Throughput Maximization in Mobile WSN Scheduling With Power Control and Rate Selection

Yosef Alayev, Fangfei Chen, Yun Hou, Matthew P. Johnson, Amotz Bar-Noy, *Member, IEEE*, Thomas F. La Porta, *Fellow, IEEE*, and Kin K. Leung

***Abstract*—We study a data dissemination scenario in which data items are to be transmitted to mobile clients via one of the stationary data access points (APs) that the clients pass by en route to their destinations. The scheduler dedicates sequences of consecutive timeslots of an AP to downloading a data item to a client during the time window in which it is in range, which corresponds to assigning a job (the client's download) to a machine (the AP) among many. The transmission rate chosen for each assignment partly corresponds to setting a machine's speed, but it also has subtler effects. The APs may control transmission power to tune its transmission range making sure that no interference occurs with neighboring APs' transmissions. The problem is a generalization of an already NP-hard parallel-machine scheduling problem in which jobs' release times and deadlines depend on the machine to which they are assigned. We define this joint timeslot, power control, and rate assignment problem formally and apply both new algorithms and adaptations of existing algorithms to it. We evaluate these algorithms through simulations which show that our proposed algorithms achieve near-optimal throughput.**

***Index Terms*—Interference, machine, mobility, networks, optimization, power control, resource allocation, scheduling, sensor, transmission control, wireless.**

I. Introduction and Motivation

In wireless sensor networks (WSNs), sensor nodes collect data of interesting events across the network and send them back to the data access points (APs), which are often stationary sensor nodes, awaiting the end users to collect the information on demand. It is often assumed in the literature that end users have immediate and unlimited access to APs via wired connections. However, if an end user is moving such that the wanted data needs to be **wirelessly** downloaded from an AP only when the user passes by, then the collection of data is subject to constrained contact windows in time. Furthermore, when there are more than one end users in the network to collect data from the given set of APs, they compete for the limited APs and constrained contact windows. In this case, how to assign the multiple APs to the mobile end users in time forms a job-machine scheduling problem with n jobs (each with weight w_i and processing time p_i, and release time r_i and deadline d_i) to be assigned to m parallel machines [1]. A valid assignment of a job i to machine k would be to dedicate machine k exclusively to job i over some interval $[s, s+p_i) \subseteq [r_i, d_i)$.

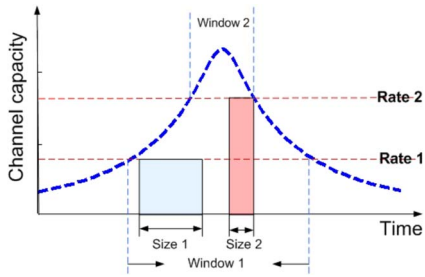

Fig. 1. Time window with various transmission rates and fixed power.

Systems for scenarios, like a rescue mission, where mobile end users need to download data items wirelessly from APs, introduce another degree of freedom to the scheduling problem, i.e., adaptive transmission rate selection. First assume APs can transmit using a constant transmission power or different channels that prevent any interference among neighboring APs. For a user-AP pair, the choice of transmission rate can affect both the contact window as well as the processing time, which in turn influences the scheduling performance. The reason is elaborated as follows.

According to the Shannon Theorem [2], (i.e., formula: $C = B \cdot \log(1 + (P/N/d^{\gamma}))$, where C—channel capacity, B—bandwidth of the channel, P/N—signal power to noise ratio, d—transmission range, γ—some attenuation constant parameter, $\gamma = 2$ for free space attenuation model), as a user passes by an AP, the capacity of the channel from the AP to the user first increases, as the user approaches the AP, and then decreases, as the user departs the AP, as shown in Fig. 1. The transmission rate of the download to the user is bounded above by the channel capacity of the AP. As a result, for a user-AP pair, choosing a lower transmission rate (i.e., Rate 1 in Fig. 1) gives a larger contact window (i.e., Window 1 in the Figure) with an AP as is seen by the flat Rate 1 line intersecting the capacity curve. Thus, lower transmission rates allow the download to start earlier and end later. For the higher rate (i.e., Rate 2 in Fig. 1) the contact window (i.e., Window 2) is shorter which means the download should start later and should terminate earlier. Intuitively, the lower the transmission rate the larger the contact

Manuscript received July 18, 2013; revised January 23, 2014; accepted March 17, 2014. Date of publication April 2, 2014; date of current version July 8, 2014. This work was supported by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The associate editor coordinating the review of this paper and approving it for publication was M. Rossi.

Y. Alayev, M. P. Johnson, and A. Bar-Noy are with the Department of Computer Science, The Graduate Center, The City University of New York, New York, NY 10016 USA.

F. Chen and T. F. La Porta are with the Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA 16802 USA.

Y. Hou and K. K. Leung are with the Department of Electrical and Electronic Engineering and Computing, Imperial College London, London SW7 2AZ, U.K.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TWC.2014.2315196

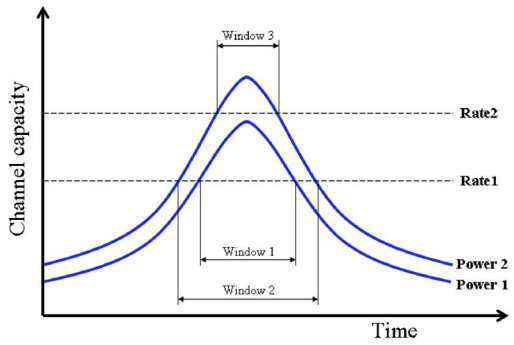

Fig. 2. Time window with various transmission rates and two different power levels.

window size giving more freedom when the transmission can be started and finished. However, the other impact of rate control is on the download duration of job, i.e., the job size. Since we can transmit the job faster with high transmission rate the size of the job is shortened. On the other hand, with lower transmission rate the size of the job is longer. Thus, lower transmission rate requires more slots on an AP.

