algorithm (NNSA) that appears in [21], which we adapt to this work by minimizing the Mean Square Error (MSE) in the place of sMAPE.

IV. MATHEMATICAL FRAMEWORK FOR MULTI-CHANNEL JOINT FORECASTING-SCHEDULING (MC-JFS)

The main contribution of this work is the design of multi-channel scheduling techniques for the Scheduling Module in Fig. 1. While we shall use the MLP forecaster described in the previous section for the Forecasting Module, our main goal in this section will be to schedule the uplink traffic of IoT devices to the IoT Gateway on multiple channels over the scheduling window. To this end, first, we formulate the general form of the scheduling optimization program that will be run inside the Scheduling Module. Second, we present a reduced program for the special case of identical channels, i.e. the case where each of the uplink channels of any given device has an identical capacity. Third, we present our MC-LAPAL heuristic that is targeted at solving the general form of the scheduling optimization program in real time.

Our scheduling scheme works over non-overlapping, adjacent MAC-layer *slots*, each of which has duration τ_{MAC}, over a scheduling window of duration T_{sch}, which is chosen to be an integer multiple of τ_{MAC}. We assume that the transmission of a burst consumes at least one slot-channel pair.[13] We allow at most one burst to be scheduled in each slot-channel pair. Furthermore, each burst is transmitted "preemptively"; in other words, the transmission of the burst may be scheduled over non-adjacent slots.

An example of such a schedule is shown in Fig. 3. In this figure, the scheduling window, which is shown on the horizontal axis, lasts only 5 MAC-layer slots. Furthermore, there

[10] As opposed to PHY layer channel estimation, which operates over PHY-layer symbols, the channel state averaged over a MAC-layer slot is sufficient for MC-JFS.

[11] The PHY-layer issues in regard to this rate-adaptive scenario will be taken up in our future work.

[12] Note that "Periodic" in this context does not imply that the same *value* is repeated at these regular time intervals.

[13] Given the typical sizes of bursts from IoT devices, each burst is expected to fill at least one slot-channel pair.

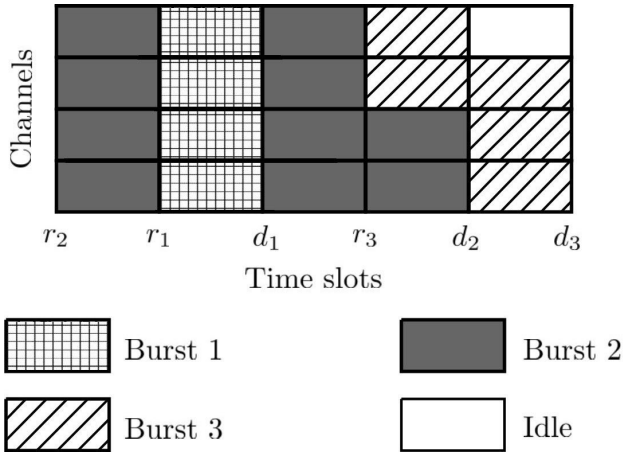


FIGURE 3. An example schedule for three bursts: In this simple example, the scheduling window lasts only 5 MAC-layer slots (shown in the horizontal direction), and there are 4 channels (shown in the vertical direction). Each slot-channel pair in this grid is assigned to at most one burst by the Scheduling Module. The generation time r_j and the deadline d_j of each burst $j \in \{1, 2, 3\}$ are also shown on the horizontal axis.

are only 4 channels, which extend over the vertical axis.[14] In this example, there are three bursts, whose bits are split across the slot-channel pairs. The generation time r_j and the deadline d_j of each burst $j \in \{1, 2, 3\}$ are shown on the horizontal axis in the figure. Preemption is exemplified by the schedule of Burst 2, whose bits appear over a set of non-adjacent time slots.

We assume that each frequency channel is of equal bandwidth (in Hertz); however, by virtue of the uplink modulation scheme used by the IoT device, the spectral efficiency and thus the data rate on each of these channels may vary. We let $\tilde{R}^{mi}$ denote the uplink data rate of device i on channel m. If burst j belongs to device i, then the data rate at which burst j can be transmitted on channel m is thus equal to $\tilde{R}^{mi}$. In order to effect the change of index from device i to any of its bursts j, we define f to be the function that maps each burst j to the device i that generated that burst; that is, $i = f(j)$. Then, we let R_{mj} denote the data rate at which burst j can be transmitted by the device i that generated the burst; that is, $R_{mj} = \tilde{R}^{m(f(j))}$.

Throughout this paper, $\mathcal{J}$ denotes the collection of bursts generated by the devices in $\mathcal{N}$ on the scheduling window. Furthermore, $\mathcal{M}$ denotes the collection of all channels that are available for scheduling. Finally, we note that the proof of each theorem that appears in this section is presented in the Appendix.

A. GENERAL FORM OF THE SCHEDULING OPTIMIZATION PROGRAM

In this section, first, we present the general form of the multi-channel scheduling optimization program. Second, we present a two-step algorithm which returns an optimal solution of this optimization program.

The general form of the scheduling optimization program is as follows:

$$\max \sum_{j \in \mathcal{J}} a_j u_j \tag{1}$$

subject to

$$\sum_{j \in \mathcal{J}} w_{mjt} \leq 1 \quad \forall t \in \mathcal{T}_+, \ \forall m \in \mathcal{M} \tag{2}$$

$$Q_{mjt} \leq C_{mj} w_{mjt} \quad \forall t \in \mathcal{T}_+, \ \forall j \in \mathcal{J}, \ \forall m \in \mathcal{M} \tag{3}$$

$$w_{mjt} \leq u_j \quad \forall t \in \mathcal{T}_+, \quad \forall j \in \mathcal{J}, \ \forall m \in \mathcal{M} \tag{4}$$

$$\sum_{t=r_j}^{d_j - 1} \sum_{m \in \mathcal{M}} Q_{mjt} = u_j a_j \quad \forall j \in \mathcal{J} \tag{5}$$

Above, a_j, r_j, d_j and C_{mj} are parameters, and w_{mjt}, u_j and Q_{mjt} are decision variables. We shall give the definitions of all of the parameters and the decision variables that appear in (1)-(5) before we state the physical meaning of the objective function and the constraints. The parameter a_j is the total number of bits in burst j. Furthermore, r_j is the *slot* at the beginning of which burst j is generated, and d_j is the slot by the beginning of which G must receive the delivery of all of the bits of burst j. The parameter C_{mj}, called the "slot capacity" or simply "capacity"[15] of channel m for burst j, is defined to be the maximum number of bits of burst j that can be transmitted on channel m in a single MAC-layer slot.[16] Thus, $C_{mj} = R_{mj}\tau_{MAC}$.

In the program above, u_j and w_{mjt} are binary decision variables, and Q_{mjt} is an integer decision variable. We note that t takes on only integer values and indexes the time slots. The variable w_{mjt} is defined only on the interval $r_j \leq t \leq d_{j-1}$, and $w_{mjt} = 1$ if any of the bits of burst j are scheduled to be transmitted on channel m in slot t and $w_{mjt} = 0$ otherwise. Above, $\mathcal{T}_+$ is the set of t's for which at least one w_{mjt} has been defined. Furthermore, $u_j = 1$ if all of the bits of burst j are transmitted by the end of slot $d_j - 1$ and $u_j = 0$ otherwise. The variable Q_{mjt} denotes the number of bits of burst j scheduled on channel m in slot t.

We now explain the physical meaning of the objective function and the constraints that appear in (1)-(5). The burst j is said to have been "completed" if all of the bits in burst j have been delivered to G by the beginning of slot d_j. Hence, the objective in (1) is to maximize the total number of bits in *completed* bursts. We focus on delivering all of the bits in any given burst because, as stated in Section III-A, a burst is defined as a set of bits such that even if a single bit of the burst is not delivered by the delay deadline, the delivery of the remaining bits of the burst is useless from the perspective of the application layer.[17] The constraint in (2) ensures that

[14] In practice and in our simulations, the duration of the scheduling window will be much larger than τ_{MAC}. Furthermore, the number of channels will typically be larger than the one in this example. This example serves merely as an illustration of the basic concepts.

[15] Throughout this paper, the term "capacity," whenever it is used by itself, refers only to "slot capacity"; it does *not* refer to Shannon capacity.

[16] Recall that we use collision-free scheduling throughout this paper.

[17] Examples of bursts that support this definition for IoT applications are given in Section III-A.

in each channel, in any given time slot, the bits of at most one burst is scheduled.[18] The constraint in (3) states that (A) if any of the bits of burst j is scheduled on channel m in slot t (that is, if $w_{mjt} = 1$), then the total number of bits of burst j on channel m in slot t, namely Q_{mjt}, is less than or equal to the maximum number of bits of burst j that can be transmitted on channel m in a single MAC-layer slot, and (B) if no bits of burst j is scheduled on channel m in slot t (that is if $w_{mjt} = 0$), then the total number of bits of burst j on channel m in slot t, namely Q_{mjt}, is zero. Thus, this constraint provides the coupling that must exist between Q_{mjt} and w_{mjt}. The constraint in (4) states that if not all of the bits of burst j are transmitted by the deadline d_j, then no slot-channel pair is allocated to burst j. Thus, this constraint provides the coupling that must exist between the w_{mjt}'s and u_j. The constraint in (5) states that the total number of bits allocated to burst j over all slot-channel pairs equals the total number of bits in burst j if all of the bits of j are transmitted by d_j (that is if $u_j = 1$), and no bits of burst j are scheduled on any of the slot-channel pairs otherwise (that is, if $u_j = 0$). Thus, this constraint provides the coupling that must exist between the Q_{mjt}'s and u_j.

We now present a two-step algorithm that returns an optimal solution of the general form of the scheduling optimization program above. The main idea behind this algorithm is to write a single constraint in the place of (3)-(5) by relaxing the variables w_{mjt} (that appear in the general form of the program) in the first step of the algorithm below and then tightening the relaxed variables. We shall prove that this two-step algorithm returns an optimal solution of the general form of the scheduling optimization program in (1)-(5). Subsequently, we will use the two-step algorithm to establish a reduction of the general form of the scheduling optimization program to that of scheduling on a single channel for the special case of identical channels.

Two-Step Algorithm:

Step 1: Solve the following program:

$$\max \sum_{j\in\mathcal{J}} a_j u_j \tag{6}$$

subject to

$$\sum_{j\in\mathcal{J}} \tilde{w}_{mjt} \leq 1 \quad \forall t \in \mathcal{T}_+, \quad \forall m \in \mathcal{M} \tag{7}$$

$$\sum_{t=r_j}^{d_j-1} \sum_{m\in\mathcal{M}} C_{mj}\tilde{w}_{mjt} \geq u_j a_j \quad \forall j \in \mathcal{J} \tag{8}$$

Above, a_j, r_j, d_j and C_{mj} are parameters, and $\tilde{w}_{mjt}$ and u_j are binary decision variables. The definitions of the parameters are the same as those in the general form of the program (1)-(5). The variable $\tilde{w}_{mjt}$ is defined only on the interval $r_j \leq t \leq d_{j-1}$, and $\tilde{w}_{mjt} = 1$ if any of the bits of burst j are allowed to be scheduled on channel m in slot t, and $\tilde{w}_{mjt} = 0$ otherwise. Above, $\mathcal{T}_+$ is the set of t's for which at least one $\tilde{w}_{mjt}$ has been defined. Furthermore, $u_j = 1$ if all of the bits of burst j are transmitted by the end of slot $d_j - 1$ and $u_j = 0$ otherwise. As in (1)-(5), we say that burst j has been "completed" if all of the bits in burst j have been transmitted to G by the beginning of slot d_j.

Thus, the objective in (6) is to maximize the total number of bits in *completed* bursts. The constraint in (7) ensures that in each channel, in any given time slot, the bits of at most one burst is scheduled. The constraint in (8) states that the sum of the maximum number of bits of burst j that can be transmitted over the set of allowed slot-channel pairs is at least the total number of bits a_j of burst j if all of the bits of j are transmitted by the deadline d_j, and no constraint is placed on the $\tilde{w}_{mjt}$'s otherwise.

Step 2: For every $j \in \mathcal{J}$ with $u_j = 1$ at the end of Step 1: First, arrange the elements of the set $\tilde{W}_j \equiv \{\tilde{w}_{mjt} | \tilde{w}_{mjt} = 1\}$ in the order of non-increasing C_{mj} (breaking ties arbitrarily), and let $\langle \tilde{W}_j \rangle$ be the resulting sequence. Second, schedule the transmission of the total number of bits a_j of burst j on the slot-channel pairs (t, m) in the order that they appear in $\langle \tilde{W}_j \rangle$ such that we allocate all of the available bits in a slot-channel pair up to its capacity C_{mj} before beginning the allocation of the remaining bits on the next slot-channel pair in the sequence. This procedure terminates when each of the a_j bits has been scheduled on some slot-channel pair. We define w_{mjt} as a binary variable such that $w_{mjt} = 1$ if any of the bits of burst j has been scheduled on (t, m), and $w_{mjt} = 0$ otherwise. The collection of w_{mjt}'s with which the procedure terminates constitutes the "schedule" for transmitting all of the bursts that satisfy $u_j = 1$ in Step 1. Finally, for all j with $u_j = 0$ at the end of Step 1, we set $w_{mjt} = 0$ for all (t, m).

Intuitively, the $\tilde{w}_{mjt}$'s in Step 1 above are temporary variables that allow a disjoint allocation of bursts to slot-channel pairs such that all of the a_j bits in each burst j can be transmitted while maximizing the total number of bits in completed bursts. For those bursts that will be completed (i.e. $u_j = 1$), the w_{mjt}'s in Step 2 above constitute the schedules obtained by tightening the loose allocation obtained in Step 1 to those slot-channel pairs that have the largest C_{mj} capacities. The tightening is targeted at reducing the transmit energy consumption of the devices by minimizing the number of slot-channel pairs over which each burst is transmitted. For any burst j that will not be scheduled at all (i.e. $u_j = 0$) in Step 1, no slot-channel pair is allocated in Step 2.

The following theorem establishes the relationship between the general form of the scheduling optimization program and the Two-Step Algorithm.

Theorem 1 (Optimality via the Two-Step Algorithm): The Two-Step Algorithm returns an optimal solution of the general form of the scheduling optimization program in (1)-(5).

B. SPECIAL FORM OF THE SCHEDULING OPTIMIZATION PROGRAM FOR IDENTICAL CHANNELS

We shall now address a special case of the scheduling optimization program in which the data rate is identical on each channel for a given device. In this case, $R_{mj} = R_j$ and

[18] This constraint is reasonable since bursts are typically occur at sparse interval for massive IoT applications. In our system design, each MAC-layer slot is devoted at most to the bits of a single burst.

$C_{mj} = C_j$ for each burst j; that is, we drop the channel index m and define R_j and C_j to be the common data rate and the common MAC-layer slot capacity, respectively, for each burst j across all of the channels.[19]

The assumption of identical channels typically arises in the scenario where the PHY layer does not communicate Channel State Information (CSI) to the MAC layer. In this case, the design at the MAC layer assumes that the data rate on each channel is identical.[20]

In the case of identical channels, the scheduling program can be formulated more simply (in lieu of (1)-(5) in Section IV-A) as follows:

$$\max \sum_{j \in \mathcal{J}} a_j u_j \tag{9}$$

subject to

$$\sum_{j \in \mathcal{J}} w_{mjt} \leq 1 \quad \forall t \in \mathcal{T}_+, \ \forall m \in \mathcal{M} \tag{10}$$

$$\sum_{t=r_j}^{d_j-1} \sum_{m \in \mathcal{M}} w_{mjt} = u_j \lceil a_j / C_j \rceil \quad \forall j \in \mathcal{J} \tag{11}$$

Now, let $M \equiv |\mathcal{M}|$. Then, we state the following theorem.

Theorem 2 (Reduction for Identical Channels): For any given IoT device, if the data rate on each of its uplink channels to the IoT Gateway is identical, then the following Reduced Two-Step Algorithm returns an optimal solution to the scheduling optimization program in (9)-(11) for identical channels:

Reduced Two-Step Algorithm:

Step 1: Solve the following "reduced program":

$$\max \sum_{j \in \mathcal{J}} a_j u_j \tag{12}$$

subject to

$$\sum_{j \in \mathcal{J}} y_{jt} \leq M \quad \forall t \in \mathcal{T}_+ \tag{13}$$

$$\sum_{t=r_j}^{d_j-1} y_{jt} = u_j \lceil a_j / C_j \rceil \quad \forall j \in \mathcal{J} \tag{14}$$

Above, y_{jt} is the number of slot-channel pairs allocated to burst j over all channels in slot t. The constraint in (13) states that for each slot, the total number of slot-channel pairs allocated to all of the bursts cannot exceed the number of channels available.[21] The constraint in (14) states that the total number of slot-channel pairs allocated to burst j equals the number of such slot-channel pairs required to send a_j bits if any of the bits of burst j are scheduled to be transmitted, and equals 0 otherwise.

Step 2: Set w_{mjt} in the original program as follows: Fix any ordering of the bursts in $\mathcal{J}$ and any ordering of the channels in $\mathcal{M}$. Then, for each slot t, starting with burst 1 and channel 1, assign channel indices that begin at channel index $\sum_{k=0}^{k=j-1} y_{kt} + 1$ and end at channel index $\sum_{k=0}^{k=j-1} y_{kt} + y_{jt}$ to burst j (where we take $y_{0t} = 0$).

Since each of the Two-Step Algorithm and the Reduced Two-Step Algorithm utilizes an optimization program in its statement, the time complexity of each of these algorithms depends on the particular method that is used to solve the optimization program. In Section V-C, we shall demonstrate the empirical time complexity of the Reduced Two-Step Algorithm for our simulations in the case of identical channels.

C. DESIGN OF THE MC-LAPAL HEURISTIC

While the Two-Step Algorithm in (6)-(8) results in a reduction in the empirical time complexity required to compute an optimal solution to the general form of the scheduling optimization program in (1)-(5), we validated in our simulations that the resulting reduction is not sufficient to produce a solution of the general form of the program in real time. Since real-time scheduling at IoT Gateways is key for the practical deployment of JFS, we designed a fast scheduling heuristic, named Multi-Channel Look Ahead Priority based on Average Load (MC-LAPAL), that solves (1)-(5) in real time.

Below, we first give an intuitive description of MC-LAPAL. Second, we state MC-LAPAL formally via pseudo-code. Third, we compute the time complexity as well as the space complexity of MC-LAPAL. The analytical results that we present on the time complexity of MC-LAPAL tie in with the empirical results on its computation time that appear in Section V-C.

1) INTUITIVE DESCRIPTION OF MC-LAPAL

In line with the basic framework for all of the scheduling techniques in this paper, MC-LAPAL aims to schedule the uplink traffic of all IoT devices in the coverage area of Gateway G over a scheduling window of duration T_{sch} based on the point estimates of the traffic of all of the IoT devices in this scheduling window. Hence, the algorithm is run once per scheduling window.

We first define the variables that we will use in describing the operation of MC-LAPAL: The number of bits of burst j that have not yet been scheduled by the beginning of the current slot t shall be denoted by $\tilde{a}_j[t]$. (From this point on, in our description of MC-LAPAL, we shall refer to "the beginning of the current slot t" as "the current slot t.")

[19] Recall that each burst j is generated by device $i = f(j)$. Hence, when the data rate of each of the uplink channels of a device is identical, this implies that the data rate for each *burst* of that device is also identical across all of the uplink channels of that device.

[20] Note that the MAC layer allocates slots to devices at the resolution of τ_{MAC}, which is much larger than the symbol interval at the PHY layer. Hence, in this case, the MAC layer aims at achieving only a coarse allocation of resources on the slot-channel grid. From the perspective of cross-layer design that allocates resources optimally by optimizing the MAC layer and the PHY layer jointly, the resulting allocation will be suboptimal except in the following scenario: In the case where the coherence bandwidth for the uplink transmission of an IoT device to Gateway G is much larger than the total bandwidth available to that device for its communication, the value of the transfer function on each of the OFDM channels allocated to that device is approximately the same. Hence, in this special case of a flat channel across the entire communication bandwidth, the assumption of identical data rates across all of the uplink OFDM channels of a device will produce an optimal resource allocation.