Therefore, selecting the transmission rate has a two-fold impact to the job-machine scheduling problem. Selecting a low transmission rate increases the job's contact window with an AP but at the same time increases the size of the job (i.e., its download duration), while selecting a high transmission rate decreases the job's contact window with an AP but at the same time decreases the size of the job. How to adaptively control the transmission rate to optimize the matching/scheduling between end users and APs is still an open issue, as existing rate control for WSNs mostly focus on resolving network congestions for data transmission from the source sensors to the APs [3] and [4].

Controlling the transmission power adds another degree of freedom to our scheduling problem. When the power is increased the channel capacity curve would shift upward as is shown in Fig. 2. On assumption that the transmission rate is fixed, using the formula $C = B \cdot \log(1 + (P/N/d^2))$, we observe that increasing transmission power allows us to transmit farther by increasing APs transmission range. Thus, for a job-AP pair increasing power (i.e., from Power 1 to Power 2 in the Fig. 2) increases the contact window (i.e., from Window 1 to Window 2 in the Fig. 2), which means that the transmission can be started earlier and finished later. Even though power control with fixed transmission rate has no effect on the job's size, since job's size by definition depends on the size of the data and transmission rate, which intuitively means that we want to always use highest transmission power, it may create interference among other APs that are transmitting as is shown in Fig. 3. In the Figure we see that using Rate 2 and Power 2, Window 2C and Window 2D overlap (i.e., interference between two neighboring APs). However, when power is reduced to Power 1, Window 1C and Window 1D do not overlap. We also see from the Figure that using lower transmission rate Rate 1 creates interference even for lower power Power 1. Thus, a balancing factor that prevents us from using the highest transmission power is interference.

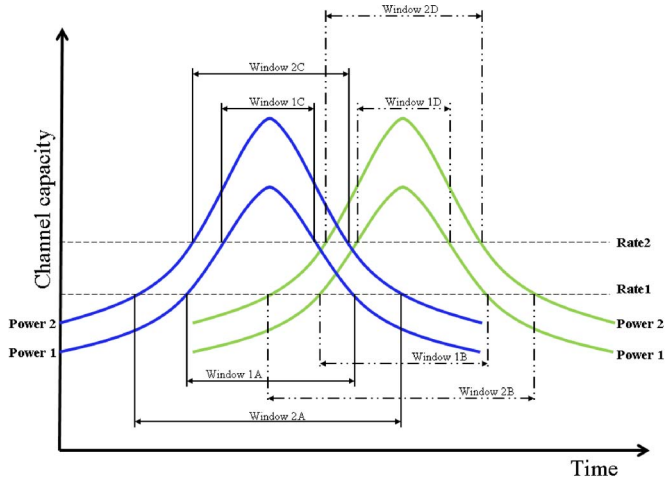

Fig. 3. Time window with various transmission rates and two different power levels and two interfering machines.

To sum up, in the scheduling problem that we study in this paper both contact windows and job sizes depend on jobs, machines to which they are being assigned to, transmission power levels of APs, and transmission data rates at which data are being transmitted on a following machine to a following job. Both transmission rate and transmission power can be controlled. The goal is to schedule job transmission on APs so as to maximize the sum of profits of all scheduled jobs (i.e., throughput maximization) while controlling transmission rate and transmission power per each job-AP pair and at the same time eliminate interference among transmitting APs (e.g., only one AP can transmit to avoid interference while the neighboring APs need to reduce their transmission range, so as not to interfere, by controlling their transmission power and/or transmission rate).

The remainder of this paper is organized as follows. Section II discusses some related work. Section III-A presents the system model. Section III-B and C present mathematical formulations of the problem and problem settings. Scheduling algorithms with rate selection and power control are presented in Section IV, followed by simulation and results in Section V. Section VI concludes the paper.

II. Related Work

The scheduling problem here lies within the family of parallel-machine scheduling. The literature on scheduling algorithms on parallel machines is enormous [1]. The most abstract problem is the *Interval Scheduling Problem* or ISP which is formulated as follows. For $\forall i \in \{1, \ldots, n\}$, given a family of intervals J_i. Selecting an interval [s, e) from J_i yields a profit of w_i. The task is to select at most one interval from each J_i so that the selected intervals are disjoint and the profit is maximized. This is the simplest model which is NP-hard [5]. The intervals may be listed explicitly or implicitly by some parameters defining a job J_i. A popular special case of ISP is where the intervals are defined by release time r_i, a deadline d_i, and a processing time p_i. To schedule job i, an interval of length p_i must be selected within the interval $[r_i, d_i)$. If we let the unit penalty U_i to be either 0 (job i is scheduled) or 1 (job i is not scheduled), then, in the standard 3-field notation for scheduling problems introduced by Lawler [6], this special

ISP is equivalent to $1|r_i|\sum_i w_i(1-U_i)$. Note that the first field in the standard notation specifies the machine environment (a single machine in our case), the second field specifies job characteristics (jobs having release times in our case), and the third field specifies optimality criteria (throughput in our case). This problem is known to be NP-hard since a special case is a Knapsack problem when all deadlines are equal and all release times are 0. In fact this problem is NP-hard in the strong sense if different integer job lengths are allowed when all release times and deadlines are integers employing a simple reduction from 3-PARTITION [7] and [8]. To generalize ISP to multiple-machine case where the machines are unrelated the problem becomes $R|r_i|\sum_i w_i(1-U_i)$, with R representing unrelated parallel machines. Due to applications that these special cases of ISP solve, they are often referred to as *throughput maximization problem (TMP)* or *real time scheduling problem* [5] and [9]–[12]. Many generalizations of this problem are NP-hard [8] when number of machines $m>1$: $r_i=0$ and identical d_i; three integer job lengths (1, 3, and q), integer deadlines but one overall release time; two integer job lengths (1 and q), integer release times and deadlines.

Many recent works on TMP provide approximation bounds for a more general setting of $R|r_i|\sum w_i(1-U_i)$ problem. Bar-Noy *et al.* in [11] give a 2-approximation for the $1|r_i|\sum w_i(1-U_i)$ and 3-approximation for the general case $R|r_i|\sum w_i(1-U_i)$ via an LP relaxation of a time-indexed formulation and rounding. They also provide a combinatorial algorithm m-Admission which has approximation bounds of $3+2\sqrt{2}$ for the unrelated machines case. Berman *et al.* in [10] give a combinatorial 2-approximation two-phase algorithm for $R|r_i|\sum w_i(1-U_i)$. Comparable results of 2-approximation are given by Bar-Noy *et al.* in [13] by employing a technique based on local-ratio which is comparable to primal-dual technique analysis. Chuzhoy *et al.* in [5] improve the approximation bound of 2 to less than 1.582 for arbitrary instances of ISP.

Many of these TMP algorithms considered machine independent contact windows [10], [11], and [13]. Recently, however, [12] has considered the TMP problem applied to mobile scenarios, where a mobile user can download from an AP only when it passes by within the AP's transmission range with machine-dependent contact windows. The problem is a generalization of TMP with job-dependant but machine-independent release times and deadlines. New algorithms with approximation guarantees are presented and evaluated. Lee *et al.* [14] study an unrelated machine scheduling where contact windows are both machine and job dependent. Their objective though is minimizing the total weighted flow time.

With an advance of wireless technology, wireless APs are capable of adjusting transmit power and data transmission rate with which an AP can communicate with the users [15] and [16]. Thus in wireless mobile applications there are other parameters that may specify/modify intervals and processing times for the TMP. The job scheduling problem relevant to adaptive rate-controlled scheduling for multimedia and other applications [17] and [18], is one in which each job $J_i=(w_i, r_i, d_i, p_{i,k})$ is instead characterized as $J_i=(w_i, r_i, \alpha_{i,k}, p_{i,k})$, where $\alpha_{i,k}=(d_i-r_i)/p_{i,k}$ is a stretch factor for J_i on machine M_k. Berman *et al.* in [19] presented a $2/(1+1/(2^{\lfloor\alpha\rfloor+1}-2-\lfloor\alpha\rfloor))$-approximation algorithm for this special case of TMP when the stretch factor α_i for each job J_i is at most α, which is a better than 2-approximation algorithm previously known. Though, the concept of a stretch factor is related to transmission rates, they are very different. In our application both jobs' processing times and contact windows depend on a transmission rate. Our multi-choice scheduling is related to a multiple-choice knapsack problem [20] in a sense where hoices are rates that determine both contact window size of a job (i.e., $[r_i, d_i)$ intervals) and processing times p_i that are also machine-dependent.

The choice of power level determines the contact window size and hence performance of the schedule. Yang *et al.* in [16] consider a problem of throughput maximization in a wireless mesh access network where operating frequency and power levels can be adjusted. The problem is approached from a game theoretical perspective. In their work the goal is to maximize the SINR and hence the throughput of both cooperative and non-cooperative APs while eliminating the interference. Peng *et al.* in [15] propose a recursive randomized algorithm to find optimal power levels and data rates for APs that would maximize the throughput. In both works, however, there is no scheduling involved since the objective is to transmit no matter to whom and at what time. As long as an AP can transmit some data with good SINR it contributes to the throughput. Our goal is to pick appropriate power level and data transmission rates for an AP for each job so as to eliminate interference with other APs as well as to maximize the schedule profit measured in sum of the weighted throughput of all scheduled jobs.

There are two models for the interference: physical and protocol. The *physical model* (e.g., SINR model) is widely considered as a reference model for physical layer behavior. However, its application in wireless sensor networks is limited due to its complexity. The *protocol model* (e.g., unified disk graph model) is simple. This is the model we use in our paper to create the interference matrix. Shi *et al.* in [21] reconcile the tension between physical and protocol models and explore the fundamental question on how to correctly use protocol interference model so as to narrow the solution gap between the physical and protocol models.

III. Problem Models

In this section we provide a formal problem definition and define an Integer Program (IP) to solve the problem. Since the problem is NP-hard, we then propose heuristic based algorithms with approximation guarantees to solve the problem. For the list of notations used in defining the models refer to Table I.