[21] The assumption that no more than one burst is allowed to be allocated to each slot-channel pair is retained.

We shall let $\tilde{\Delta}_j[t]$ denote the duration that remains, measured at the current slot t, until the beginning of slot d_j, namely until the delay deadline of burst j. Furthermore, $\mathcal{J}_{active}[t]$ denotes[22] the set of "active bursts," which is defined to be the set of bursts, each of which satisfies the following two properties: (1) even though the burst has been generated, it has not been processed in its entirety; (2) the delay deadline of the burst has not expired by the current slot t. We define the "effective load" for each $j \in \mathcal{J}_{active}[t]$, measured at the current slot t, as $\gamma_j[t] \equiv \tilde{a}_j[t]/\tilde{\Delta}_j[t]$. Furthermore, we let $\mathcal{J}_{upcoming}[t]$ denote the set of bursts whose generation times are later than the current time slot t.[23] Then, we define $r_{min}[t] \equiv \min_{j\in\mathcal{J}_{upcoming}[t]} r_j$. Finally, for each burst j, let $\langle\mathcal{M}_j\rangle$ denote the sequence of channels that have been sorted in the descending order of R_{mj}.

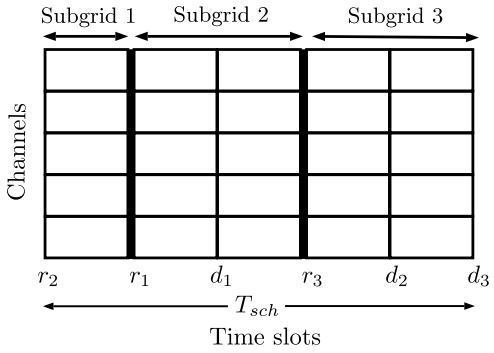


FIGURE 4. An example of the decomposition of the scheduling window into subgrids in the operation of MC-LAPAL.

We shall describe MC-LAPAL intuitively before we state it formally: The main idea behind this algorithm is that the algorithm decomposes the slot-channel grid that extends over the entire scheduling window into subgrids, each of which extends between the generation times of two successive bursts within the scheduling window, as shown in the example in Fig. 4. In this figure, there are three subgrids, namely those that span the intervals $[r_2, r_1]$, $[r_1, r_3]$, and $[r_3, d_3]$. (The last interval extends from the latest generation time r_3 to the end of the scheduling window.) The algorithm creates schedules for successive subgrids; that is, in forming its schedule, the algorithm "looks ahead" only to the next burst generation time rather than to the end of the entire scheduling window. Now, within each subgrid, the algorithm orders the bursts in the set $\mathcal{J}_{active}[t_k]$ in the order of non-increasing $\gamma_j[t_k]$, where t_k is the time at the beginning of the subgrid k. (The ordering breaks ties arbitrarily wherever necessary.) Starting at the beginning of the ordered list of bursts, the algorithm examines each burst j and decides whether it can be "processed," that is, whether it can be scheduled by using the available resources up to (and excluding) slot d_j. If it can be processed, the algorithm schedules the bits of burst j on the available slot-channel pairs in that subgrid, selecting the channels in the order in which they appear in $\langle\mathcal{M}_j\rangle$. (The only exception is that the last slot-channel pair that will be used for scheduling burst j is selected as one whose capacity is greater than or equal to and closest to the remaining number of bits of burst j that are to be scheduled.) After each of the slot-channel pairs in subgrid k has been allocated to some burst, if there is a burst such that only a part of its bits could be allocated within subgrid k, the remaining bits of this burst shall be carried over to the next subgrid $k+1$. (Note that there can be at most one such "left-over" burst.) After the algorithm finishes its allocations for subgrid k, it moves onto the next subgrid $k+1$ and repeats this procedure until it reaches the end of the scheduling window.

[22]This definition of the set of active bursts is identical to the one that appeared in [22] in the description of the Priority based on Average Load (PAL) heuristic, which was targeted at the *single-channel* case. In contrast, we develop a heuristic for the *multi-channel* case in this paper. The full set of differences between the single-channel and the multi-channel cases appear in Section II.

[23]Recall that JFS forms a schedule based on the *forecast* bits of IoT devices. Hence, the "generation time" of a burst is the generation time of a *forecast* burst within the current scheduling window. Similarly, all references to bursts within the scheduling window are those that refer to *forecast* bursts.

2) FORMAL DESCRIPTION OF MC-LAPAL

The pseudo-code of MC-LAPAL is shown in Fig. 5. In this pseudo-code, we assume that the following parameters are globally available: the total number of channels M, the duration of the scheduling window T_{sch}, the duration of a MAC-layer slot τ_{MAC}, the capacity matrix $\mathbf{C}$ (whose entry (m, j) is equal to C_{mj}), the collection of burst generation times $\{r_j\}_{j\in\mathcal{J}}$, and the entire set of bursts $\mathcal{J}$. Each element of $\mathcal{J}$ is a burst, which is a structure that contains the following fields: (1) burstID, which is the unique identification (ID) of a burst, (2) deviceID, which is the unique ID of the device that generated the burst, (3) residualNumberOfBits (which has been initialized to a_j), (4) generationTime, which is the global index of the slot in which the burst is generated, and (5) deadline, which is the global index of the slot by the beginning of which the burst must be delivered.

We now state only the essential definitions and explanations that complement the pseudo-code for MC-LAPAL in Fig. 5. On Line 4, the SetupSequenceOfSubgrids function takes $\{r_j\}_{j\in\mathcal{J}}$ as inputs and calculates the sequence of subgrids $\langle\mathcal{D}\rangle$. Each element of this sequence, namely a subgrid, is comprised of the following fields: (1) beginningOfSubgrid, which denotes the global slot index for the beginning of the subgrid, and (2) endOfSubgrid, which denotes the global slot index for the end of the subgrid. On Line 5, the schedule matrix $\mathbf{S}$ is initialized to the zero schedule matrix, whose number of rows equals M and whose number of columns equals the number of MAC-layer slots within the scheduling window, computed as T_{sch}/τ_{MAC}. The entry (m, t) of $\mathbf{S}$ equals i if slot t on channel m has been allocated to device $i \in \{1, \ldots, N\}$ and equals 0 if the slot-channel pair (m, t) has *not* been allocated to any device. In each iteration k of the outer loop (Line 6) of MC-LAPAL, the algorithm calls the ComputeSchedule function (Line 7), which computes a schedule given the leftOver burst as well as $\langle\mathcal{J}_{active}\rangle$ for the *current* subgrid $\langle\mathcal{D}\rangle[k]$ and returns the resulting leftOver burst and $\langle\mathcal{J}_{active}\rangle$ to be passed to the next subgrid $\langle\mathcal{D}\rangle[k+1]$ for scheduling.

MC-LAPAL:

```
1  schedule MC-LAPAL {
2    burstSeq ⟨J_active⟩ = ⟨φ⟩;
3    burst leftOver = φ;
4    ⟨D⟩ = SetupSequenceOfSubgrids({r_j}_{j∈J});
5    schedule S = zeros(M, T_sch/τ_MAC);
6    for(k = 0; k < length(⟨D⟩); k++) {
7      [leftOver, ⟨J_active⟩] =
           ComputeSchedule(&S, &(⟨D⟩[k]),
                           leftOver, ⟨J_active⟩, C);
8    }
9    return S;
10 }
11 [burst, burstSeq] ComputeSchedule(
                 schedule* s, subgrid* d, burst leftOver,
                 burstSeq ⟨J_active⟩, matrix C̃) {
12   t_init = d → beginningOfSubgrid;
13   ⟨J̃⟩ = FindBurstsForSubgrid(J, t_init);
14   ⟨J'⟩ = Concatenate(⟨J_active⟩, ⟨J̃⟩);
15   ⟨γ[t_init]⟩ = CalculateAverageLoad(⟨J'⟩, t_init);
16   ⟨J_sorted⟩ = Sort(⟨J'⟩, ⟨γ[t_init]⟩);
17   ⟨J_sorted⟩ = Concatenate(leftOver, ⟨J_sorted⟩);
18   nBursts = length(⟨J_sorted⟩);
19   for(j = 0; j < nBursts; j++) {
20     [status, noSpaceLeft] =
           ScheduleBurst(s, &(⟨J_sorted⟩[0]), d, C̃);
21     if(status ∈ {completed, skipped})
22       RemoveBurst(⟨J_sorted⟩, 0);
23     else {      //status == thereIsLeftOver
24       leftOver = ⟨J_sorted⟩[0];
25       RemoveBurst(⟨J_sorted⟩, 0);
26       return [leftOver, ⟨J_sorted⟩];
27     }
28     if(noSpaceLeft) break;
29   }
30   return [φ, ⟨J_sorted⟩];
31 }
```

FIGURE 5. Pseudo-code of the MC-LAPAL heuristic.

Similarly, we describe the ComputeSchedule function, which begins on Line 11 in Fig. 5. On Line 13, the FindBurstsForSubgrid function finds the sequence of bursts $\langle\tilde{\mathcal{J}}\rangle$, each of whose generation time falls at the beginning of the current subgrid. (Recall that $\mathcal{J}$, which appears as an argument of the FindBurstsForSubgrid function, denotes the *entire* set of bursts over the current scheduling window.) We define $\tilde{a}_j$ for each burst j in $\mathcal{J}$ as the value of the residualNumberOfBits field of burst j. On Line 15, the CalculateAverageLoad function calculates, for each burst j in $\langle\mathcal{J}'\rangle$, the average load at the beginning of slot t_{init} as $\gamma_j[t_{\text{init}}] = \tilde{a}_j/(d_j - t_{\text{init}})$.

The *for* loop that begins on Line 19 attempts to schedule each burst in $\langle\mathcal{J}_{sorted}\rangle$ within the current subgrid. We shall see that if the scheduling of the 0th element of $\langle\mathcal{J}_{sorted}\rangle$ has been attempted (ending in success or failure), $\langle\mathcal{J}_{sorted}\rangle$ will be updated by removing this 0th element. On Line 20, the 0th element of $\langle\mathcal{J}_{sorted}\rangle$ is attempted to be scheduled by the ScheduleBurst function, which returns two flags: First, the "status" flag takes one of the following values: completed, skipped, and thereIsLeftOver. (The meaning of each of these values will be explained shortly.) Second, the "noSpaceLeft" flag equals 1 if the entire set of slot-channel pairs in the current subgrid have been filled by the ScheduleBurst function and equals 0 otherwise. From this point onwards, we shall refer to the burst that is being scheduled as the "current burst." If all of the bits of the current burst have so far been scheduled by the ScheduleBurst function on Line 20, the status flag is set to "completed." If the ScheduleBurst function has decided to skip over the current burst, then the status flag is set to "skipped." If only a part of the current burst has been scheduled and thus there are remaining bits of the current burst (after the scheduling of the entire subgrid has been completed), then the status flag is set to "thereIsLeftOver."

In Fig. 6, we display the pseudo-code of the ScheduleBurst function that is called by the ComputeSchedule function. We define t_{end} as the latest time slot up to which the bits of burst j can be scheduled within the current subgrid. On Line 11, the total capacity of the available slot-channel pairs until t_{end}, denoted by $c_{\text{totalWithinSubgrid}}$, is calculated. On Line 12, the total capacity of the available slot-channel pairs until d_j, denoted by $c_{\text{totalUpToDeadline}}$, is calculated. On Line 15, the elements of $(*s)$[24] that correspond to the available slot-channel pairs between t_{init} and t_{end} are set to i. On Lines 19-40, the ScheduleBurst function makes a reservation for device i on the available slot-channel pairs within the current subgrid in a manner that will be described below.

The goal of Line 20 is to find the subsequence $\langle\mathcal{M}_+\rangle$ of the entire set of channels $\mathcal{M}$ such that for each channel in $\langle\mathcal{M}_+\rangle$, there is at least one slot-channel pair that is available between t_{init} and t_{end}. To this end, first, the submatrix of the schedule matrix $(*s)$ that contains only the columns between t_{init} and t_{end} is formed. Second, the function *Prod* multiplies, for each row of this submatrix, all of the entries along that row, which results in an $M \times 1$ vector.[25] Third, the function *Find* returns the sequence of indices of all of the zero entries of the resulting vector. On Line 21, the SortChannels function sorts the channels in $\langle\mathcal{M}_+\rangle$ in descending order with respect to the sequence of capacities $\tilde{\mathbf{C}}[\langle\mathcal{M}_+\rangle, j]$ for burst j and returns the resulting *sorted* sequence of channels $\langle\mathcal{M}_+^{\text{sorted}}\rangle$. The goal of Lines 22-39 is to allocate the resources in each channel $\langle\mathcal{M}_+^{\text{sorted}}\rangle[m']$ (where $m' \in \{0, \ldots, \text{length}(\langle\mathcal{M}_+^{\text{sorted}}\rangle) - 1\}$) for burst j in the order in which the channels appear in $\langle\mathcal{M}_+^{\text{sorted}}\rangle$.

3) TIME AND SPACE COMPLEXITY OF MC-LAPAL

In this section, we state our results on the worst-case time complexity and space complexity of MC-LAPAL.

[24] $(*s)$ is the schedule matrix to which s points.

[25] The second argument of Prod, namely "2," indicates that the multiplication is performed along the second dimension of the submatrix.

ScheduleBurst:

[**bool**, **bool**] ScheduleBurst(schedule* s,
burst* J_{current}, subgrid* d, matrix $\tilde{\mathbf{C}}$) {
  $t_{\text{init}} = d \rightarrow$ beginningOfSubgrid;
  $g_{\text{end}} = d \rightarrow$ endOfSubgrid;
  $j = J_{\text{current}} \rightarrow$ burstID;
  $i = J_{\text{current}} \rightarrow$ deviceID;
  $\tilde{a}_j = J_{\text{current}} \rightarrow$ residualNumberOfBits;
  $d_j = J_{\text{current}} \rightarrow$ deadline;
  $t_{\text{end}} = \min(g_{\text{end}}, d_j) - 1$;
  $\mathbf{S}_{\text{emptyWithinSubgrid}} = ((*s)[:, t_{\text{init}} : t_{\text{end}}] == 0)$;
  $\mathbf{S}_{\text{emptyUpToDeadline}} = ((*s)[:, t_{\text{init}} : d_j - 1] == 0)$;
  $c_{\text{totalWithinSubgrid}}$ =
    Sum($\tilde{\mathbf{C}}[:, j]^T \mathbf{S}_{\text{emptyWithinSubgrid}}$);
  $c_{\text{totalUpToDeadline}}$ =
    Sum($\tilde{\mathbf{C}}[:, j]^T \mathbf{S}_{\text{emptyUpToDeadline}}$);
  if($\tilde{a}_j > c_{\text{totalUpToDeadline}}$) status = skipped;
  else if($(d_j > g_{\text{end}})\&\&(\tilde{a}_j > c_{\text{totalWithinSubgrid}})$) {
    $((*s)[(*s) == 0])[:, t_{\text{init}} : t_{\text{end}}] = i$;
    $\tilde{a}_j = \tilde{a}_j - c_{\text{totalWithinSubgrid}}$;
    $J_{\text{current}} \rightarrow$ residualNumberOfBits $= \tilde{a}_j$;
    status = thereIsLeftOver;
  } **else** {
    $\langle \mathcal{M}_+ \rangle$ = Find(Prod($(*s)[:, t_{\text{init}} : t_{\text{end}}], 2) == 0$);
    $\langle \mathcal{M}_+^{\text{sorted}} \rangle$ = SortChannels($\langle \mathcal{M}_+ \rangle, \tilde{\mathbf{C}}[\langle \mathcal{M}_+ \rangle, j]$);
    for($m' = 0; m' <$ length($\langle \mathcal{M}_+^{\text{sorted}} \rangle$); m'++) {
      $m = \langle \mathcal{M}_+^{\text{sorted}} \rangle [m']$;
      $\langle \mathcal{T}_+^m \rangle$ = Find($(*s)[m, t_{\text{init}} : t_{\text{end}}] == 0$);
      for($t' = 0; t' <$ length($\langle \mathcal{T}_+^m \rangle$); t'++) {
        if($\tilde{a}_j \leq \tilde{\mathbf{C}}[m, j]$) {
          $m^* = \underset{\tilde{m} \in \langle \mathcal{M}_+ \rangle}{\text{argmin}} \{\tilde{\mathbf{C}}[\tilde{m}, j] \mid \tilde{\mathbf{C}}[\tilde{m}, j] \geq \tilde{a}_j\}$;
          $\langle \mathcal{T}_+^{m^*} \rangle$ = Find($(*s)[m^*, t_{\text{init}} : t_{\text{end}}] == 0$);
          $(*s)[m^*, \langle \mathcal{T}_+^{m^*} \rangle [0]] = i$;
          $J_{\text{current}} \rightarrow$ residualNumberOfBits $= 0$;
          status = completed;
          break;
        } **else** {
          $(*s)[m, \langle \mathcal{T}_+^m \rangle [t']] = i$;
          $\tilde{a}_j = \tilde{a}_j - \tilde{\mathbf{C}}[m, j]$;
        }
      }
      if(status == completed) break;
      else $\langle \mathcal{M}_+ \rangle$ = Remove($\langle \mathcal{M}_+ \rangle, m$);
    }
  }
  if(Prod($(*s)[:, t_{\text{init}} : g_{\text{end}} - 1]$, "all") != 0)
    noSpaceLeft = 1;
  else noSpaceLeft = 0;
  return [status, noSpaceLeft];
}

FIGURE 6. Pseudo-code of the ScheduleBurst function.

Let $V \equiv T_{sch}/\tau_{MAC}$; that is, we let V denote the number of MAC-layer slots in a scheduling window. Recall that M is the total number of channels. Furthermore, let J denote the total number of bursts over the entire scheduling window; that is, $J \equiv |\mathcal{J}|$.

Theorem 3 (Time Complexity of MC-LAPAL): The worst-case time complexity of MC-LAPAL is

$$\mathcal{O}((J \log J) \min(J, V) + JM \log M + JMV).$$

Theorem 4 (Space Complexity of MC-LAPAL): The worst-case space complexity of MC-LAPAL is $\mathcal{O}(M(J+V))$.

The most important implication of the above two theorems is that both the worst-case time complexity and the space complexity of MC-LAPAL are polynomial in the system parameters V, M, and J. When we take into account the fact that the logarithmic terms are weakly growing functions, we see that the asymptotic growth with respect to each of these parameters is close to linear for both the time and the space complexity. This fact implies that MC-LAPAL can be implemented in practice at IoT Gateways.

In Section V-C, we shall compare the worst-case time complexity of MC-LAPAL obtained in Theorem 3 above against the empirical computation time of MC-LAPAL that we observe in our simulations in Section V.

D. REACTIVE PROTOCOL BENCHMARKS: MC-RAB AND MC-ERAB

In this section, we describe two protocols that we have developed, which will serve as reactive protocol benchmarks against which to compare the performance of MC-JFS.

MC-RAB is a protocol that reacts to the current traffic demand and allocates the traffic on multiple channels. MC-RAB uses a parameter p_{RAB}, which is similar to the access probability p_{ACB} in the Access Class Barring (ACB) protocol [1]. However, in contrast with ACB, which governs only the access of devices and not the scheduling of data traffic, MC-RAB is targeted at both giving access to devices *and* scheduling the device traffic on multiple channels via reservations.

In MC-RAB, in each slot t, the state of each device i is uniquely described by the set of the following indicator variables: (1) $z_{mi}^{\text{TX}}[t] = 1$ if and only if device i is transmitting on channel m in slot t, and $z_{mi}^{\text{TX}}[t] = 0$ otherwise, (2) $z_{mi}^{\text{backoff}}[t] = 1$ if and only if device i is in backoff on channel m in slot t, and $z_{mi}^{\text{backoff}}[t] = 0$ otherwise, and (3) $z_{mi}^{\text{waiting}}[t] = 1$ if and only if device i is waiting (for any of the other devices' reservations to terminate) on channel m in slot t, and $z_{mi}^{\text{waiting}}[t] = 0$ otherwise.