A. Abstract Job-Scheduling Model

We consider $\mathcal{M}=\{M_1,\ldots,M_m\}$ machines deployed in a given field and $\mathcal{J}=\{J_1,\ldots,J_n\}$ mobile users traveling in the field. Each user has a single job. Each job j is associated with a profit w_j, which indicates the value of successfully scheduling that job. (We assume without loss of generality that $w_j>0$ since zero-profit jobs can be discarded.) The optimization objective is to maximize the total profits of scheduled jobs. A user can be scheduled to download its job from any, but only one machine. Assume that both transmission data rates and transmission

TABLE I
NOTATION

$\mathcal{M}=\{M_1,...,M_m\}$	machines/APs
$\mathcal{J}=\{J_1,...,J_n\}$	jobs/mobile users ("cars")
$\mathcal{R}=\{R_1,...,R_K\}$	communication rate levels
$\mathcal{P}=\{P_1,...,P_q\}$	power levels
i,j	job/car index
k,l	machine/AP index
ρ,ρ'	communicate rate variables
π,π'	power level variables
s	job start time variable
w_j	J_j's weight value (profit)
$p_{jk\rho}$	processing time of J_j when run on (i.e., downloading from) M_k using rate ρ
$[s,e)=[s,s+p_{jk\rho})$	half-open time interval from s to $s+p_{jk\rho}$, i.e., a "job instance"
$r_{jk\rho\pi}$ and $d_{jk\rho\pi}$	release time and deadline for J_j on M_k using ρ and π, i.e., the times when J_j enters and leaves M_k's communication range
v	marginal value of a job instance, relative to current stack state
$\mathcal{I}_{k,\ell}(\pi,\pi',\rho,\rho')$	1/0 interference matrix between M_k using power π and rate ρ and M_ℓ using π' and ρ'
$\mathcal{I}$	the maximum # mutually non-interfering machines that may simultaneously interfere with a single other machine
$x_{jk\rho\pi s}$	1/0 decision variable indicating that we schedule job instance $[s,s+p_{jk\rho})$ of J_j on M_k with rate ρ and power π

power levels are finitely discretized. A transmission rate out of $\mathcal{R}=\{R_1,\ldots,R_K\}$ predefined rate levels needs to be adopted for the download. Let $\mathcal{P}=\{P_1,\ldots,P_q\}$ be the set of discrete power levels which machines can select when transmitting data to each job. Then scheduling a given job i means choosing 1) the machine k to assign it to, 2) the rate ρ at which to run it, 3) the power level π to use, and 4) the start time s at which to begin the job. Equivalently, each tuple $(j,k,\rho,\pi,s)\in\{jobs\}\times\{machines\}\times\{rates\}\times\{power\ \ levels\}\times\{times\}$ is one possible way to schedule or *job instance* of a job, and this scheduling is either chosen (providing profit w_j) or not. The parameters ρ and/or π are omitted when the rate and/or the power level is fixed.

A job can be scheduled—that is, a job instance is valid—only if the job is performed between its *release time* and *deadline*. These values both depend on the job, the machine, the rate, and the power: $r_{jk\rho\pi}$ and $d_{jk\rho\pi}$, while the job's processing time or *length* $p_{jk\rho}$ depends on the job, the machine, and the rate. More formally, the time interval of the job instance must be contained within the release time / deadline interval: $[s,s+p_{jk\rho})\subseteq[r_{jk\rho\pi},d_{jk\rho\pi})$.

With each pair of machines we can associate a zero-one interference matrix for each selectable power level and each selectable transmission data rate as $\mathcal{I}_{(M_k,M_\ell)}[\mathcal{P}\times\mathcal{P}\times\mathcal{R}\times\mathcal{R}]$. The entry is zero if two transmitting machines with selected power levels and selected transmission rates do not interfere, and one otherwise. That is, $\mathcal{I}_{(M_1,M_2)}[P_1,P_2,R_1,R_2]=1$ means that M_1 selecting power P_1 and transmission rate R_1 would interfere with M_2 that selected power P_2 and transmission rate R_2.

Recall that the intended application of this abstract problem is the transfer of data between cars and nodes. In this interpretation, user j's data item is job j and node k is machine k. We define the contact window associated with a chosen rate and power level of a job-machine pair to be the period of time within which the Shannon capacity [2] between the machine and the user is higher than the chosen rate (see Fig. 1). Thus, release times and deadlines (i.e., the boundaries of the time interval in which car/job j can communicate with node/machine k) to download from the machines are job, machine, rate, and power dependent. The time it takes to download a job is the processing time that is also job, machine, and rate dependent. The processing times do not change for different power levels for a fixed transmission rate. The objective of the scheduling problem is then to find, for each job, a machine, a transmission rate, a transmission power and a set of consecutive timeslots (defined by a starting timeslot), so as to maximize the total scheduled job profit while at the same time making sure that the transmission of any job by one machine does not interfere with the transmission of jobs from any other machine when powers and rates are adjusted.