All of the generated bursts of device i are ordered with respect to their generation times. The device moves onto the processing of its next burst only after it has received the acknowledgment that all parts of the current burst j have successfully been received by Gateway G.[26]

All of the bits of burst j on which device i has not yet received an acknowledgment within a time-out duration[27]

[26] For massive IoT, the delay deadline of each burst j, namely Δ_j, which is determined by the IoT application, is typically less than or equal to the traffic generation period T_i of the device i that generates burst j. This implies that the delay deadline of the current burst that is processed will have expired by the time that the next burst of the same device is generated.

[27] The time-out duration may be set to the minimum time required for the acknowledgment from G to be received by device i.

from G are queued in the transmission queue. We define the "residual" of burst j as the entire set of those bits of burst j whose reception has not been acknowledged by G within the time-out duration. In our simulations, in order to give full advantage to MC-RAB over the proactive protocols that we have designed in Sections IV-A and IV-C, we assume that in MC-RAB, any acknowledgment from G is received instantaneously by each device to which the acknowledgment is transmitted.

We shall now describe how device i processes the residual $\tilde{j}$ of the current burst j. (Note that this includes the case in which burst j is being processed for the first time. In that case, the residual $\tilde{j}$ is the entire burst j.) At the beginning of each slot t, we shall let $\tilde{\mathcal{M}}_i^+[t]$ denote the set of available channels on which device i is not waiting (i.e. $z_{mi}^{\text{waiting}}[t]=0$), is not in backoff (i.e. $z_{mi}^{\text{backoff}}[t]=0$), and is not transmitting ($z_{mi}^{\text{TX}}[t]=0$) based on a reservation that has already been made for device i on channel m in slot t. At the beginning of slot t, if slot t has already been reserved for device i, then $z_{mi}^{\text{TX}}[t]=1$; otherwise, if device i has the residual $\tilde{j}$ in its transmission queue, it decides to access each channel $m \in \tilde{\mathcal{M}}_i^+[t]$ with probability p_{RAB} independently across all of the channels in $\tilde{\mathcal{M}}_i^+[t]$. Let $\mathcal{M}_{\tilde{j}}^{\text{sel}}[t]$ denote the subset of channels that have been selected by device i using this procedure for residual $\tilde{j}$ at the current time slot t. Let $\tilde{\Delta}_j[t]$ denote the time remaining at the current time t until the deadline d_j.[28] Then, device i divides the set of bits in residual $\tilde{j}$ as follows: First, the maximum number of bits of residual $\tilde{j}$ that device i can send on each channel m from the beginning of slot t to the end of slot d_j-1 is calculated as $a_{m\tilde{j}}^{\max}[t] \equiv C_{mj}\lceil \frac{\tilde{\Delta}_j[t]}{\tau_{MAC}} \rceil \quad \forall m \in \mathcal{M}_{\tilde{j}}^{\text{sel}}[t]$. Second, let $B_{\tilde{j}}$ denote the number of bits in residual $\tilde{j}$. Furthermore, let $\tilde{\mathcal{J}}_m$ denote the set of bits of residual $\tilde{j}$ that i decides to send on channel m, and let $A_{\tilde{\mathcal{J}}_m}$ denote the number of bits in $\tilde{\mathcal{J}}_m$. Then, i computes $A_{\tilde{\mathcal{J}}_m}$ by setting the number of bits of residual $\tilde{j}$ allocated to channel m as directly proportional to the slot capacity C_{mj} of channel m up to the maximum number of bits $a_{m\tilde{j}}^{\max}[t]$ that can be transmitted on channel m by the deadline of residual $\tilde{j}$. That is,

$$A_{\tilde{\mathcal{J}}_m} = \min\left\{ B_{\tilde{j}} \frac{C_{mj}}{\sum_{m' \in \mathcal{M}_{\tilde{j}}^{\text{sel}}[t]} C_{m'j}}, a_{m\tilde{j}}^{\max}[t] \right\} \tag{15}$$

Thus, for each $\tilde{\mathcal{J}}_m$, the processing time (measured in MAC-layer slots) is $p_{\tilde{\mathcal{J}}_m} = \lceil A_{\tilde{\mathcal{J}}_m}/C_{mj} \rceil$. The device i prepends the value of $p_{\tilde{\mathcal{J}}_m}$ to the set of bits $\tilde{\mathcal{J}}_m$ that are transmitted on channel $m \in \mathcal{M}_{\tilde{j}}^{\text{sel}}[t]$.[29] For each channel $m \in \mathcal{M}_{\tilde{j}}^{\text{sel}}[t]$, whenever G receives the value of $p_{\tilde{\mathcal{J}}_m}$, if $p_{\tilde{\mathcal{J}}_m} \geq 2$, G reserves the next $(p_{\tilde{\mathcal{J}}_m}-1)$ slots[30] of channel m for the set of bits $\tilde{\mathcal{J}}_m$ and acknowledges the reception of $\tilde{\mathcal{J}}_m$ by sending the set of slot-channel pairs that have been reserved for $\tilde{\mathcal{J}}_m$ to *all of the devices*[31] (including i). We shall denote this acknowledgment by $\text{ACK}_{\tilde{\mathcal{J}}_m}$. When each of the other devices besides i receives this $\text{ACK}_{\tilde{\mathcal{J}}_m}$, it transitions into the wait state for channel m for the next $(p_{\tilde{\mathcal{J}}_m}-1)$ slots.

If device i has transmitted $\tilde{\mathcal{J}}_m$ to G on channel m and does not receive the $\text{ACK}_{\tilde{\mathcal{J}}_m}$ (which happens only due to an uplink collision on channel m in our model), device i re-queues $\tilde{\mathcal{J}}_m$ for re-transmission and goes into the backoff state for channel m ($z_{mi}^{\text{backoff}}=1$) for a duration that is modeled as an exponential random variable with parameter λ.[32] If device i has received acknowledgments that encompass all of the bits of burst j, it moves onto its next burst (whenever the next burst is generated) and repeats the procedure above for this new burst.

MC-ERAB, which stands for "Multi-Channel Enhanced Reservation-based Access Barring," is a protocol that enhances MC-RAB by adding only the following extra condition, which aids in improving the throughput performance of the protocol: After device i completes the channel selection process that computes $\mathcal{M}_{\tilde{j}}^{\text{sel}}[t]$ for residual $\tilde{j}$, device i checks the condition $\sum_{m' \in \mathcal{M}_{\tilde{j}}^{\text{sel}}[t]} a_{m'\tilde{j}}^{\max}[t] \geq B_{\tilde{j}}$. This condition states that the total capacity (in bits) available across all of the set of selected channels is sufficient to send all of the $B_{\tilde{j}}$ bits of residual $\tilde{j}$. If the condition is satisfied, device i calculates $A_{\tilde{\mathcal{J}}_m}$ in accordance with (15) and continues the protocol as in MC-RAB; otherwise, in the next time slot $t+1$, if $t+1 < d_j$, device i repeats the entire process that begins with the selection of the set of channels $\mathcal{M}_{\tilde{j}}^{\text{sel}}[t+1]$ anew until either the condition above is satisfied or the delay deadline of residual $\tilde{j}$ has expired.[33]

V. RESULTS

In this section, our main goal is to compare the performance of MC-JFS against those of MC-RAB and MC-ERAB, which serve as reactive benchmark protocols. To this end, first,

[28] The deadline for the residual of burst j is d_j, namely the deadline for burst j.

[29] In MC-RAB, Gateway G acknowledges the reception of *each* such set $\tilde{\mathcal{J}}_m$ without waiting for successive such sets to accumulate in order to ensure that the deadline of this set of bits does not expire due to an unnecessary delay at G.

[30] Note that there is no separate access channel in MC-RAB. Thus, data reception at G has already begun in the first slot, which contains both the value of $p_{\tilde{\mathcal{J}}_m}$ and those bits of $\tilde{\mathcal{J}}_m$ that fall in the first slot. As a result, G makes a reservation for the *remaining* number of slots that will be required to receive the rest of $\tilde{\mathcal{J}}_m$.

[31] The downlink control channel, on which this information is broadcast to all of the devices in $\mathcal{N}$ simultaneously, typically does not constitute the bottleneck for IoT.

[32] We follow the common usage in the literature that models the backoff duration as an exponentially distributed random variable with parameter λ. However, note that since the MAC-layer slots impose a discrete structure, the corresponding random variable in our case has a geometric distribution whose parameter is calculated based on the λ of the underlying exponential random variable.

[33] Recall that for massive IoT, the delay constraint of burst j, which relates to the IoT application, is typically less than or equal to the traffic generation interval T_i of the device i that generates burst j. Hence, new bursts typically do not accumulate at device i as i attempts to send the current burst by repeating this procedure until d_j.

we describe our simulation methodology. Second, we discuss our results for the general case of the scheduling optimization program (Section IV-A) and for the special case of identical channels, in which single-channel reduction is possible (Section IV-B). Finally, we present our results on the computation time of our MC-LAPAL heuristic.

A. METHODOLOGY

In this subsection, first, we describe our methodology for traffic generation and forecasting. Second, we state how we determine the channel capacities based on our wireless channel model. Third, we specify our computation platform on which all of our simulation results have been obtained.

1) TRAFFIC GENERATION AND FORECASTING

First, for traffic generation and forecasting, we use the same data sets for IoT device traffic generation as in [22] and [23], which were formed based on the data collection and processing methodology (which includes bootstrapping traffic generation patterns from a set of IoT devices) detailed in Section VI-A of [23].

We use the same set of MLP forecasters for the above data set as in [23]. In particular, regarding the parameters discussed in Section III-C of the current work, we set $(n_0^i)_{\min}$ to 3 and $(n_0^i)_{\max}$ to 100. The set of delay constraints across the IoT devices are as follows: $\Delta^{(1)} = 180$, $\Delta^{(2)} = 180$, $\Delta^{(3)} = 1$, $\Delta^{(4)} = 0.5$, $\Delta^{(5)} = 2$, $\Delta^{(6)} = 180$, $\Delta^{(7)} = 600$ and $\Delta^{(8)} = 3600$ s. The set of traffic generation intervals for the same set are $T_1 = 180$, $T_2 = 180$, $T_3 = 180$, $T_4 = 180$, $T_5 = 3600$, $T_6 = 180$, $T_7 = 3600$ and $T_8 = 3600$ s. We set $T_{sch} = 100$ s in our simulations.[34]

2) WIRELESS CHANNEL MODEL

Second, we shall explain how we obtain the values of the parameters $\tilde{R}^{mi}$ and $\tilde{C}^{mi}$ for each $m \in \mathcal{M}$ and each $i \in \mathcal{N}$ based on a wireless channel model at the physical layer, for both the general case and for the special case of identical channels.

For the general case (Section IV-A), we model the channel in the time domain for OFDM systems according to the Rayleigh fading model [50]. In this model, the complex channel gain $h_i[l]$ of tap l for the uplink of device i has a real part $\bar{a}_i[l]$ and an imaginary part $\bar{b}_i[l]$, which are independent, circular symmetric Gaussian random variables. Each of the $\bar{a}_i[l]$ and $\bar{b}_i[l]$ has mean 0 and variance $\kappa/2L$, where κ is a positive parameter, and L is the number of channel taps. In our simulations, we set $\kappa = 0.1$.[35] For simplicity, in this work, we assume that the number of channel taps L is identical across all of the devices.[36] We took $L = 10$ in our simulations.

For the uplink of device i to Gateway G, the complex gains across all of the channels (in the frequency domain) shall be denoted by the sequence $\langle \tilde{h}_{mi} \rangle$ over $m \in \mathcal{M}$. This sequence of complex channel gains is obtained by taking the M-point discrete Fourier transform of the sequence formed by zero padding the sequence $\langle h_i[l] \rangle$ (which ranges over $1 \leq l \leq L$) by an additional $M - L$ entries [50].[37]

We let P denote the total transmit power of *each* of the devices in $\mathcal{N}$ over the entire set of M channels. Because we assume an OFDM system, the amount of transmit power allocated by a device to a single OFDM subcarrier on the uplink of a device is P/M.[38] By virtue of Shannon's capacity formula for an Additive White Gaussian Noise (AWGN) channel,[39] the data rate[40] on channel m for device i is $\tilde{R}^{mi} = \tilde{W}_m \log_2(1 + (|\tilde{h}_{mi}|^2 (P/M))/(N_o \tilde{W}_m))$, where $\tilde{W}_m$ is the bandwidth (in Hz) of channel m (which is assumed to be identical across all devices), and N_o is the value of the single-sided power spectral density of the white noise at the receiver of Gateway G, which is identical for every channel m for the uplink of every device i. We let $N_o = 5 \times 10^{-3}$ (in Watts/Hz) $\forall m \in \mathcal{M}$ and $\forall i \in \mathcal{N}$ in our simulations.

We let W denote the common total bandwidth used by all of the devices to communicate to the Gateway. In our simulations, we set the total communication bandwidth $W = 26$ MHz and the bandwidth of each subcarrier $\tilde{W}_m = 1$ MHz.[41] Hence, $M = 26$ in our simulations.[42] Finally, the MAC-layer capacity of each channel m for device i is given by $\tilde{C}^{mi} = \tilde{R}^{mi} \tau_{MAC}$, where $\tau_{MAC} = 0.1$ second.

For the special case of identical channels (Section IV-B), the channel capacity on the uplink of device i is chosen as

[34] Even though this results in the generation of at most one burst from each device over the scheduling window, this is a practical choice for T_{sch} in order to be able to compute schedules in real time. The capability to schedule over a much longer scheduling window requires a generalization of the multi-scale algorithm that appeared in [23] to the case of multiple channels, which is beyond the scope of the current work.

[35] For this value of $\kappa = 0.1$ and the remaining channel parameters, which shall be specified in this subsection, the communication system is in the power-limited regime. This choice of operating the system in the power-limited regime is especially suitable for battery-limited IoT devices.

[36] Since we focus on the MAC-layer scheduling of IoT devices in this work rather than the design of physical layer schemes, taking the number of channel taps to be the same across all devices provides a coarse approximation. The multipath profiles of the IoT devices to the IoT Gateway will have a similar number of taps if the physical environment that exists from each device to the Gateway is similar to that of the other devices.

[37] Note that $M \geq L$.

[38] Since our work is aimed at the MAC layer, we do not use Discrete Multitone Modulation (DMT), which would distribute the total power via waterfilling based on the uplink channel gain, if there were CSI fed back to the device on the downlink.

[39] The channel seen by each OFDM subcarrier is approximately a flat channel; hence, Shannon's capacity formula for the AWGN channel can be applied to each such subcarrier.

[40] We assume that well-known capacity-achieving channel codes, such as turbo or Low Density Parity Check (LDPC) codes, are used at the physical layer on each such OFDM subcarrier (i.e., on each channel $m \in \mathcal{M}$ in our terminology). As a result, we take the data rate to be the Shannon capacity achieved in this way on each such OFDM subcarrier.

[41] While this paper presents fundamental work on MC-JFS, we based our choice of the total available bandwidth as well as the bandwidth per subcarrier on one of the possible choices for the physical layer configuration of the IEEE 802.11ah standard.

[42] The reason for keeping M fixed in our simulations is that for a constant total bandwidth W and an equal power allocation of P/M by a device for each subcarrier, the total capacity does not vary significantly as M changes. In contrast, the total capacity varies as P changes.

the numerical average of the capacities $\tilde{C}^{mi}$ over $m \in \mathcal{M}$ that were obtained for the general case above. This choice allows us to compare the network performance for the special case of identical channels against that for the general case.

In our simulation results that appear in the next section, we shall examine the network perform performance for values of P in the set $\{5, 10, 15, 20\}$ mW. While this paper constitutes fundamental work and is not based on any particular standard, the reason for our selection of these particular values of P is that 5 and 10 mW correspond to the transmit power levels for the IEEE 802.11ah standard in China and Europe [60], while the values 15 and 20 mW correspond to range extensions on these systems via an increase in the transmit power. As a result, while this is not the main goal of this paper, our simulation results also give indications on the network performance that would be attained in systems similar to those that implement the physical layer of the IEEE 802.11ah standard.[43]

3) COMPUTATION PLATFORM

Third, we note that all of the experiments were performed on a 2.30 GHz Intel Xeon Gold 5118 24-Core Processor with 128 GB of RAM running Python 3.7 under the Windows 10 operating system with no other concurrent compute-intensive processes. In addition, the *docplex* library on Python was used to produce an optimal solution of the program in (12)-(14), which appears as part of the Reduced Two-Step Algorithm. (We note that the branch and cut method for integer linear programs in this library was used in solving (12)-(14).)

B. NETWORK PERFORMANCE

Our aim in this section is to present the results on the network performance of MC-JFS. First, for the general case, we demonstrate the performance of our proactive heuristic MC-LAPAL against those of the reactive protocols, namely MC-RAB and MC-ERAB. Second, for the special case in which the capacities of those channels on the uplink of any individual device are identical, we compare the performance of these protocols against that of optimal scheduling.

1) PERFORMANCE COMPARISON FOR THE GENERAL CASE

In this subsection, we present our results for the general case (Section IV-A), in which the uplink capacity of a device across distinct channels need not be identical and are generated according to the wireless channel model described in Section V-A.

We shall build our results in a bottom-up fashion, beginning with low-level metrics, such as the average fraction of the capacity of a single slot-channel pair that is allocated and the fraction of utilization of the slot-channel grid, and subsequently use the insights gained in order to explain the behavior of high-level network performance metrics, such as uplink cross-layer throughput η and the average transmit energy consumption per bit.

Recall that P denotes the total transmit power of each device (which is identical across all devices) over the entire set $\mathcal{M}$ of uplink channels. In the results that follow, we shall graph each metric in four subplots, each of which corresponds to a distinct value of P in the set $\{5, 10, 15, 20\}$ mW. In each of the subplots, we keep the value of P fixed and vary the number of devices N.[44] In all of our simulations, the percentage of devices in each device class remains at 25%.[45]

For each of the reactive protocols, namely MC-RAB and MC-ERAB, for each N, we maximize the throughput η by selecting the parameters $(\lambda_{\text{RAB}}, p_{\text{RAB}})$ for MC-RAB and $(\lambda_{\text{ERAB}}, p_{\text{ERAB}})$ for MC-ERAB via exhaustive search. Hence, all of the plots for the reactive protocols in this section correspond to their η-optimal performance for each N.

We begin our discussion with the presentation of low-level performance metrics. For the MC-LAPAL scheduling scheme, we let g denote the average fraction of the capacity of a single slot-channel pair, assuming that this pair is allocated for the transmission of bits. Fig. 7 displays g under perfect forecasts as well as under MLP forecasting. In this figure, first, we see that g is greater than 0.988 across all N and all P. Since each burst fills slot-channel pairs successively in the MC-LAPAL scheduling scheme, the only potential inefficiency that would result in the underallocation of the capacity of a slot-channel pair could occur in the last slot-channel pair that is used by a given burst.[46] Now, Line 27 of the ScheduleBurst function (Fig. 6) ensures that this last slot-channel pair that is allocated has a capacity as close as possible (among the remaining set of slot-channel pairs that could be allocated) to the remaining number of bits in the current burst that is being scheduled in this slot-channel pair. The fact that a high g is achieved in all of these subplots is due to the relatively high spread of channel capacities that exists in this general case.

Second, we see in Fig. 7 that as P increases, in general, g slightly decreases across all N. The reason is that as P increases, on average, the capacity of the last slot-channel pair that is filled by a burst increases. Since the set of traffic bursts across all P are identical in our simulations, this results in a slight decrease, on average, in g. Third, we see that this decrease, as a function of P, is faster for MC-LAPAL under MLP forecasting compared with that under perfect forecasts. Our examination of the forecast traffic generation patterns under MLP forecasting has revealed that small bursts exist in these forecast traffic generation patterns at instances where

[43] It is important to note that this paper does *not* implement the MAC layer of IEEE 802.11ah, as our aim is to employ MC-JFS at the MAC layer rather than that of the IEEE 802.11ah standard.