TABLE II
IP FORMULATION WITH ADJUSTABLE RATES AND POWERS

$$\max \sum_{j=1}^{n}\sum_{k=1}^{m}\sum_{\rho=1}^{K}\sum_{\pi=1}^{q}\sum_{s=r_{jk\rho\pi}}^{d_{jk\rho\pi}-p_{jk\rho}} w_j\cdot x_{jk\rho\pi s}$$

$$\text{s.t.} \sum_{j=1}^{n}\sum_{\rho=1}^{K}\sum_{\pi=1}^{q}\sum_{u=s-p_{jk\rho}+1}^{s} x_{jk\rho\pi u}\le 1 \qquad \forall\, k,s \qquad (1)$$

$$\sum_{k=1}^{m}\sum_{\rho=1}^{K}\sum_{\pi=1}^{q}\sum_{s=r_{jk\rho\pi}}^{d_{jk\rho\pi}-p_{jk\rho}} x_{jk\rho\pi s}\le 1 \qquad \forall\, j \qquad (2)$$

$$x_{jk\rho\pi s}+\sum_{u=s-p_{i\ell\rho'}+1}^{s} x_{i\ell\rho'\pi' u}+\mathcal{I}_{k,\ell}(\pi,\pi',\rho,\rho')\le 2 \qquad \forall\,(k\neq\ell),(i\neq j)\ \forall\,\pi,\pi',\rho,\rho',s \qquad (3)$$

$$x_{jk\rho\pi s}\in\{0,1\} \qquad (4)$$

B. IP Formulations

The problem can be formulated as an Integer Program (IP). Even though in general solving an IP is an NP-hard problem, for moderate input sizes of the problem instances an IP can give a solution in reasonable time. For larger instances LP-relaxation may provide useful bounds to compare heuristics.

We extend an IP formulation of [12] to one where multiple transmission rates and powers are possible. The core concept in the formulation is that of a *job instance*, as defined above. There is a 0/1 decision variable $x_{jk\rho\pi s}$ corresponding to each job instance $[s,s+p_{jk\rho})$ of J_j, when potentially on M_k, with transmission rate $\rho\in\mathcal{R}$ and using transmission power $\pi\in\mathcal{P}$, i.e., job instance $\{j,k,\rho,\pi,s\}$. The decision variable $x_{jk\rho\pi s}$ is 1 if this job instance is scheduled, and 0 otherwise. The IP formulation is shown in Table II.

The optimization objective here is to maximize the sum of weight of all scheduled jobs, over all possible legal job assignments $x_{jk\rho\pi s}$. The first constraint ensures mutual exclusion, that is, no multiple jobs are scheduled simultaneously on a single machine. The second constraint prevents any single job from being scheduled more than once. The third constraint ensures that there is no interference among transmitting APs.

TABLE III
IP FORMULATION WITH ADJUSTABLE POWERS

$$\max \sum_{j=1}^{n}\sum_{k=1}^{m}\sum_{\pi=1}^{q}\sum_{s=r_{jk\pi}}^{d_{jk\pi}-p_{jk}} w_j \cdot x_{jk\pi s}$$

$$\text{s.t.} \sum_{j=1}^{n}\sum_{\pi=1}^{q}\sum_{u=s-p_{jk}+1}^{s} x_{jk\pi u} \le 1 \qquad \forall\, k, s \qquad (5)$$

$$\sum_{k=1}^{m}\sum_{\pi=1}^{q}\sum_{s=r_{jk\pi}}^{d_{jk\pi}-p_{jk}} x_{jk\pi s} \le 1 \qquad \forall\, j \qquad (6)$$

$$x_{jk\pi s} + \sum_{u=s-p_{i\ell}+1}^{s} x_{i\ell\pi' u} + \mathcal{I}_{k,\ell}(\pi,\pi') \le 2 \qquad \forall\,(k\neq\ell),(i\neq j),\ \forall\,\pi,\pi',s \qquad (7)$$

$$x_{jk\pi s} \in \{0,1\} \qquad (8)$$

TABLE IV
IP FORMULATION WITH ADJUSTABLE RATES

$$\max \sum_{j=1}^{n}\sum_{k=1}^{m}\sum_{\rho=1}^{K}\sum_{s=r_{jk\rho}}^{d_{jk\rho}-p_{jk\rho}} w_j \cdot x_{jk\rho s}$$

$$\text{s.t.} \sum_{j=1}^{n}\sum_{\rho=1}^{K}\sum_{u=s-p_{jk\rho}+1}^{s} x_{jk\rho u} \le 1 \qquad \forall\, k, s \qquad (9)$$

$$\sum_{k=1}^{m}\sum_{\rho=1}^{K}\sum_{s=r_{jk\rho}}^{d_{jk\rho}-p_{jk\rho}} x_{jk\rho s} \le 1 \qquad \forall\, j \qquad (10)$$

$$x_{jk\rho s} \in \{0,1\} \qquad (11)$$

The fourth constraint restricts x variables to be 0/1 integer. For an LP-relaxation, we can relax this constraint to be any number in interval [0, 1].

We use indices $i, j \in \{1,\ldots,n\}$ for jobs, $k, \ell \in \{1,\ldots,m\}$ for machines, $\rho, \rho' \in \{1,\ldots,K\}$ for rates, $\pi, \pi' \in \{1,\ldots,q\}$ for power levels, and $s \in \{1,\ldots,t\}$ for timeslots. The pairs of ρ and π variables are used in the constraints ensure that no two job assignments interfere with one another. We can then express release time and deadline for job j on machine k with transmission rate ρ and power level π as $r_{jk\rho\pi}$ and $d_{jk\rho\pi}$. The processing time for job j, machine k, and transmission rate ρ is $p_{jk\rho}$. Let s indicate the starting time of job j on machine k with transmission rate ρ and transmission power π if this job assignment (job instance) is chosen. In such a case $r_{jk\rho\pi} \le s$ and $s + p_{jk\rho} \le d_{jk\rho\pi}$.