[44] Since we focus on a particular scheduling window, the number of devices whose bursts fall in this window depends on the particular realization of bursts. Hence, the values of N for which each of the plots displays measurements have irregular spacing.

[45] We shall demonstrate the network performance under varying percentages of devices from each device class in our future work.

[46] Recall that no more than one burst is allowed to be scheduled on any given slot-channel pair.

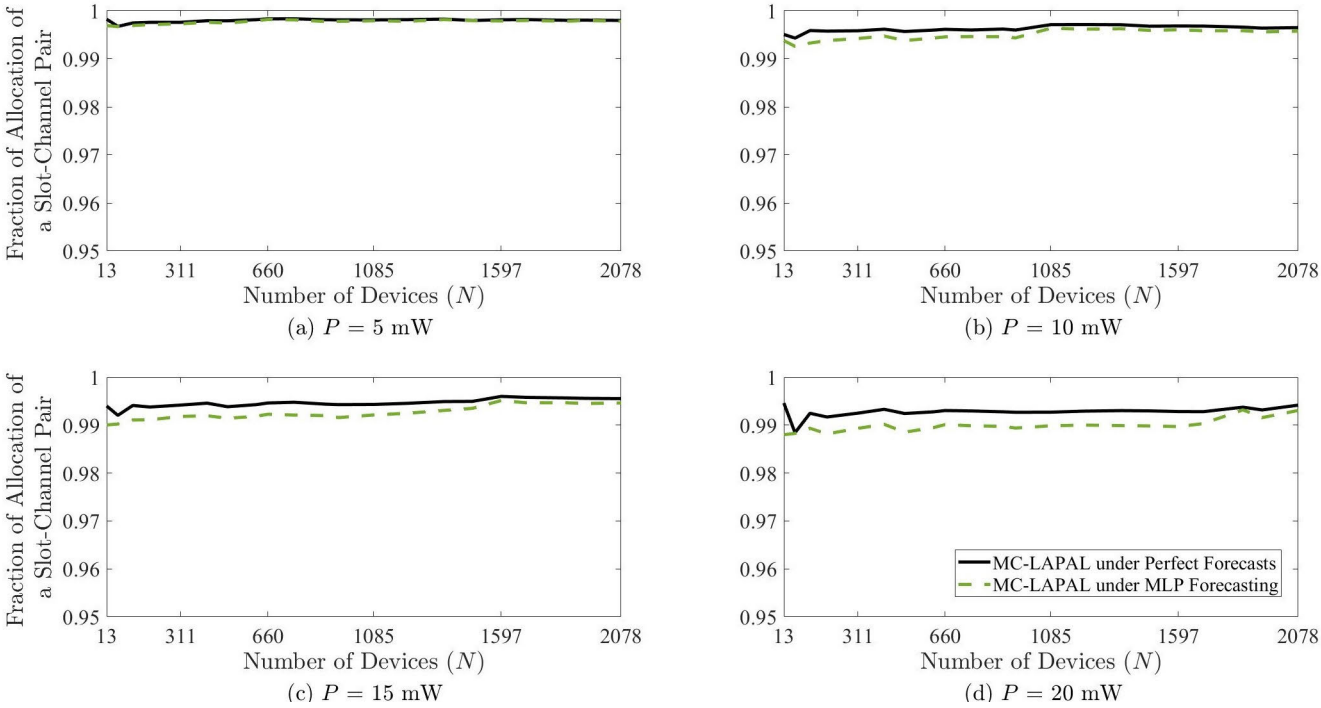


FIGURE 7. Average fraction of allocation of the capacity of a slot-channel pair.

no such burst occurs under perfect forecasts (i.e. in reality). These instances of burst overestimation result in the allocation of slot-channel pairs such that each such forecast burst fills only a small fraction of a slot-channel pair, thus reducing g. However, the overall conclusion of Fig. 7 is that MC-LAPAL, on average, almost fills each slot-channel pair that it allocates.

We define the "slot-channel pair utilization" of a scheduling scheme as the fraction of slot-channel pairs, on each of which an actual transmission of bits[47] occurs over the entire slot-channel grid.[48] Fig. 8 displays the fraction of utilization of the entire slot-channel grid.[49] We shall analyze the utilization of MC-LAPAL completely before we move onto those of the reactive protocols. In all of the four subplots of this figure, we see that the utilization of MC-LAPAL under both perfect forecasts and MLP forecasting increases approximately linearly until a point at which saturation occurs (i.e. the graph becomes approximately constant). We shall call the N at which this saturation occurs as the "point of saturation" and the approximate value of the utilization after the point of saturation the "saturation level." Across all four subplots, we see that the saturation level is lower for MC-LAPAL under MLP forecasting than under perfects forecasts. Furthermore, we see that the point of saturation shifts to the right as P increases.

We shall now focus on a particular subplot, Fig. 8(a), explain the trends there, and subsequently generalize the arguments to the remaining subplots. Recall that as the number of devices N increases, the percentage of devices from each of the four device classes remains at 25%. Forming the traffic generation patterns via bootstrapping, to which we referred in Section V-A, produces an approximately linear growth of the total traffic load as a function of N. In Fig. 8(a), this linear growth of the traffic load results in the approximately linear increase of the slot-channel pair utilization of MC-LAPAL under perfect forecasts until almost full utilization has been achieved at $N = 660$ devices.[50] In contrast, we see that MC-LAPAL under MLP forecasting does not achieve full utilization for any N. The reason is that MLP overestimates the amount of traffic that is generated over the scheduling window. As a result of this overestimation, MC-LAPAL overallocates slot-channel pairs in the slot-channel grid; however, not all of these slot-channel pairs are utilized by the actual traffic. Due to the fact that almost

[47] Whenever an actual transmission of bits of a burst occurs on a given slot-channel pair, we consider that slot-channel pair to have been utilized *without* regard to whether the delay deadline for that burst has expired or not. That is, utilization, in this sense, is a low-level metric that does not incorporate the Quality-of-Service (QoS) requirements. These requirements are incorporated into throughput η, which shall be shown in Fig. 10.

[48] Note that we measure utilization via the actual transmission of bits after the network traffic has been realized rather than by the fraction of slot-channel pairs that are allocated for transmission. For MC-LAPAL, in order for a slot-channel pair to be utilized, it must have been allocated in advance.

[49] Recall that this grid has M channels, which appear as rows, and T_{sch}/τ_{MAC} slots, which appear as columns, and hence contains a total of MT_{sch}/τ_{MAC} slot-channel pairs over one scheduling window.

[50] The reason that MC-LAPAL under perfect forecasts does not achieve exactly full utilization at $N = 660$ is that the traffic generation times of distinct devices may coincide; that is, the distribution of the generated traffic load over the scheduling window is not perfectly homogeneous.

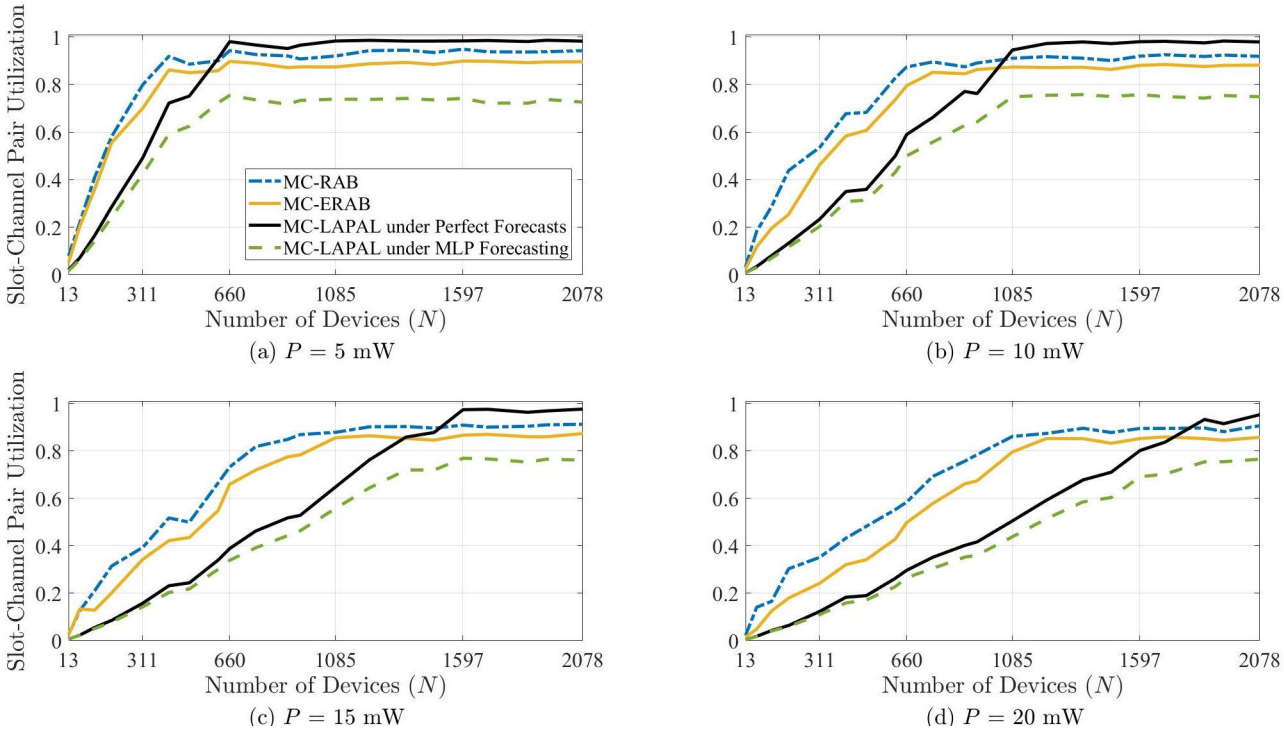


FIGURE 8. Fraction of utilization of the slot-channel grid.

all of the slot-channel pairs have been reserved, the utilization of MC-LAPAL under MLP forecasting saturates at approximately 0.75 beyond $N = 660$ devices.

We see a similar trend in the utilization of MC-LAPAL in Fig. 8(b), (c), and (d). As the maximum transmit power P per device increases, we see that saturation occurs at increasingly larger N. We also see that the saturation level for MC-LAPAL under MLP forecasting remains approximately the same across all P in this figure.

We now analyze the utilization of the reactive protocols in Fig. 8. First, even though MC-RAB is a reactive protocol, in our simulations, because the processing times are typically much greater than 1 slot, whenever a device i is able to access a given channel m for burst j, the remaining slots for the residual of that burst j are reserved for device i. Since the other devices transition to the waiting state during this reservation, MC-RAB is able to achieve a relatively high channel utilization that is competitive with the proactive protocol MC-LAPAL. Furthermore, we see that there exists a point of saturation for MC-RAB as well. This point of saturation shifts to the right (in N) as P increases.

Second, regarding the utilization of MC-ERAB in Fig. 8, recall that in addition to the mechanism of MC-RAB, MC-ERAB stipulates the extra condition that the total capacity (in bits) available across all of the selected channels is not less than the number of bits of residual $\tilde{j}$. If this condition is not satisfied, in the next time slot, device i repeats the entire process that begins with the selection of the set of channels anew until either the extra condition is satisfied or the delay deadline of residual $\tilde{j}$ has expired. That is, whenever MC-ERAB realizes that there is not sufficient total capacity to send the current residual, it re-attempts to randomly select a new set of channels until it encounters a set that has sufficient capacity for that residual. Due to the channel selection procedure of MC-ERAB, this protocol tends to arrive at a selection of channels that have capacities that are typically larger than those of the channels selected by MC-RAB. This leads to a lower utilization of the slot-channel grid for MC-ERAB than for MC-RAB. That is, MC-ERAB is able to utilize a fewer number of slot-channel pairs and achieves a higher average capacity per slot-channel pair than MC-RAB.

We see that the gap between the utilization of MC-RAB and that of MC-ERAB widens as P increases across Fig. 8(a), (b), (c) and (d). The reason is that the slot capacities are larger for higher values of P, and MC-ERAB is able to select these larger slot capacities as P increases.

Now, for each protocol, we define the "total allocated capacity" as the sum of the capacities of all of the slot-channel pairs that are allocated[51] by the protocol on the slot-channel grid (over a single scheduling window) for the entire set of devices. In Fig. 9, we display the total allocated capacity. First, we see that for all of the protocols under examination, the total allocated capacity grows approximately linearly until a point of saturation is reached

[51] In particular, for MC-RAB and MC-ERAB, in addition to the reserved slot-channel pairs, the slot-channel pairs on which a device accesses Gateway G without any collisions are also considered to have been allocated.

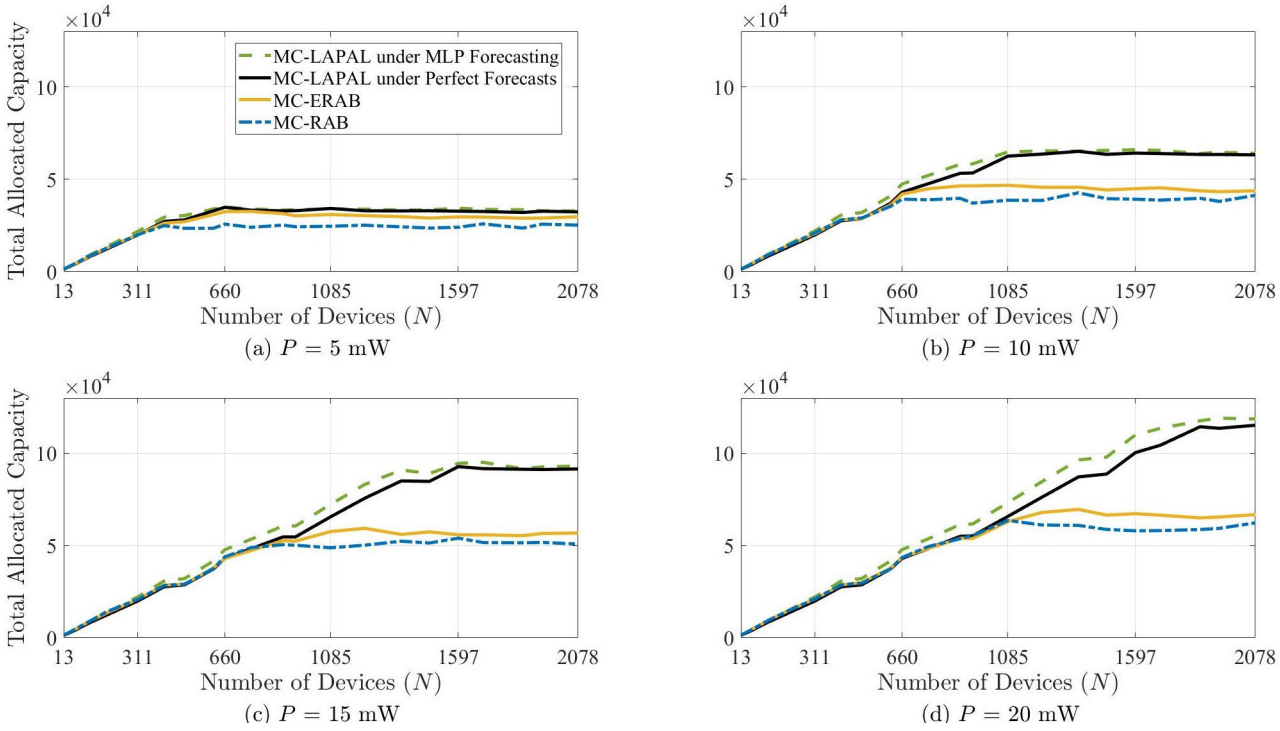


FIGURE 9. Total allocated capacity for all devices over the slot-channel grid.

and retains this saturation level beyond that point.[52] We see that for each of these protocols, the point of saturation shifts to the right as P increases. Furthermore, the saturation level also increases with P; however, the increase is faster for MC-LAPAL (under both perfect forecasts and MLP forecasting) than for the reactive protocols. Second, we see that MC-LAPAL under MLP forecasting overallocates the total capacity compared with that under perfect forecasts. We also see that this overallocation increases with P.

We see in Fig. 9(a) that the total allocated capacity for MC-RAB grows until $N = 416$ devices and saturates for larger N. A similar trend is observed across all of the subplots in this figure for MC-RAB, where the point of saturation shifts to the right and the saturation level rises as P increases. The reason that MC-LAPAL achieves a higher saturation level than MC-RAB for large P is that the small gap in channel utilization between MC-LAPAL and MC-RAB that appears in Fig. 8 is magnified in Fig. 9 due to the fact that the slot capacities grow larger as P increases.

We are now in a position to discuss our first high-level performance metric η, namely, the uplink cross-layer throughput (Section III-A), which we shall call "throughput" for short. The throughput performance appears in Fig. 10. In this figure, first, we see that the throughput for MC-LAPAL under MLP forecasting remains close to that under perfect forecasts as P increases. This implies that MLP forecasting serves as an effective forecasting scheme for MC-JFS.

[52] In this regard, we apply the same terminology as we did for Fig. 8.

Second, we see in Fig. 10 that MC-LAPAL outperforms both MC-RAB and MC-ERAB with respect to η. The performance difference between our MC-JFS heuristic, namely MC-LAPAL, and those of the reactive protocols, namely MC-RAB and MC-ERAB, increases as P increases. The main reason for this gap in throughput performance is that MC-LAPAL successfully uses the forecasts in scheduling the traffic in advance, whereas the reactive protocols can at best make reservations based on the traffic load that occurs at each time instant.

Third, in each of the four subplots of Fig. 10, we see that throughput begins its decline at an N that is smaller than the corresponding point of saturation in Fig. 8. By examining the resulting schedules, we have confirmed that the reason that its decline does not begin exactly at the point of saturation of utilization is as follows: Due to the existence of delay constraints on a sufficiently large fraction of devices that are less than the scheduling window, as N increases, the burst generation instances of distinct devices fall in closer proximity. For devices whose delay constraints are short, this causes a fraction of the bursts of these devices not to be scheduled, even though the utilization across the slot-channel grid has not yet saturated.

In Fig. 10(a), we see that the throughput of MC-ERAB is slightly lower than but comparable to those of MC-LAPAL under perfect forecasts. Furthermore, we see that the throughput of MC-ERAB is competitive with that of MC-LAPAL under MLP forecasting; however, MC-LAPAL significantly outperforms MC-ERAB as P increases. The reason is that

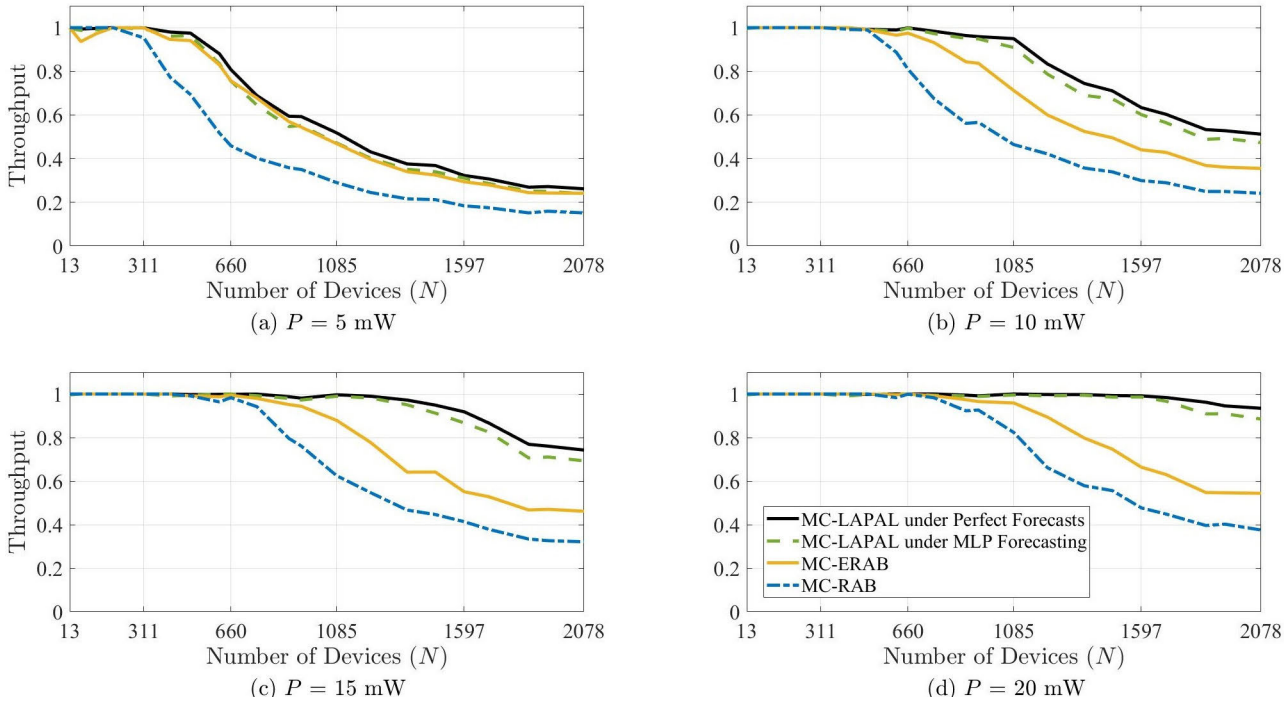


FIGURE 10. Uplink cross-layer throughput η.