Controllable Power But Fixed Rates: To study how power control affects the scheduling performance independently from rate control, we can assume that there is only a single transmission rate that the APs may employ and model the problem for adjustable power levels only. In this case both release times and deadlines are job-machine-power dependant as $r_{jk\pi}$ and $d_{jk\pi}$, but the processing times are job-machine dependent only as in p_{jk}. The interference matrix then is modified as $\mathcal{I}_{(M_k,M_\ell)}[\mathcal{P}\times\mathcal{P}]$. The entry is zero, if two transmitting machines do not interfere at given transmitting power levels, and one otherwise. The x variables will drop ρ subscript since the rate is fixed. The IP can be modified by dropping the third summation from the objective and the second summation from both the first and second constraints. The formulation is given in Table III.

The constraints are as before. The first constraint ensures mutual exclusion, that is no multiple jobs are scheduled simultaneously on a single machine. The second constraint prevents any single job from being scheduled more than once. The third constraint ensures that there is no interference among transmitting APs. The fourth constraint restricts x variables to be 0/1 integer.

Controllable Rates But Fixed Powers: When APs are located in such a way that no interference is possible for the highest selected power level for each AP we can eliminate the interference constraint from the formulation altogether. The release times, deadlines, and processing times in this case become job-machine-rate dependent only and are expressed as $r_{jk\rho}, d_{jk\rho}, p_{jk\rho}$, respectively. The x variables will drop the π subscript since the powers are fixed. This can help us study how adjustable rates affect the scheduling performance. The IP formulation is modified by dropping the fourth summation from the objective and the third summation from both the first and second constraints. The formulation is given in Table IV.

The first set of constraints is for mutual exclusion which prevents multiple jobs from being scheduled on a single machine. The second set of constraint prevents each job from being scheduled more than once. The third constraint restricts decision variable to be integers 0 or 1.

We have implemented these IPs in AMPL and solved them with CPLEX. For large instances we can relax integrality constraints to solve an LP-relaxation of IPs.

C. Application Setting

In this section we describe our environment and discuss two different settings on which we base our solutions. In the system we have stationary Access Points deployed within a geographical region and mobile clients with information needs that are traveling towards a mission site. We call the time period during which a client can communicate with the AP the *contact window*. A client must receive its information from any one of the AP within this time window. The window is decided by the speed vector, the route, the relative location of the AP, the transmission rate and transmission power of the AP.

The system has slotted time, so the task of the system is to decide how we allocate these timeslots to different clients. There are two types of problem settings we consider: offline and centralized online. In the offline setting, everything required to solve the problem is known, i.e., the data that is requested, the path of each mobile client including their speed. This setting is relevant for the scheduling a planned rescue missions. All scheduling may be done before the missions start. In the centralized online, nothing about the job is known until its client appears on a geographical region. Once the client is in the region, everything about the job becomes known, provided that nothing unpredictable happens. For example data delivery to busses in the city, where busses may be added or removed from service, but their routes are known and fixed. A central system has information about the busses in service and schedules the delivery of data. In the next section we design algorithms for both settings and prove approximation bounds.

IV. ALGORITHMS

In this section we design algorithms for the TMP($\mathcal{J}, \mathcal{M}, \mathcal{R}, \mathcal{P}$) problem. First we prove the hardness of our scheduling problem. We prove hardness by means of a Cook reduction from the Knapsack, which is NP-hard [7].

Theorem 1: Solving TMP($\mathcal{J}, \mathcal{M}, \mathcal{R}, \mathcal{P}$) optimally is NP-hard.

Proof: Given an instance of a Knapsack with capacity B and knapsack items i where $i \in \{1, \ldots, n\} \subset \mathbb{N}$. Each item i has a profit w_i and a size p_i. We create an instance of TMP($\mathcal{J}, \mathcal{M}, \mathcal{R}, \mathcal{P}$) where there is only one machine, one transmission rate and one power level (i.e., TMP($\mathcal{J}, M_1, R_1, P_1$)). For each knapsack item i associate a J_i of TMP($\mathcal{J}, M_1, R_1, P_1$), where all J_i have release time equal 0 and deadline equal B. The processing time of J_i is equal to the size of the knapsack item i, which is p_i. The weight of J_i is equal to the profit of knapsack item i, which is w_i. An optimal solution to TMP($\mathcal{J}, M_1, R_1, P_1$) is an optimal solution to Knapsack. □

Since the general problem is NP-hard even in a restricted setting where there is only one choice for a transmission rate and transmission power, we propose heuristic based algorithms by extending existing combinatorial algorithms which in some cases preserve the approximation guarantees. We adapt Two-Phase algorithm of [10] and Admission algorithm of [11] to design our algorithm to solve machine-job-rate and machine-job-rate-power dependent scheduling problems. We note that both of these algorithms were adopted in machine-job dependent scheduling windows settings in [12] without losing approximation guarantees.

A. Admission-Based Algorithms

In an Admission algorithm [11], jobs are considered in the order of non-decreasing end times. The algorithm schedules jobs machine-by-machine m times, and hence an algorithm is called m-Admission. The approximation ratio of this algorithm is $3 + 2\sqrt{2}$.

First we design algorithms that assume that transmission powers are fixed and no interference occur when adapting different transmission rates.

m-Admission—All Rates Algorithm: We propose an "m-Admission—all rates" algorithm. (Refer to Algorithm 1), which is a straightforward extension of the "m-Admission" algorithm in [12] where all possible rate levels are considered. The algorithm proceeds as follows:

- For each job-AP combination (j, k), choose a rate ρ. Find out the contact window size $T_\rho = (d_{jk\rho} - r_{jk\rho})$ timeslots and job size $p_{jk\rho}$ timeslots associated with the chosen rate ρ. Then, enumerate $N_{jk\rho} = T_{jk\rho} - p_{jk\rho} + 1$ job instances, each with incremental starting timeslot s where $s \in \{r_{jk\rho}, \ldots, d_{jk\rho} - p_{jk\rho}\}$.
- Perform step 1 with all combinations job-machine-rate triplets (j, k, ρ) until all job instances (j, k, ρ, s) are enumerated.
- Run m-Admission on all the job instances to obtain the final schedule.

It is worth noting that "m-Admission—all rates" provides the same approximation guarantee (i.e., $3 + 2\sqrt{2}$) as the m-Admission algorithm of [11], since all possible rate levels are considered here.

Algorithm 1 m-Admission Algorithms

procedure ENUMERATION STAGE
// m-Admission—all rates
Enumerate all job instances (j, k, ρ, s)
OR
// m-Admission—max-ratio rate selection
For each (j, k), pre-select ρ^* such that $\rho^*_{j,k} = \arg_\rho \max(d_{jk\rho} - r_{jk\rho}/p_{jk\rho})$
(i.e., Enumerate all job instances $(j, k, \rho^*_{j,k}, s)$)
end procedure
Input: Enumerated job instances
Output: Feasible schedule of jobs on m machines
procedure JOB SELECTION STAGE
  Let $S \leftarrow \varnothing$ be admission schedule
  for *each machine* k **do**
    $I \leftarrow$ the set of all job instances $(job, weight, beginning, ending)$
    Note: Consider the jobs that are not yet scheduled on previous machines.
    sort I in order of non-decreasing ending (conflicts are resolved by considering higher weighted instances first, then if the weights are the same the order is according to bigger beginning)
    Let $A \leftarrow \varnothing$ be schedule on machine k
    while $I \neq \varnothing$ **do**
      let $J_j \in I$ be a job instance that terminates earliest
      $I \leftarrow I \setminus \{J_j\}$
      if J_j *is not yet scheduled* **then**
        let C_j be the set of jobs in A overlapping with J_j
        let W be the total weight of C_j
        if $W = 0$ or $w_j > 2 \cdot W$ **then**
          $A \leftarrow A \cup \{J_j\} \setminus C_j$
        end if
      end if
    end while
    Append all jobs in A to S
  end for
  return S
end procedure

m-Admission—Max-Ratio Rate Selection Algorithm: In systems with large number of possible rate levels, the algorithm "m-Admission—all rates" that enumerates job instances with all possible rate levels could be too complex. One way to avoid this is to find an existing one rate for each job-AP pair that can be considered rather than all possible rates. Our heuristic selects a rate so that the ratio of contact window size to the job size is maximized. More formally:

- For each job-AP combination (j, k), choose a rate $\rho^*_{j,k}$ such that $\rho^*_{j,k} = \arg_\rho \max(d_{jk\rho} - r_{jk\rho}/p_{jk\rho})$.
- Each job-AP pair has a chosen rate $\rho^*_{j,k}$.

- Enumerate all job instances with the chosen rate with all different combination of $(j, k, \rho^*_{j,k}, s)$.
- Run m-Admission on the job instances to obtain the final schedule.

The algorithm will run faster on expense of losing the approximation guarantee.

Centralized-Online Algorithm: The m-Admission algorithm can be extended to the centralized online setting. Rather than applying algorithm machine-by-machine, we can extend it so that the algorithm works on machines in parallel where we schedule the earliest finishing job among all the machines in each step. The extended algorithm is called *Global Admission* (Refer to Algorithm 2). To implement our centralized algorithm, we apply Global-Admission algorithm iteratively (Refer to Algorithm 3).

Algorithm 2 Global-Admission Algorithm

procedure GLOBAL-ADMISSION
  Let $A \leftarrow \varnothing$ be global-admission schedule
  Let $I \leftarrow$ the set of all job instances
  while $I \neq \varnothing$ **do**
    let $J_j \in I$ be a job instance that terminates earliest
    $I \longleftarrow I \setminus \{J_j\}$
    if $J_j \notin A$ **then**
      let $C_j \leftarrow$ the set of jobs $\in A$ overlapping with J_j
      let $W \leftarrow$ the total weight of C_j
      if $W = 0$ or $w_j > 2 \cdot W$ **then**
        $A \longleftarrow A \cup \{J_j\} \setminus C_j$
      end if
    end if
  end while
  return A
end procedure

Algorithm 3 Centralized-Online Algorithm

procedure CENTRALIZED-ONLINE
  for each moment t and a new job J_j arrives:
    Fix all scheduled jobs J_i with starting time $s \leq t$
    Remove other jobs from the scheduled job list
    Call *Global-Admission* with all unscheduled jobs
  end for
end procedure

B. Two-Phase Based Algorithms

Just like with the m-Admission algorithm, we can adopt a Two-Phase algorithm of [10] which guarantees a slightly better performance. In the first phase the algorithm pushes job instances in order of non-decreasing end times onto a stack, assuming that these job instances have large enough weight compared to conflicting instances already on the stack. In the second phase, the algorithm pops the job instances from the stack to place them into a non-overlapping schedule. When a job instance enters the stack, it is pushed with a positive difference of its weight and the sum of all the weights of the overlapping instances on the stack. This in effect guarantees that the weight of the stack is equal to the weight of the schedule formed in the second phase. Just like with the m-Admission algorithm, we can enumerate all instances with all combinations of job-machine-rate triplets (j, k, ρ) until all job instances (j, k, ρ, s) are enumerated, where $s \in \{r_{jk\rho}, \ldots, d_{jk\rho} - p_{jk\rho}\}$. Then run the Two-Phase algorithm with all the instances. It is worth noting that the approximation ratio of 2 still holds in this case. We call this algorithm "Two-Phase—all rates". We can further extend the algorithm where the rates are pre-selected based on max-ratio of contact window and job size. Afterwards, rather than running an m-Admission, we run the Two-Phase algorithm. We call this algorithm "Two-Phase—max-ratio rate selection". Refer to the pseudo-code of the Two-Phase Algorithm in [10].