MC-LAPAL orders the channels with respect to their capacities for the uplink from every device and selects the channels that have a larger capacity. In contrast, the channels in MC-ERAB are selected randomly. In addition, MC-LAPAL selects to transmit those bursts that have larger γ's, thereby giving priority to delivering those bursts with a larger number of bits before their delay deadlines. In contrast, MC-ERAB merely reacts to the actual bursts in the order in which they are generated and has no ability to prioritize bursts across distinct devices in advance.

Across all of the four subplots of Fig. 10, the reason that MC-ERAB outperforms MC-RAB by a significant margin is that due to the extra condition by which MC-ERAB enhances MC-RAB, MC-ERAB knows in advance whether the total capacity of the selected channels is sufficient for the current residual at hand. By repeating the channel selection procedure whenever MC-ERAB sees that the entire residual cannot be transmitted over the currently selected set of channels, it ensures that the only time that the residual is not delivered completely is due to the presence of a collision on at least one of the selected channels. In contrast, MC-RAB suffers a throughput decrease whenever the selected set of channels does not have sufficient capacity to deliver the current residual.

Fig. 11 displays the η-optimal values of the p_{RAB} and the p_{ERAB} parameters of MC-RAB and MC-ERAB protocols, respectively. First, we note that in general, these optimal values decrease with N in order to allow an increasing number of devices to access the IoT Gateway. Fig. 12 displays the η-optimal value of $1/\lambda$, where λ is the exponential back-off parameter that is common to both MC-RAB and MC-ERAB. We see that as N grows, the optimal number of MAC-layer slots for which a device that runs either of these protocols should back off is, on average, 1 to 4 slots for all P.

Fig. 13 displays the fraction of slot-channel pairs that experience collision in the slot-channel grid for MC-RAB and MC-ERAB. In this figure, we see that this fraction is less than 0.06 for both of these protocols. The reason that this fraction remains below this threshold for all N is that the parameters p_{RAB} and p_{ERAB} for MC-RAB and MC-ERAB respectively as well as λ for both of these protocols have been selected optimally via exhaustive search for each value of N, as shown in Fig. 11 and Fig. 12.

Fig. 14 displays the fraction of idle slot-channel pairs. For MC-LAPAL, this fraction plus the fraction of utilization in Fig. 8 equals 1.[53] In contrast, for each of MC-RAB and MC-ERAB, the fraction of idle slot-channel pairs, the fraction of utilization (Fig. 8) and the fraction of slot-channel pairs that experience collision (Fig. 13) sum to 1.

In Fig. 14(a), for MC-RAB, we see that the fraction of idle slot-channel pairs falls until $N = 416$, as the utilization (in Fig. 8) increases and the fraction of collisions in the slot-channel grid (in Fig. 13) remains below 0.04. This behavior is due to the fact that the traffic load is relatively light for small N, which leaves many of the slot-channel pairs idle. Beyond $N = 416$, we see that the fraction of idle slot-channel

[53]Note that utilization, which occurs after the realization of the actual traffic, rather than allocation is used in determining which slot-channel pairs remain idle.

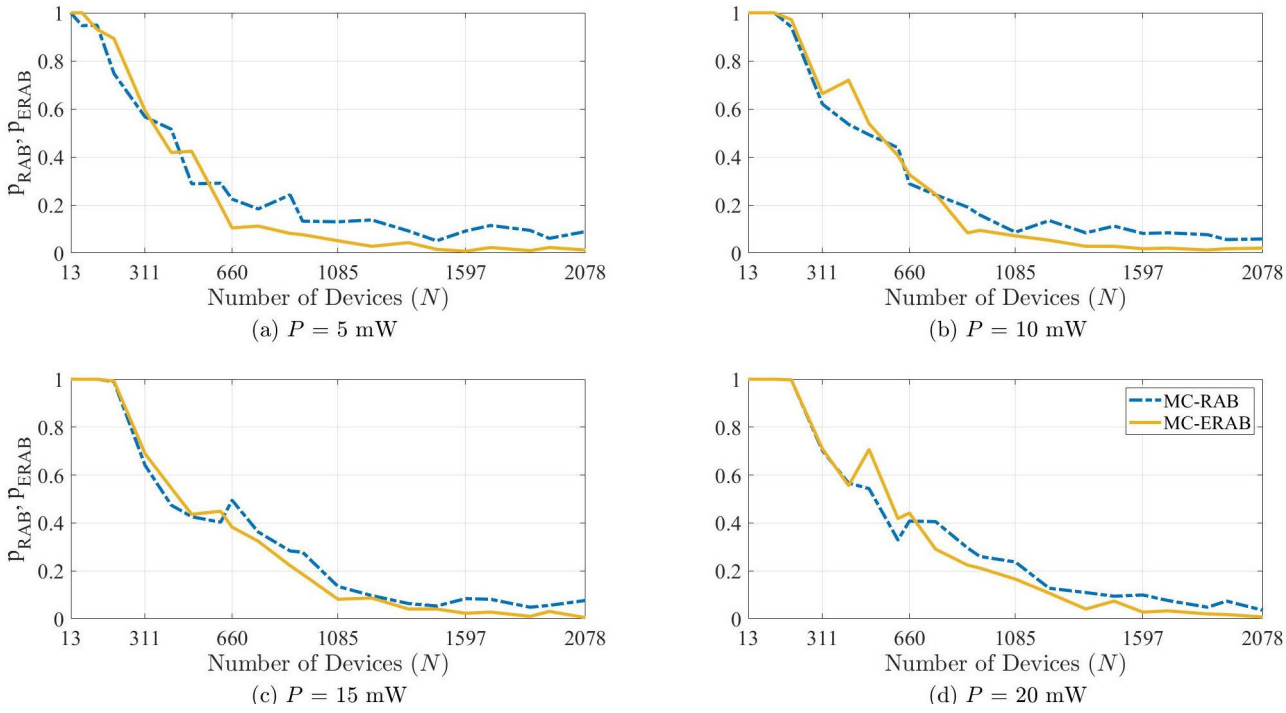


FIGURE 11. The optimal values of the p_{RAB} and p_{ERAB} parameters for the MC-RAB and MC-ERAB protocols, respectively.

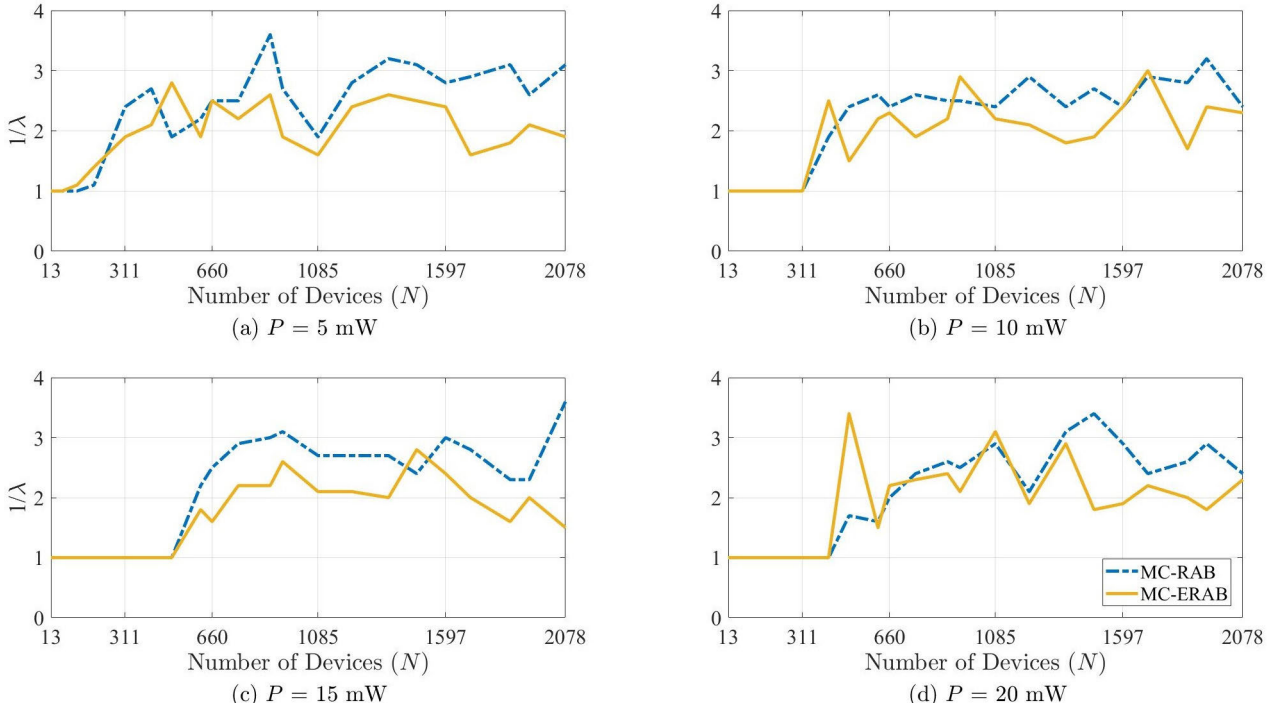


FIGURE 12. The optimal values of $1/\lambda$ for the MC-RAB and MC-ERAB protocols.

pairs is low as the utilization of MC-RAB saturates. We see that a similar trend for MC-RAB holds across all P. Furthermore, as P increases, the point of saturation for utilization shifts to the right. Concurrently, the point at which the fraction of idle slots falls to a constant value shifts to the right as P increases. In this figure, we also see that the fraction of

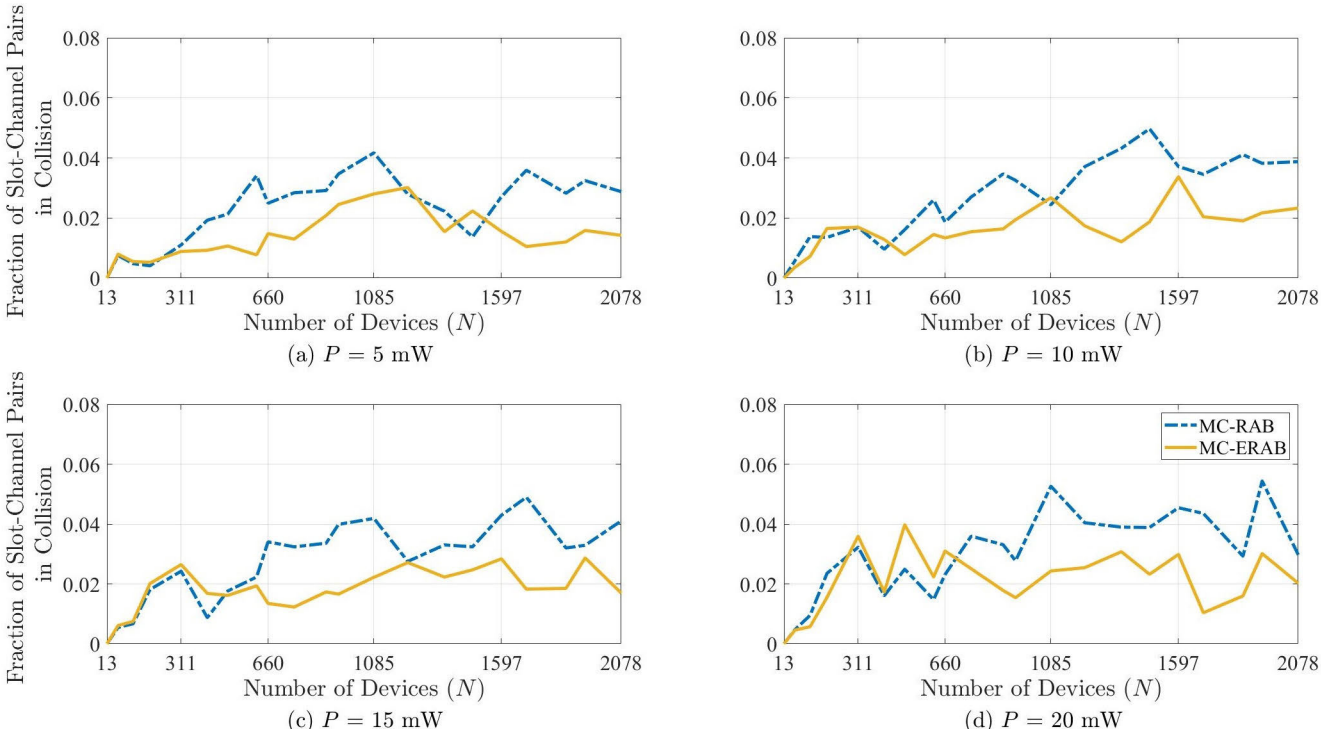


FIGURE 13. Fraction of slot-channel pairs that experience collision in the slot-channel grid.

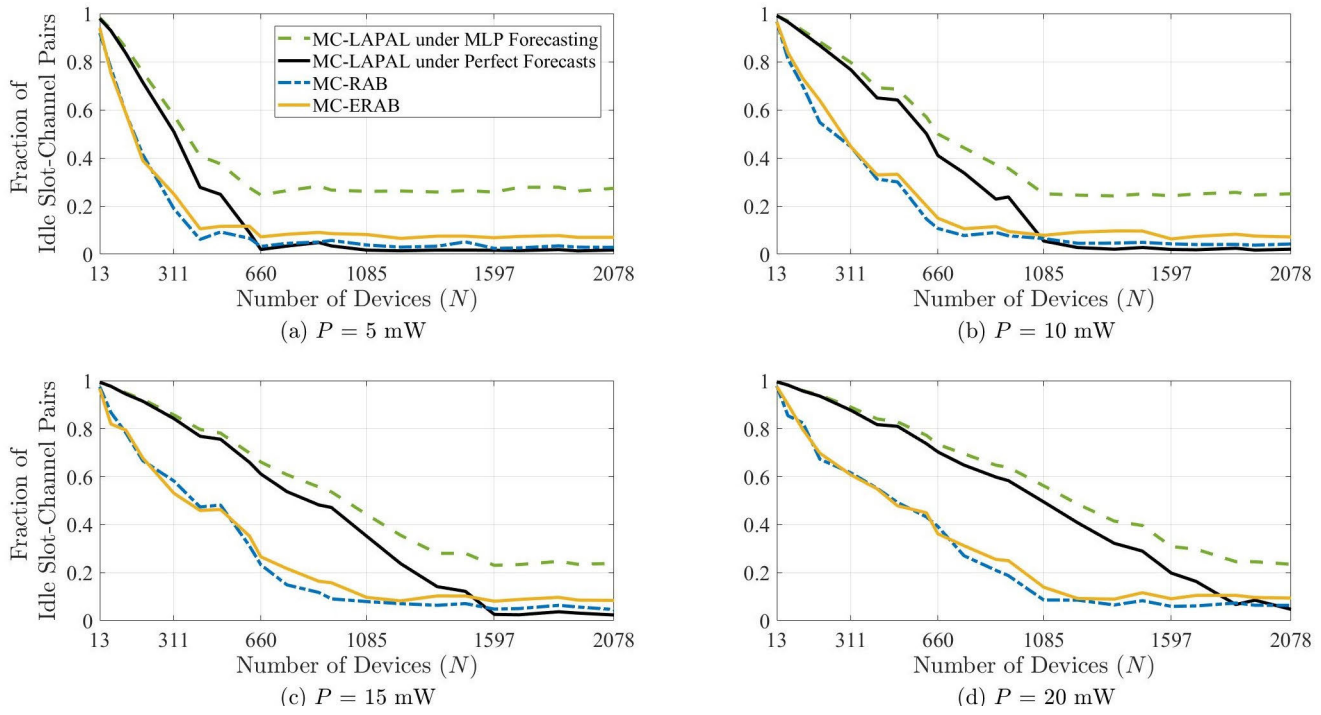


FIGURE 14. Fraction of idle slot-channel pairs in the slot-channel grid.

idle slot-channel pairs under MC-ERAB is comparable to but slightly larger, on average, than under MC-RAB for all N and all P.

We shall now present the results on our second high-level network performance metric, which relates to transmit energy consumption. Note that each device spends P/M Watts of

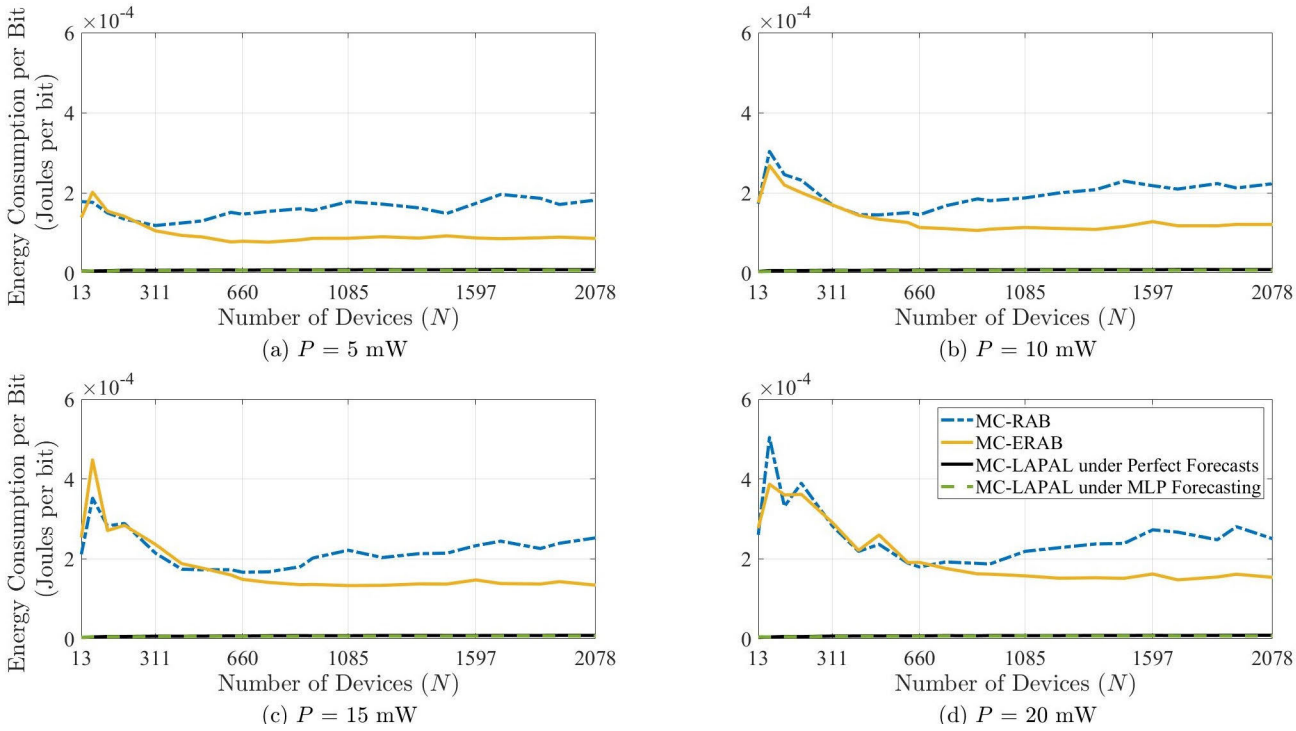


FIGURE 15. The metric $\mathcal{E}$, which is defined as the ratio of the total transmit energy consumption of all of the IoT devices in the network to the total number of bits in successfully delivered bursts.

transmit power whenever it transmits on any one of the slot-channel pairs. We define $\mathcal{E}$ as the ratio of the total transmit energy consumption of all devices in $\mathcal{N}$ to the total number of bits in successfully delivered bursts on the entire slot-channel grid.

The performance of all of the protocols with respect to $\mathcal{E}$ is displayed in Fig. 15. We now focus on Fig. 15(a). First, even though the throughput of MC-ERAB is close to that of MC-LAPAL in Fig 10(a), $\mathcal{E}$ for MC-ERAB is at least one order of magnitude larger than that of MC-LAPAL in Fig. 15(a). The reason is that MC-LAPAL orders the channels with respect to their capacities and successively uses the channel with the largest capacity among all channels that have not yet been used in scheduling the delivery of a forecast burst.[54] In contrast, MC-ERAB is not able to order all of the channels and select the channels with respect to their capacities in advance. As a consequence, MC-LAPAL is able to deliver the same traffic with much fewer slot-channel pairs than MC-ERAB. Since a fixed transmit energy is incurred per device per such slot-channel pair, MC-LAPAL achieves a much lower $\mathcal{E}$ than MC-ERAB.

Second, by zooming into Fig. 15 as shown in Fig. 16, we see that MC-LAPAL under MLP forecasting achieves a lower $\mathcal{E}$ than that under perfect forecasts. Third, in Fig. 15(a), we see that the energy consumption of MC-RAB is larger than that of MC-ERAB for $N > 311$. The reason for this result is two-fold: The primary reason is that since the throughput of MC-RAB is significantly lower than that of MC-ERAB beyond $N = 311$ devices, the denominator in the definition of $\mathcal{E}$, namely the number of bits in successfully delivered bursts, is smaller for MC-RAB than for MC-ERAB, which results in a larger value of $\mathcal{E}$ for MC-RAB. The secondary reason is that even though MC-ERAB is not able to order all of the channels with respect to their capacities, due to the extra condition imposed by MC-ERAB, it has the ability to reject a randomly selected set of channels if they do not possess a sufficiently large capacity to deliver the current burst. As a result, compared with MC-RAB, MC-ERAB arrives at a selection of channels that tend to have larger capacities compared with those selected by MC-RAB. When channels with larger capacities are selected, for the same traffic load, the number of slot-channel pairs on which devices transmit is reduced, thereby reducing $\mathcal{E}$. Due to these primary and secondary reasons, MC-RAB incurs the largest $\mathcal{E}$ for sufficiently large N, followed by MC-ERAB, which is, in turn, followed by MC-LAPAL. This trend is observed in all of the subplots of Fig. 15, across which P increases.

[54]The only exception is the last channel selected for a forecast burst, which is set as on Line 27 of the ScheduleBurst function in Fig. 6 in order to keep g close to 1 in Fig. 7.

2) PERFORMANCE COMPARISON FOR THE SPECIAL CASE OF IDENTICAL CHANNELS

For the case of identical channels, for brevity, we shall present the results only for the high-level network performance

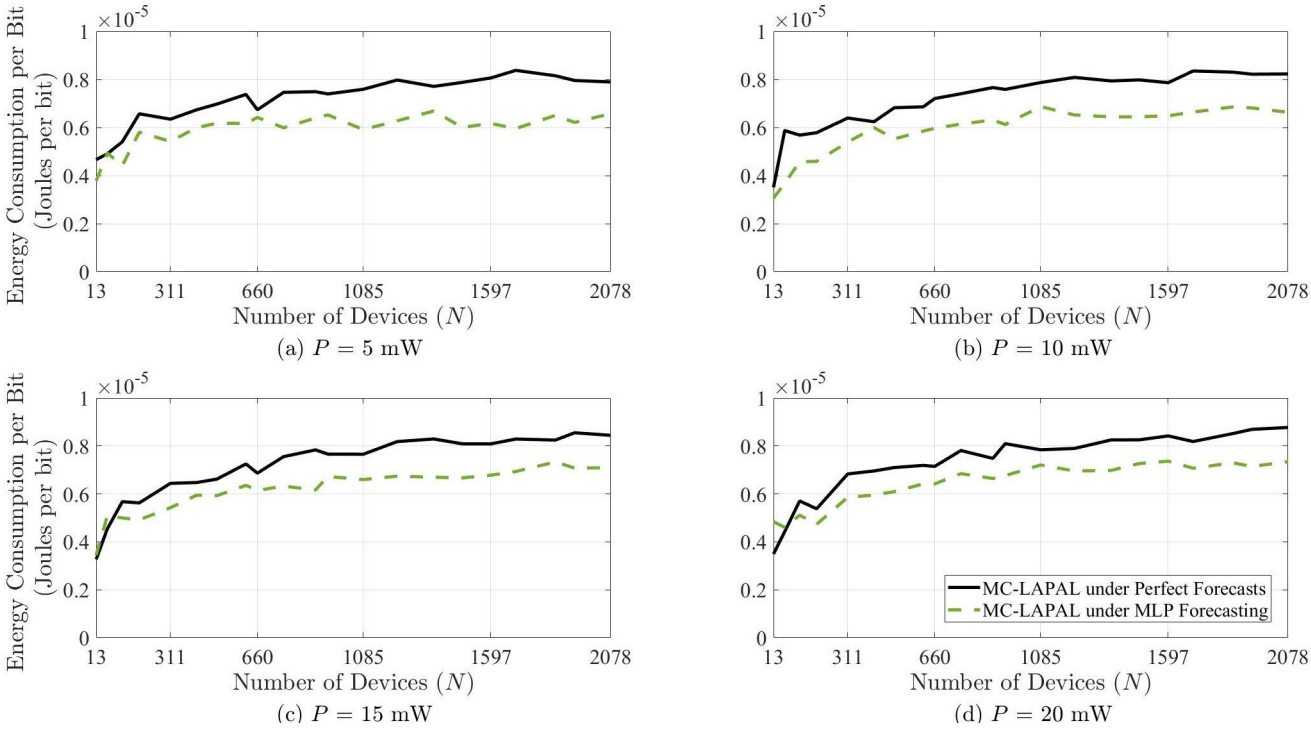


FIGURE 16. Zoom-in to Fig. 15 in order to compare $\mathcal{E}$ for MC-LAPAL under perfect forecasts versus under MLP forecasting.

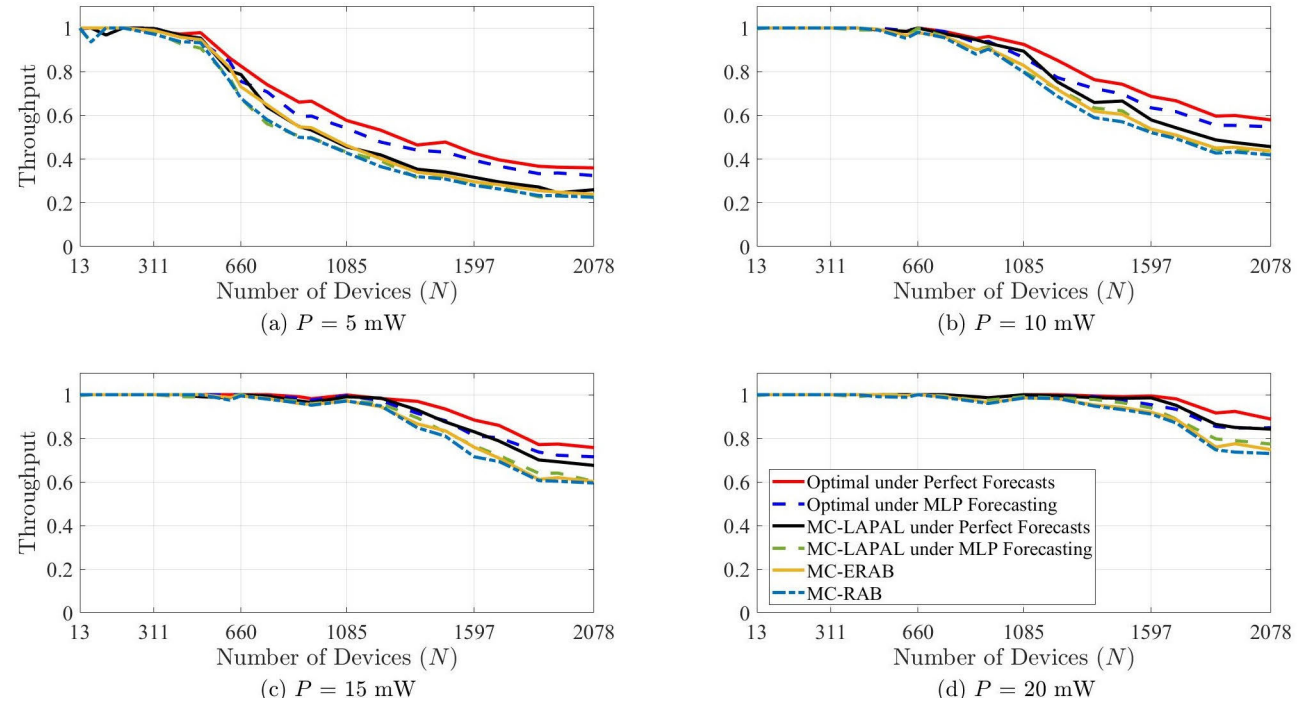


FIGURE 17. Throughput in the case of identical channel capacities for the uplink of each of the IoT devices.

metrics rather than the full set of low-level metrics that were presented for the general case. However, in our explanations, we shall draw upon the differences that we have observed in regard to the low-level metrics between the general case and the case of identical channels.

In Fig. 17, we display the throughput achieved by optimal scheduling[55] (under both perfect forecasts and MLP

[55]Note that the results for optimal scheduling were obtained for the special case of identical channels by virtue of the Reduced Two-Step Algorithm described in Section IV-B.

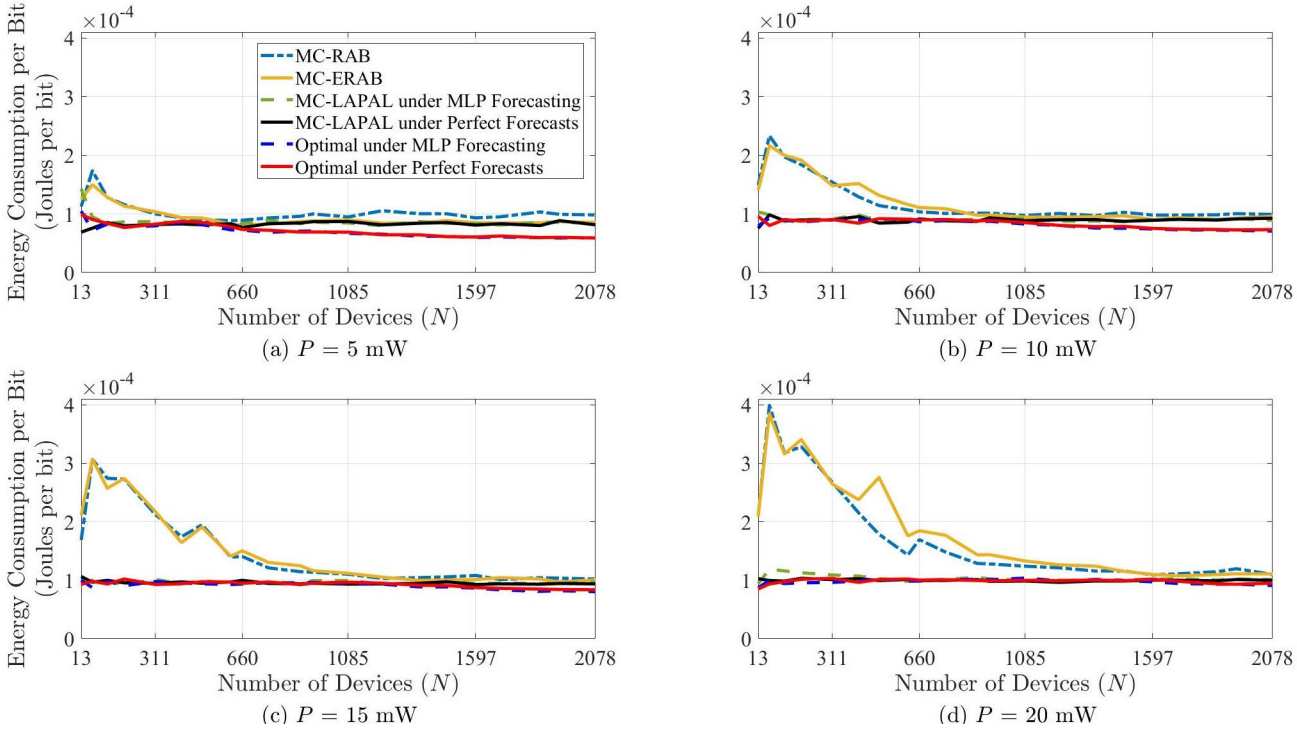


FIGURE 18. The ratio of the total transmit energy consumption of all devices to the total number of bits successfully delivered bursts in the case of identical channel capacities for the uplink of each of the IoT devices.

forecasting) in addition to those of MC-LAPAL, MC-ERAB and MC-RAB for the case of identical channels. First, in Fig. 17(a), we see that the gap between the throughput of optimal scheduling and that of MC-LAPAL widens beyond $N = 660$ devices. Furthermore, the performance of MC-LAPAL is close to those of the reactive protocols MC-RAB and MC-ERAB. However, we see that as P increases across the four subplots in Fig. 17, the throughput of MC-LAPAL approaches that of optimal scheduling while the gap between the throughput of MC-LAPAL and those of the reactive protocols widens.

Second, recalling that the capacity of a single uplink channel of any given IoT device was taken (Section V-A) to be the numerical average of the capacities of the M uplink channels of the same device, we see in Fig. 17 that the throughput differences among MC-LAPAL, MC-ERAB and MC-RAB are smaller than those obtained for the corresponding plots in Fig. 10 for the general case. The reason is that in the case of identical channels, the spread in capacities exists only across the uplink channels of distinct IoT devices, whereas in the general case, the capacity of each of the M uplink channels seen by an IoT device has a positive standard deviation around the mean capacity calculated for the case of identical channels. This fact reduces the throughput gains that both MC-LAPAL and MC-ERAB are able to attain against MC-RAB because the former protocols are not able to capitalize as much on the spread in these channel capacities for the case of identical channels as they do for the general case.

In Fig. 18, we display $\mathcal{E}$ for the case of identical channels. First, since the schedules produced by the optimal solver and by MC-LAPAL are collision-free, no transmit energy is spent except on successful transmissions. Hence, for all N, the $\mathcal{E}$ for each of these is approximately constant.[56] Second, in Fig. 18(a), we see that the values of $\mathcal{E}$ are similar for MC-RAB, MC-ERAB, MC-LAPAL and for optimal scheduling (under both perfect forecasts and MLP forecasting for the latter two schemes) across all N. However, as P grows across the four subplots in Fig. 18, the values of $\mathcal{E}$ for MC-RAB and MC-ERAB become increasingly larger than those of MC-LAPAL and optimal scheduling below approximately $N = 660$ in Fig. 18(b), $N = 1085$ in Fig. 18(c), and $N = 1597$ in Fig. 18(d). However, for values of N that are larger than these approximate thresholds, the values of $\mathcal{E}$ achieved by MC-RAB and MC-ERAB remain close to those attained by MC-LAPAL and optimal scheduling.

C. COMPUTATION TIME

In this subsection, our goal is two-fold: (1) We present our results on the computation time[57] of MC-LAPAL for the general case. In addition, we compare the computation time of optimal scheduling, which utilizes the Reduced Two-Step Algorithm, against that of MC-LAPAL for the special case of identical channels. (2) We discuss the real-time operation of JFS and show that the computation time of scheduling

[56] For these proactive scheduling protocols, the reason that $\mathcal{E}$ is not exactly constant for all N is that the average fraction of the total allocated capacity of a slot-channel pair varies as a function of N.

[57] In this paper, the term "computation time" refers to the execution time of the algorithm in question.

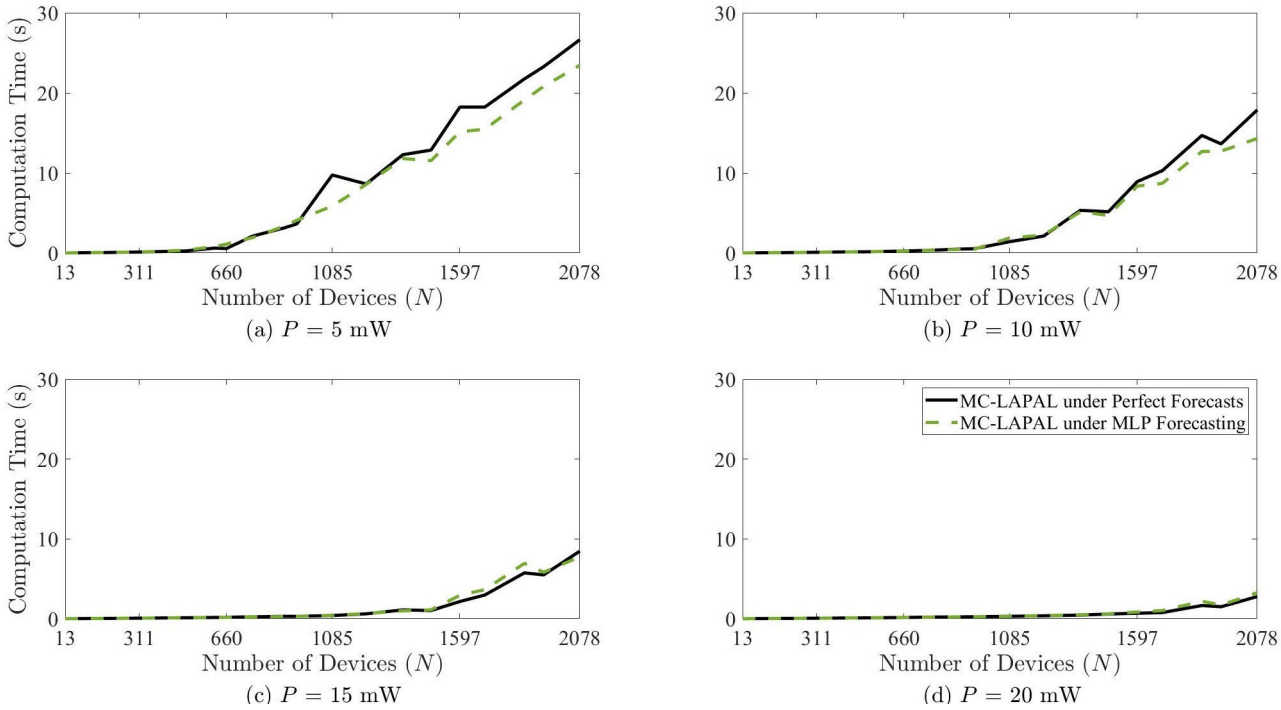


FIGURE 19. Computation time of MC-LAPAL for the general case.

empirically dominates over that of forecasting in our simulations on MC-JFS.

1) RESULTS ON THE COMPUTATION TIME

In Fig. 19, we display the computation time of MC-LAPAL for the general case. First, across all P, we see that the computation time increases very slowly in N in that regime in which the throughput in Fig. 10 has not yet begun its visible decline to below 1. For each P, beyond this point, the computation time grows approximately linearly in N. Second, we see that for fixed N, the computation time grows smaller as P increases.

In Fig. 20, we display the computation time of optimal scheduling that uses the Reduced Two-Step Algorithm (Section IV-B) and that of MC-LAPAL for the special case of identical channels. We see that the computation time for optimal scheduling grows slowly in N up to the point at which the throughput in Fig. 17 begins its visible decline to below 1. Beyond this point, the computation time of optimal scheduling grows significantly in N. In contrast, while the same trend is observed for MC-LAPAL, its computation time remains at least two orders of magnitude smaller than that of optimal scheduling for those values of N beyond the point at which the throughput begins to fall visibly below 1.

The empirical results shown in Fig. 19, which were *not* obtained for the worst-case scanerio (that maximizes the time complexity), have trends that are similar to those derived in Theorem 3 in Section IV-C3 on the worst-case time complexity of MC-LAPAL. Noting that V and M in the statement of Theorem 3 are fixed in our simulation set-up, Theorem 3 implies that for large J, the upper bound obtained on the worst-case time complexity grows asymptotically as $J \log J$. In our simulations, due to the bootstrapping of the traffic generation patterns from a representative set of IoT devices, the total number of bursts J to be scheduled over a single scheduling window grows approximately linearly with the total number of devices N. Thus, the approximately linear asymptotic growth of the computation time of MC-LAPAL in all of the subplots of Fig. 19 suggests that the empirical time complexity is close to that obtained analytically for the worst case.[58]

2) REAL-TIME OPERATION OF JFS

The execution time T^{JFS} of a JFS system at an IoT Gateway is the sum of the execution time T^{FM} of the Forecasting Module and the execution time T^{SM} of the Scheduling Module in Fig. 1; that is, $T^{JFS} = T^{FM} + T^{SM}$. The reason is that the Scheduling Module must wait for all of the forecasters in the Forecasting Module to finish their forecasts before the forecast traffic can be scheduled. We let T_i^{fc} denote the execution time of the forecaster of device i. Since all of the forecasters run in parallel, the forecaster that finishes last determines the execution time of the Forecasting Module; that is, $T^{FM} = \max_{i \in \mathcal{N}} T_i^{fc}$.

[58] Although not visible on the scale of the plots in Fig. 20, the same approximately linear asymptotic growth in J for MC-LAPAL also holds for the empirical results in the case of identical channels.

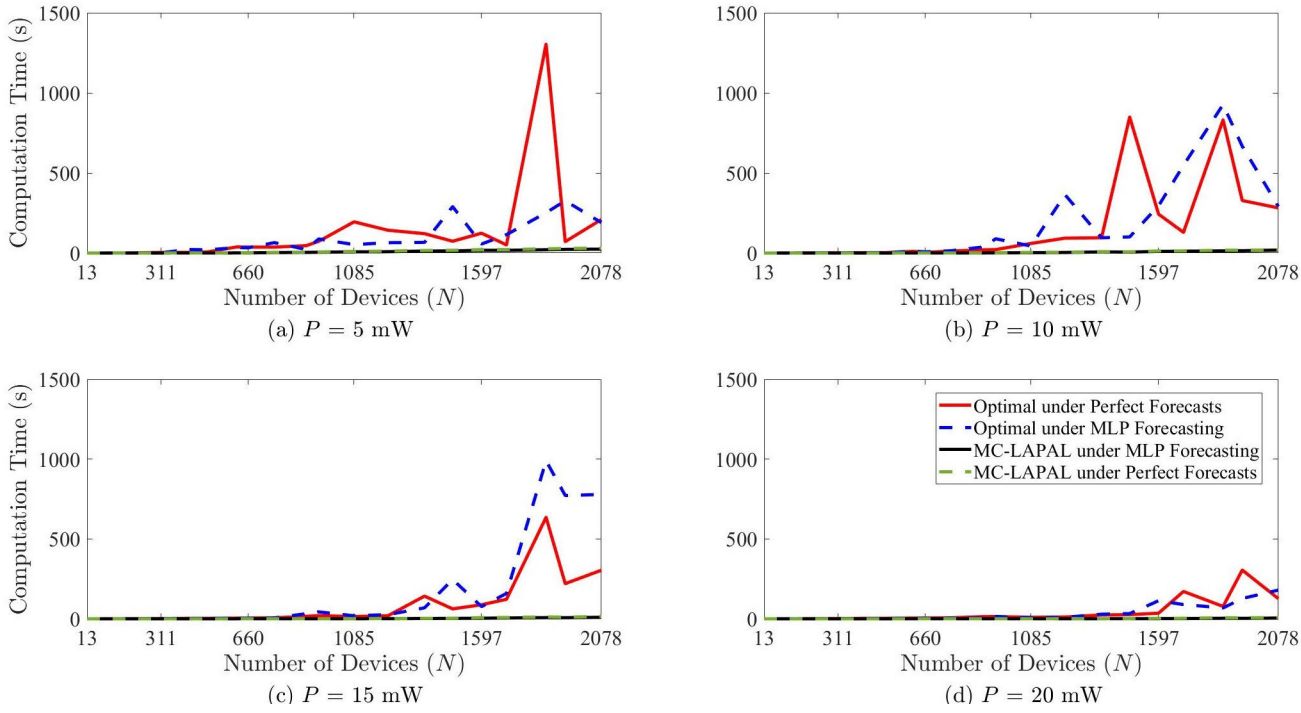


FIGURE 20. Computation time of optimal scheduling and MC-LAPAL for the special case of identical channels.

In order for a JFS system to operate in real time, the schedule for all of the devices on a given scheduling window must be ready (and communicated to the devices on the downlink) by the beginning of the scheduling window. This implies that for each scheduling window, the JFS computation must begin at least a duration T^{JFS} before the beginning of the scheduling window. Thus, for real-time operation, the Forecasting Module *cannot* use the past samples of the traffic generation patterns of the devices that accumulate within a duration T^{JFS} before the beginning of the scheduling window. That is, for the real-time operation of a JFS system, the feature selection at the forecaster must be performed in a manner such that no past input samples are selected as features within a duration of T^{JFS} before the scheduling window begins. We shall undertake a complete examination of the impact of the execution time T^{JFS} on network performance in our future work.

We now discuss the contributions of T^{FM} and T^{SM} to T^{JFS} in our simulations: As stated in Section V-A1, since the same data set is used as in [22], we utilize Table 3 in [22] to form a conservative estimate of $\max_{i\in\mathcal{N}} T_i^{fc}$ as the sum of the mean and two standard deviations measured over the set of MLP-based forecasters. As can be seen in Table 3 of that work, the largest such quantity holds for the VBP class in the data set for which the mean is 0.004 s and the standard deviation is 0.001 s; hence, $T^{FM} = 0.006$ s serves as our approximation. Compared with the computation time for multi-channel scheduling in Fig. 19 and 20, we see that for $N > 1597$ devices, across all P, T^{SM} is at least several orders of magnitude larger than T^{FM}.[59] Hence, we see that empirically, the execution time for scheduling dominates over that of forecasting in the total execution time T^{JFS}. Hence, all of the results on the computation time of scheduling that appear in Fig. 19 and 20 can be practically taken as the value of T^{JFS} for each simulation.

VI. CONCLUSION

In Multi-Channel Joint Forecasting-Scheduling (MC-JFS), an IoT Gateway forecasts the future traffic generation patterns of individual IoT devices in its coverage area and schedules the future traffic of these devices in advance on multiple channels over a scheduling window. This approach to the solution of the massive access problem of IoT stands in sharp contrast with reactive protocols in which the IoT Gateway merely reacts to the current traffic demand.

In this paper, we have focused on the development of MAC-layer scheduling schemes for MC-JFS. First, we developed a proactive scheduling heuristic, named MC-LAPAL, for the general case. Second, for the special case in which the uplink channel capacities seen by an individual IoT device are identical, we developed a reduction of this problem by virtue of which an optimal scheduling solution can be obtained in real time.

[59] Since we used the same computational platform (stated in Section V-A3 for both forecasting and scheduling, we expect the ratio of T^{SM} to T^{FM} to remain approximately the same if we scale the computational resources available.

For the general case, we compared the network performance of MC-LAPAL against those of two multi-channel reactive protocols, namely MC-RAB and MC-ERAB. We showed that the combination of MLP for forecasting and MC-LAPAL for multi-channel scheduling outperforms both MC-RAB and MC-ERAB. Furthermore, the performance gap widens as the maximum transmit power of an IoT device increases and the number of IoT devices in the coverage area grows. For the special case in which the uplink channel capacities from an IoT device to the IoT Gateway are identical, we showed that the combination of MLP forecasting and our MC-LAPAL heuristic provides a network performance that is close to that of optimal scheduling under perfect forecasts. Furthermore, we showed that the computation time of MC-LAPAL scales approximately linearly with the number of devices. These results jointly imply that MC-LAPAL serves as an effective multi-channel scheduling heuristic for MC-JFS.

In our future work, first, we plan to generalize MC-JFS from the single-gateway setting in this paper to one with multiple gateways in order to accommodate mobile IoT devices that change coverage areas. Second, we shall investigate new scheduling heuristics targeted at IoT devices with tighter delay constraints. Third, we plan to apply the general framework that we have developed in this paper to Multiple Input Multiple Output (MIMO) systems in the presence of spatial channels. Finally, we plan to develop novel methodologies to simulate tens of thousands of IoT devices in a given coverage area, as are expected to be found in smart cities of the near future.

APPENDIX

In this appendix, we present the proofs of all of the theorems in this article.

Proof of Theorem 1 (Optimality via the Two-Step Algorithm): Let $\mathbf{W}$ denote the 3D matrix of w_{mjt}'s, $\tilde{\mathbf{W}}$ the 3D matrix of $\tilde{w}_{mjt}$'s, $\mathbf{u}$ the vector of u_j's, and $\mathbf{Q}$ the 3D matrix of Q_{mjt}'s in all of the programs in which these variables appear. We shall show the following two facts: (I) A feasible solution of (1)-(5) is a feasible solution of (6)-(8). (II) The $(\mathbf{W}, \mathbf{u})$ produced by the Two-Step Algorithm is an optimal solution of (6)-(8) and a feasible solution of (1)-(5). These two facts then imply that the Two-Step algorithm returns an optimal solution of (1)-(5).

Now, in order to show (I), consider a feasible solution of (1)-(5). Then, under the substitution of w_{mjt} for each $\tilde{w}_{mjt}$ in (6)-(8), we note that (2) is identical to (7), and that (3) and (5) imply (8). Thus, all of the constraints in (6)-(8) are implied by the constraints in (1)-(5). Thus, every feasible solution of the former is a feasible solution of the latter program.

In order to show (II), we note that the tightening procedure performed on the optimal solution of (6)-(8) produces a solution that satisfies the constraints (7) and (8) while retaining the same value of the objective function in (6). Hence, the $\mathbf{W}$ produced at the end of Step 2 is an optimal solution of (6)-(8) (under the substitution of w_{mjt} for $\tilde{w}_{mjt}$). Now, we augment the $(\mathbf{W}, \mathbf{u})$ obtained at the end of Step 2 by computing the corresponding matrix $\mathbf{Q}$ under the definition of Q_{mjt} as the number of bits of burst j scheduled on channel m in slot t. Based on this augmentation, we let $\mathbf{p}^*$ denote the $(\mathbf{W}, \mathbf{u}, \mathbf{Q})$ found by the Two-Step Algorithm. Then, $\mathbf{p}^*$ satisfies (2) since it satisfies (7). Now, note the following facts: First, $\mathbf{p}^*$ satisfies (3) due to the tightening procedure in Step 2. Second, $\mathbf{p}^*$ satisfies (4) by the fact that the construction in Step 2 tightens those bursts with $u_j = 1$ and does not schedule the bursts with $u_j = 0$. Third, $\mathbf{p}^*$ satisfies (5) by the definition of $\mathbf{Q}$ and the tightening procedure in Step 2. Hence, $\mathbf{p}^*$ is a feasible solution of (1)-(5).

Thus, (I) and (II) above imply that the Two-Step algorithm returns an optimal solution of (1)-(5).

Proof of Theorem 2 (Reduction for Identical Channels): In Step 1, since $y_{jt} = \sum_{m \in \mathcal{M}} w_{mjt}$ and $R_{mj} = R_j$ for identical channels, (11) and (14) are equivalent. Summing (10) over m shows that it implies (13). Now, since the objective functions (9) and (12) are identical, this implies that every optimal 3D matrix of w_{mjt}'s in (9)-(11) has associated with it, a unique optimal solution of y_{jt}'s in (12)-(14). Now, given any optimal solution $\mathbf{y}^* \equiv \{\{y_{jt}\}\}$ of (12)-(14), the 3D matrix of w_{mjt}'s produced by Step 2 of the Reduced Two-Step Algorithm satisfies (10) by construction and is an optimal solution of (9)-(11) that is associated with $\mathbf{y}^*$.

Proof of Theorem 3 (Time Complexity of MC-LAPAL): For brevity, (1) a programming statement in the pseudo-codes in Fig. 5 and 6 shall be said to "be $\mathcal{O}(F)$" (where F is any mathematical expression of the parameters V, J and M) if and only if that programming statement has worst-case time complexity $\mathcal{O}(F)$, and (2) any such programming statement that is *not* mentioned explicitly in the rest of this proof has been evaluated to be $\mathcal{O}(1)$.

In Fig. 5, Lines 4 and 5 are $\mathcal{O}(J)$ and $\mathcal{O}(MV)$, respectively. Now, we shall divide the computation of the worst-case time complexity of the rest of the MC-LAPAL function (including[60] its subroutine *ComputeSchedule*) into two parts, which are labeled (I) and (II) below.

(I) We shall evaluate the worst-case time complexity of the collection of statements in *ComputeSchedule* except the call to the *ScheduleBurst* function that appears on Line 20 of Fig. 5. To this end, note that each of the Lines 13 and 15 is $\mathcal{O}(J)$, and Line 16 is $\mathcal{O}(J \log J)$. Thus, the sequence of programming statements on Lines 12-18 is $\mathcal{O}(J \log J)$. Now, on Line 6, the number of subgrids, namely length($\langle \mathcal{D} \rangle$), is $\mathcal{O}(\min(J, V))$. Thus, the worst-case time complexity incurred by calling Lines 12-18 in the *for* loop on Line 6 is $\mathcal{O}((J \log J) \min(J, V))$, which corresponds

[60]For the purposes of this proof, we encourage the reader to think of the *ComputeSchedule* function, not as a separate function (as shown in Fig. 5) but rather as if the pseudo-code of *ComputeSchedule* were written directly into Line 7, thereby eliminating the function call.

to the first term of the sum in the statement of this theorem.[61]

(II) We shall evaluate the worst-case time complexity incurred over the entire scheduling window by all of those programming statements (in the functions *MC-LAPAL* and in *ComputeSchedule*) that lead to a call to the *ScheduleBurst* function. To this end, let H denote the total number of calls to the *ScheduleBurst* function over the entire scheduling window. The worst-case time complexity of this part will be computed as that of the product of H with the worst-case time complexity of the *ScheduleBurst* function. To this end, in (II.1) below, we shall show that H is $\mathcal{O}(J)$. Second, in (II.2) below, we shall show that the *ScheduleBurst* function is $\mathcal{O}(M \log M + MV)$. The product of $\mathcal{O}(J)$ and $\mathcal{O}(M \log M + MV)$ will then give the second and third terms in the sum that appears in the statement of this theorem. We now show these results:

(II.1) For subgrid k, let l_k, s_k and c_k denote the number of times that the *ScheduleBurst* function returns the status *leftOver*, *skipped*, and *completed*, respectively, on Line 20 of Fig. 5. Thus, $H = \sum_k l_k + \sum_k s_k + \sum_k c_k$, where each sum is performed over all of the subgrids over the scheduling window. Now, since there can be at most one left-over burst in each subgrid, $l_k \leq 1$. Thus, $\sum_k l_k \leq \min(J, V)$. Now, $\sum_k s_k + \sum_k c_k$ is $\mathcal{O}(J)$. Thus, H is $\mathcal{O}(\min(J, V) + J)$. Since $\min(J, V) + J \leq 2J$, H is $\mathcal{O}(J)$.

(II.2) In Fig. 6, each of the Lines 9 to 12, 15 and 20 is $\mathcal{O}(MV)$.[62] On Line 21, computing the capacities that correspond to all of the channels is $\mathcal{O}(M)$, and sorting the channel capacities is $O(M \log M)$; hence, Line 21 is $\mathcal{O}(M \log M)$. The number of times that the *for* loop that begins on Line 22 iterates is $\mathcal{O}(M)$. Furthermore, Line 24 is $\mathcal{O}(V)$ since it iterates over the time slots for a given channel. The number of times that the *for* loop that begins on Line 25 iterates is $\mathcal{O}(V)$. Note that Lines 27 to 32 are called only once due to the *break* statement on Line 32. Line 27 is $\mathcal{O}(M)$, and Line 28 is $\mathcal{O}(V)$. Thus, Lines 27 and 28 are $\mathcal{O}(M + V)$.[63] Putting these results together, we see that Lines 22-40 are $\mathcal{O}(MV + (M + V))$. In this expression, the first term is due to the multiplication of the number of times that the outer (Line 22) and inner (Line 25) *for* loops are iterated through any of the $\mathcal{O}(1)$ statements that appear in this double *for* loop. The second term is due to Lines 27 and 28. Since $MV \geq M + V$ for integers $M \geq 2$ and $V \geq 2$, we see that Lines 22-40 are $\mathcal{O}(MV)$. Adding the time complexity incurred on Line 21, namely $\mathcal{O}(M \log M)$, to that incurred on Lines 22-40, we obtain the result that the *ScheduleBurst* function is $\mathcal{O}(M \log M + MV)$.

The time complexity of MC-LAPAL is the sum of the time complexities of Lines 4 and 5 in Fig. 5, namely $\mathcal{O}(J)$ and $\mathcal{O}(MV)$, respectively, and the results of (I) and (II) above. That is, the time complexity of MC-LAPAL is $\mathcal{O}(J) + \mathcal{O}(MV) + \mathcal{O}((J \log J) \min(J, V)) + \mathcal{O}(JM \log M + JMV) = \mathcal{O}((J \log J) \min(J, V) + JM \log M + JMV)$.

Proof of Theorem 4 (Space Complexity of MC-LAPAL): For brevity, we utilize the same two conventions that appear in the first paragraph of the proof of Theorem 3, except that each reference to complexity is to space complexity (rather than to time complexity).

First, each of the globally available parameters r_j, d_j, a_j, and Δ_j, which ranges over $\mathcal{J}$, is $\mathcal{O}(J)$. Furthermore, $\mathbf{C}$ is $O(MJ)$, and $\mathcal{M}$ is $O(M)$. Hence, the worst-case space complexity incurred due to the representation of the global parameters is $\mathcal{O}(J) + \mathcal{O}(MJ) + \mathcal{O}(M) = \mathcal{O}(MJ)$.

Second, in Fig. 5, on Line 2, $\langle \mathcal{J}_{active} \rangle$ is $\mathcal{O}(J)$. On Line 4, $\langle \mathcal{D} \rangle$ is $\mathcal{O}(\min(J, V))$. On Line 5, $\mathbf{S}$ is $\mathcal{O}(MV)$. Each of the $\langle \tilde{\mathcal{J}} \rangle$, $\langle \mathcal{J}' \rangle$, $\langle \gamma[t_{\text{init}}] \rangle$, and $\langle \mathcal{J}_{sorted} \rangle$ is $\mathcal{O}(J)$ on Lines 13, 14, 15 and 17, respectively. As a result, the worst-case space complexity incurred within the *MC-LAPAL* and the *ComputeSchedule* functions in Fig. 5 is $\mathcal{O}(J) + \mathcal{O}(\min(J, V)) + \mathcal{O}(MV) + \mathcal{O}(J) = \mathcal{O}(J + MV)$.

Third, in Fig. 6, each of the $\mathbf{S}_{\text{emptyWithinSubgrid}}$ and $\mathbf{S}_{\text{emptyUpToDeadline}}$ on Lines 9 and 10 is $\mathcal{O}(MV)$, respectively. Each of the $\langle \mathcal{M}_+ \rangle$ and $\langle \mathcal{M}_+^{\text{sorted}} \rangle$ on Lines 20 and 21 is $\mathcal{O}(M)$, respectively. Furthermore, each of the $\langle \mathcal{T}_+^m \rangle$ and $\langle \mathcal{T}_+^{m^*} \rangle$ on Lines 24 and 28 is $\mathcal{O}(V)$, respectively. As a result, the worst-case space complexity incurred within the *ScheduleBurst* function in Fig. 6 is $\mathcal{O}(MV) + \mathcal{O}(M) + \mathcal{O}(V) = \mathcal{O}(MV)$.

Summing over the worst-case space complexities obtained in all of the three parts above, we have $\mathcal{O}(MJ) + \mathcal{O}(J + MV) + \mathcal{O}(MV) = \mathcal{O}(MJ + MV) = \mathcal{O}(M(J + V))$.

[61] Intuitively, this first term corresponds to the following facts: Sorting the channels on Line 16 of Fig. 5, which is $\mathcal{O}(J \log J)$, is the dominant term in the programming statements on Lines 12-18 that appear in the *ComputeSchedule* function. This dominant term is multiplied by the number of times for which the *for* loop iterates on Line 6, which is $\mathcal{O}(\min(J, V))$.

[62] On Lines 11 and 12, we compute the total capacity within each subgrid and that up to the deadline by summing the capacities of the slot-channel pairs. That is, in placing the bound on the time complexity of this step, we use this more efficient implementation, which does not use matrix multiplication.

[63] Due to the *break* statement on Line 32, this term is not multiplied by the number of iterations in the outer and the inner *for* loops.

REFERENCES

[1] F. Ghavimi and H.-H. Chen, "M2M communications in 3GPP LTE/LTE-A networks: Architectures, service requirements, challenges, and applications," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 2, pp. 525–549, 2nd Quart., 2015.

[2] M.-Y. Cheng, G.-Y. Lin, H.-Y. Wei, and A. Hsu, "Overload control for machine-type-communications in LTE-advanced system," *IEEE Commun. Mag.*, vol. 50, no. 6, pp. 38–45, Jun. 2012.

[3] A. Zanella, M. Zorzi, A. F. dos Santos, P. Popovski, N. Pratas, C. Stefanovic, A. Dekorsy, C. Bockelmann, B. Busropan, and T. A. H. J. Norp, "M2M massive wireless access: Challenges, research issues, and ways forward," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2013, pp. 151–156.

[4] *RACH Intensity of Time Controlled Devices*, document TSG RAN WG2 #69bis, R2-102296, 3GPP, Vodafone, Apr. 2010. [Online]. Available: http://www.3gpp.org/DynaReport/TDocExMtg–R2-69b–28031.htm

[5] S.-Y. Lien, T.-H. Liau, C.-Y. Kao, and K.-C. Chen, "Cooperative access class barring for machine-to-machine communications," *IEEE Trans. Wireless Commun.*, vol. 11, no. 1, pp. 27–32, Jan. 2012.

[6] Z. Alavikia and A. Ghasemi, "Collision-aware resource access scheme for LTE-based machine-to-machine communications," *IEEE Trans. Veh. Technol.*, vol. 67, no. 5, pp. 4683–4688, May 2018.

[7] J. Liu, W. Zhou, and L. Song, "A novel congestion reduction scheme for massive Machine-to-Machine communication," *IEEE Access*, vol. 5, pp. 18765–18777, 2017.

[8] Y.-C. Pang, S.-L. Chao, G.-Y. Lin, and H.-Y. Wei, "Network access for M2M/H2H hybrid systems: A game theoretic approach," *IEEE Commun. Lett.*, vol. 18, no. 5, pp. 845–848, 2014.

[9] L. Tello-Oquendo, I. Leyva-Mayorga, V. Pla, J. Martinez-Bauset, J.-R. Vidal, V. Casares-Giner, and L. Guijarro, "Performance analysis and optimal access class barring parameter configuration in LTE-A networks with massive M2M traffic," *IEEE Trans. Veh. Technol.*, vol. 67, no. 4, pp. 3505–3520, Apr. 2018.

[10] Y. Liu, C. Yuen, X. Cao, N. U. Hassan, and J. Chen, "Design of a scalable hybrid MAC protocol for heterogeneous M2M networks," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 99–111, Feb. 2014.

[11] T.-M. Lin, C.-H. Lee, J.-P. Cheng, and W.-T. Chen, "PRADA: Prioritized random access with dynamic access barring for MTC in 3GPP LTE-A networks," *IEEE Trans. Veh. Technol.*, vol. 63, no. 5, pp. 2467–2472, Jun. 2014.

[12] H. Jin, W. T. Toor, B. C. Jung, and J.-B. Seo, "Recursive pseudo-Bayesian access class barring for M2M communications in LTE systems," *IEEE Trans. Veh. Technol.*, vol. 66, no. 9, pp. 8595–8599, Sep. 2017.

[13] L. Tello-Oquendo, D. Pacheco-Paramo, V. Pla, and J. Martinez-Bauset, "Reinforcement learning-based ACB in LTE-A networks for handling massive M2M and H2H communications," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–7.

[14] A. Aijaz and A. H. Aghvami, "Cognitive machine-to-machine communications for Internet-of-Things: A protocol stack perspective," *IEEE Internet Things J.*, vol. 2, no. 2, pp. 103–112, Apr. 2015.

[15] N. Shahin, R. Ali, and Y.-T. Kim, "Hybrid slotted-CSMA/CA-TDMA for efficient massive registration of IoT devices," *IEEE Access*, vol. 6, pp. 18366–18382, 2018.

[16] A. Aijaz, S. Ping, M. R. Akhavan, and A.-H. Aghvami, "CRB-MAC: A receiver-based MAC protocol for cognitive radio equipped smart grid sensor networks," *IEEE Sensors J.*, vol. 14, no. 12, pp. 4325–4333, Dec. 2014.

[17] I. Park, D. Kim, and D. Har, "MAC achieving low latency and energy efficiency in hierarchical M2M networks with clustered nodes," *IEEE Sensors J.*, vol. 15, no. 3, pp. 1657–1661, Mar. 2015.

[18] L. Liang, L. Xu, B. Cao, and Y. Jia, "A cluster-based congestion-mitigating access scheme for massive M2M communications in Internet of Things," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 2200–2211, Jun. 2018.

[19] M. Shirvanimoghaddam, M. Dohler, and S. J. Johnson, "Massive non-orthogonal multiple access for cellular IoT: Potentials and limitations," *IEEE Commun. Mag.*, vol. 55, no. 9, pp. 55–61, 2017.

[20] P. Si, J. Yang, S. Chen, and H. Xi, "Adaptive massive access management for QoS guarantees in M2M communications," *IEEE Trans. Veh. Technol.*, vol. 64, no. 7, pp. 3152–3166, Jul. 2015.

[21] M. Nakip, B. C. Gül, V. Rodoplu, and C. Güzeliş, "Comparative study of forecasting schemes for IoT device traffic in machine-to-machine communication," in *Proc. 4th Int. Conf. Cloud Comput. Internet Things (CCIOT)*, 2019, pp. 102–109.

[22] M. Nakip, V. Rodoplu, C. Güzeliş, and D. T. Eliiyi, "Joint forecasting-scheduling for the Internet of Things," in *Proc. IEEE Global Conf. Internet Things (GCIoT)*, Dec. 2019, pp. 1–7.

[23] V. Rodoplu, M. Nakip, D. T. Eliiyi, and C. Guzelis, "A multiscale algorithm for joint forecasting–scheduling to solve the massive access problem of IoT," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8572–8589, Sep. 2020.

[24] H. Farag, M. Gidlund, and P. Österberg, "A delay-bounded MAC protocol for mission- and time-critical applications in industrial wireless sensor networks," *IEEE Sensors J.*, vol. 18, no. 6, pp. 2607–2616, Mar. 2018.

[25] A.-T.-H. Bui, C. T. Nguyen, T. C. Thang, and A. T. Pham, "Design and performance analysis of a novel distributed queue access protocol for cellular-based massive M2M communications," *IEEE Access*, vol. 6, pp. 3008–3019, 2018.

[26] M. E. Tanab and W. Hamouda, "A scalable overload control algorithm for massive access in machine-to-machine networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.

[27] J. Mišić, V. B. Mišić, and N. Khan, "Sharing it my way: Efficient M2M access in LTE/LTE-A networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 1, pp. 696–709, Jan. 2017.

[28] J. S. Kim, S. Lee, and M. Y. Chung, "Efficient random-access scheme for massive connectivity in 3GPP low-cost machine-type communications," *IEEE Trans. Veh. Technol.*, vol. 66, no. 7, pp. 6280–6290, Jul. 2017.

[29] C.-Y. Oh, D. Hwang, and T.-J. Lee, "Joint access control and resource allocation for concurrent and massive access of M2M devices," *IEEE Trans. Wireless Commun.*, vol. 14, no. 8, pp. 4182–4192, Aug. 2015.

[30] J. Guo, X. Zhang, R. Wang, and X. Tao, "Joint access control and resource allocation for machine type communications," in *Proc. IEEE/CIC Int. Conf. Commun. China (ICCC Workshops)*, Oct. 2017, pp. 1–6.

[31] M. Centenaro, L. Vangelista, S. Saur, A. Weber, and V. Braun, "Comparison of collision-free and contention-based radio access protocols for the Internet of Things," *IEEE Trans. Commun.*, vol. 65, no. 9, pp. 3832–3846, Sep. 2017.

[32] V. Petkov and K. Obraczka, "Collision-free medium access based on traffic forecasting," in *Proc. IEEE Int. Symp. World Wireless, Mobile Multimedia Netw. (WoWMoM)*, Jun. 2012, pp. 1–9.

[33] V. Petkov and K. Obraczka, "The case for using traffic forecasting in schedule-based channel access," in *Proc. IEEE Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2011, pp. 208–212.

[34] Y. Edalat, J.-S. Ahn, and K. Obraczka, "Smart experts for network state estimation," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 3, pp. 622–635, Sep. 2016.

[35] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, "A comprehensive survey on machine learning for networking: Evolution, applications and research opportunities," *J. Internet Services Appl.*, vol. 9, no. 1, p. 16, Dec. 2018.

[36] A. Eswaradass, X.-H. Sun, and M. Wu, "Network bandwidth predictor (NBP): A system for online network performance forecasting," in *Proc. 6th IEEE Int. Symp. Cluster Comput. Grid (CCGRID)*, vol. 1, May 2006, p. 4.

[37] P. Cortez, M. Rio, M. Rocha, and P. Sousa, "Internet traffic forecasting using neural networks," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Jul. 2006, pp. 2635–2642.

[38] P. Bermolen and D. Rossi, "Support vector regression for link load prediction," *Comput. Netw.*, vol. 53, no. 2, pp. 191–201, Feb. 2009.

[39] S. Chabaa, A. Zeroual, and J. Antari, "Identification and prediction of Internet traffic using artificial neural networks," *J. Intell. Learn. Syst. Appl.*, vol. 2, no. 3, p. 147, 2010.

[40] Y. Zhu, G. Zhang, and J. Qiu, "Network traffic prediction based on particle swarm BP neural network," *J. Netw.*, vol. 8, no. 11, pp. 2685–2691, Oct. 2013.

[41] Y. Li, H. Liu, W. Yang, D. Hu, and W. Xu, "Inter-data-center network traffic prediction with elephant flows," in *Proc. IEEE/IFIP Netw. Operations Manage. Symp. (NOMS)*, Apr. 2016, pp. 206–213.

[42] Z. Chen, J. Wen, and Y. Geng, "Predicting future traffic using hidden Markov models," in *Proc. IEEE 24th Int. Conf. Netw. Protocols (ICNP)*, Nov. 2016, pp. 1–6.

[43] P. Poupart, Z. Chen, P. Jaini, F. Fung, H. Susanto, Y. Geng, L. Chen, K. Chen, and H. Jin, "Online flow size prediction for improved network routing," in *Proc. IEEE 24th Int. Conf. Netw. Protocols (ICNP)*, Nov. 2016, pp. 1–6.

[44] S. Chinchali, P. Hu, T. Chu, M. Sharma, M. Bansal, R. Misra, M. Pavone, and S. Katti, "Cellular network traffic scheduling with deep reinforcement learning," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 766–774.

[45] S. K. Sharma and X. Wang, "Distributed caching enabled peak traffic reduction in ultra-dense IoT networks," *IEEE Commun. Lett.*, vol. 22, no. 6, pp. 1252–1255, Jun. 2018.

[46] A. P. Silva, K. Obraczka, S. Burleigh, and C. M. Hirata, "Smart congestion control for delay- and disruption-tolerant networks," in *Proc. 13th Annu. IEEE Int. Conf. Sens., Commun., Netw. (SECON)*, Jun. 2016, pp. 1–9.

[47] J. Huang, C.-C. Xing, S. Y. Shin, F. Hou, and C.-H. Hsu, "Optimizing M2M communications and quality of services in the IoT for sustainable smart cities," *IEEE Trans. Sustain. Comput.*, vol. 3, no. 1, pp. 4–15, Jan. 2018.

[48] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *Proc. 15th ACM Workshop Hot Topics Netw. (HotNets)*, 2016, pp. 50–56.

[49] T.-H. Chen, J.-W. Chang, and H.-Y. Wei, "Dynamic inter-channel resource allocation for massive M2M control signaling storm mitigation," in *Proc. IEEE 84th Veh. Technol. Conf. (VTC-Fall)*, Sep. 2016, pp. 1–5.

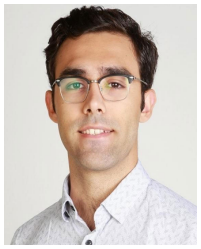
[50] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge, U.K.: Cambridge Univ. Press, 2005.

[51] P. Vamvakas, E. E. Tsiropoulou, and S. Papavassiliou, "On controlling spectrum fragility via resource pricing in 5G wireless networks," *IEEE Netw. Lett.*, vol. 1, no. 3, pp. 111–115, Sep. 2019.

[52] P. Vamvakas, E. E. Tsiropoulou, and S. Papavassiliou, "Dynamic spectrum management in 5G wireless networks: A real-life modeling approach," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2019, pp. 2134–2142.

[53] T. Park, G. Lee, and W. Saad, "Message-aware uplink transmit power level partitioning for non-orthogonal multiple access (NOMA)," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–6.
[54] M. S. Elbamby, M. Bennis, W. Saad, M. Debbah, and M. Latva-Aho, "Resource optimization and power allocation in in-band full duplex-enabled non-orthogonal multiple access networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 12, pp. 2860–2873, Dec. 2017.
[55] N. Gans, H. Shen, Y.-P. Zhou, N. Korolev, A. McCord, and H. Ristock, "Parametric forecasting and stochastic programming models for call-center workforce scheduling," *Manuf. Service Oper. Manage.*, vol. 17, no. 4, pp. 571–588, Oct. 2015.
[56] R. R. Appino, J. A. G. Ordiano, R. Mikut, T. Faulwasser, and V. Hagenmeyer, "On the use of probabilistic forecasts in scheduling of renewable energy sources coupled to storages," *Appl. Energy*, vol. 210, pp. 1207–1218, Jan. 2018.
[57] C. Singhal and S. De, *Resource Allocation in Next-Generation Broadband Wireless Access Networks*. Hershey, PA, USA: IGI Global, 2017.
[58] W. Feng, J. Wang, Y. Chen, X. Wang, N. Ge, and J. Lu, "UAV-aided MIMO communications for 5G Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1731–1740, Apr. 2019.
[59] C. Liu, W. Feng, Y. Chen, C.-X. Wang, and N. Ge, "Cell-free satellite-UAV networks for 6G wide-area Internet of Things," *IEEE J. Sel. Areas Commun.*, early access, Aug. 24, 2020, doi: 10.1109/JSAC.2020.3018837.
[60] E. Kocan, B. Domazetovic, and M. Pejanovic-Djurisic, "Range extension in IEEE 802.11ah systems through relaying," *Wireless Pers. Commun.*, vol. 97, no. 2, pp. 1889–1910, Nov. 2017.

VOLKAN RODOPLU (Member, IEEE) received the B.S. degree *(summa cum laude)* in electrical engineering from Princeton University, in 1996, and the M.S. and Ph.D. degrees in electrical engineering from Stanford University, in 1998 and 2003, respectively. He worked with the Wireless Research Division of Texas Instruments, Dallas, TX, USA, and with Tensilica Inc., Santa Clara, CA, USA. He served as an Assistant Professor of electrical engineering at the University of California, Santa Barbara, from 2003 to 2009, where he was promoted to the position of an Associate Professor with tenure. He is currently an Associate Professor of electrical engineering and a Marie Skłodowska-Curie Fellow of the European Commission at Yaşar University, İzmir, Turkey. His publications have received over 1000 citations in the Web of Science Citation Index. His current research interests include the Internet of Things, predictive networks, smart cities, and visible light communication. He is the winner of the National Science Foundation CAREER Award, USA, the University of California Regents' Junior Faculty Fellowship, Stanford Department of Electrical Engineering Outstanding Service Award, the Stanford Graduate Fellowship (as the Andreas Bechtolsheim Fellow), the Stanford Department of Electrical Engineering Fellowship, the John W. Tukey Award from the American Statistical Association, G. David Forney Award, and the George B. Wood Legacy Prize.



MERT NAKIP received the B.Sc. (Hons.) and M.Sc. degrees in electrical-electronics engineering from Yaşar University, İzmir, Turkey, in 2018 and 2020, respectively. He is currently pursuing the Ph.D. degree with the Institute of Theoretical and Applied Informatics, Polish Academy of Sciences, Gliwice, Poland. His design of a multi-sensor fire detector via machine learning methods was ranked #1 nationally at the Industry-Focused Undergraduate Graduation Projects Competition organized by TÜBİTAK (Turkish Scientific and Technological Research Council). His thesis focused on the application of machine learning methods to the IoT and was supported by the National Graduate Scholarship Program of TÜBİTAK 2210C in High-Priority Technological Areas. He is currently a Research Assistant with the Institute of Theoretical and Applied Informatics, Polish Academy of Sciences.

ROOZBEH QORBANIAN received the B.Sc. degree in industrial engineering from the Isfahan University of Technology, Isfahan, Iran, in 2013, and the master's degree in industrial engineering from the University of Tehran, Tehran, Iran, in 2015. He is currently pursuing the Ph.D. degree with the Luxembourg Centre for Logistics and Supply Chain Management, University of Luxembourg. He was ranked in the top 1% in the university entrance exams in Iran for both the bachelor's and master's degrees. He received a merit-based fellowship for his undergraduate education. He worked as a Research Scholar in the Scientific Research Project "Scheduling Algorithms for Wireless Communication" with Yaşar University, İzmir, Turkey. His Ph.D. thesis is supported by FNR (Luxembourg National Research Fund) under the BRIDGES program for an academic-industry partnership. His current research interests include scheduling, decision making under uncertainty, and machine learning.

DENIZ TÜRSEL ELIIYI received the B.S., M.S., and Ph.D. degrees in industrial engineering from Middle East Technical University, Ankara, Turkey, in 1998, 2001, and 2004, respectively. She worked with the Middle East Technical University, Dokuz Eylül University, the Izmir University of Economics, and Yaşar University, where she was a Professor and the Chair of the Department of Industrial Engineering. She was a Visiting Researcher with the Department of Business Information Technology, Virginia Polytechnic Institute and State University, USA, in 2013 and 2017, and with the Department of Computer Science, University of St. Andrew, U.K., in 2018. She is currently a Professor of industrial engineering and the Dean of the School of Engineering, Izmir Bakırçay University, İzmir, Turkey. She has authored over 100 journal and conference papers, and participated in over 15 scientific and industrial research projects. She received an Achievement in Education Award from Yaşar University in 2017.