Two-Phase Algorithm with Controllable Power and Rate Levels Algorithm: The Two-Phase algorithm can also be extended for the case with adaptive power control and transmission rate. Both rate and power levels are controllable. We must adapt different power levels and transmission rates ensuring that there is no interference between transmitting APs. The extended algorithm is called *Two Phase Algorithm—Rate-Power-Control (2PA-RPC)* (Refer to Algorithm 4).

Algorithm 4 Two Phase Algorithm—Rate-Power-Control (2PA-RPC)

Let $tot(x.i)$ be the total *value* of job instances of i on the stack
Let $TOT(x)$ be the total *value* of instances of jobs *other than* $x.i$ on the stack, with ending $> x.s$ and interfering with $(x.k, x.\pi, x.\rho)$
for each job i **do** $done[i] \leftarrow false$;
end for
procedure PHASE ONE: EVALUATION
  $L \leftarrow$ the set of all job instances $x = (i, k, \pi, \rho, s, e, w)$
  sort L in order of nondecreasing ending (ties are resolved by considering higher weighted instances first, then if the weights are the same the order is according to bigger beginning, and by machine arbitrarily; the approximation guarantee does not depend on this tie-breaking method)
  $S \leftarrow$ an empty stack
  for *each* x *from* L
    $v \leftarrow x.w - tot(x.i) - TOT(x)$
    if $v > 0$ **then**
      push $((i, k, \pi, \rho, s, e, v), S)$
    end if
  end for
end procedure
procedure PHASE TWO: SCHEDULING
  for *each machine* k
    $occupied[k] \longleftarrow t$
  end for
  while $S \neq \varnothing$
    $(i, k, \pi, \rho, s, e, v) \leftarrow \text{pop}(S)$;

22: **if** $done[i] = false$ and $e \leq occupied[k]$ and $(i, k, \pi, \rho, s, e, v)$ does not interfere with any jobs already scheduled on other machines
23: add $(i, k, \pi, \rho, s, e, w)$ to solution
24: $done[i] \leftarrow true$
25: $occupied[k] \leftarrow s$
26: **end if**
27: **end while**
28: **end procedure**

Above, we defined a job instance as a tuple (i, k, π, ρ, s). For convenience, we will write two additional (computed) parameters within the job instance tuple. In addition to containing the start time s we add the end time $e = s + p_{j,k,\rho}$ (where $p_{j,k,\rho}$ is the processing time determined by the job, the machine, and the communication rate). We say job instance x *conflicts with* job y if $y.s < x.e \leq y.e$ (or vice versa), or if x and y are instances of the same job. (We use the dot notation $x.s$, $x.e$, etc., to indicate these values for a given job instance x.) We also add a parameter v representing the *value* of a *job instance* x relative to the current state of the stack *at the time when* x *is being considered*: the job instance's weight minus the values of the job instances on the stack that x conflicts with. Intuitively, the job instance's value represents the the marginal increase in total weight that this job instance would provide.

Algorithm 4 is the generalization of Berman's Two Phase Algorithm to our more general setting in which there are k (sometimes interfering) machines. This requires us to order all job instances, rather than proceeding machine-by-machine, and we must take into consideration interference between the machines. Our algorithm works as follows: in phase one, job instances (for all machines) are considered in order of nondecreasing ending and pushed onto the stack (in line 12 of the algorithm) if their (marginal) value is greater than that of the job instances there they would preclude scheduling. In phase two, instances are popped off of the stack and scheduled, with the instances they preclude popped off and thrown away. Each job instance added to the actual schedule (in line 23) contains a w rather than a v because w is the actual weight it provides—and no other job instance conflicting with it will be scheduled.

Two machines may interfere, depending on their locations, their power levels, and their transmission rates. That is, the entities that interfere with one another in this problem setting are pairs of (machine, power level, rate) triplets, e.g., $(x.k, x.\pi, x.\rho)$ interferes with $(y.k, y.\pi, y.\rho)$. (This generalizes a simpler, special case model on which pairs of machines interfere.) Two (machine, power, rate) tuples with the same machine in each pair *always* interfere.

A parameter appearing in the approximation guarantee is $\mathcal{I}$, which is the maximum number of mutually non-interfering machines that may simultaneously interfere with a single other machine. That is, imagine the interference graph between machines resulting from a power assignment to every speed. $\mathcal{I}+1$ is the size of the largest claw (i.e., $K_{1,\mathcal{I}}$) contained in any such interference graph (for a given problem instance). A value yielding a looser approximation guarantee is the largest degree of all such graphs. Of course, $\mathcal{I} = 0$ when there is no interference between machines.

Theorem 2: The approximation ratio of 2PA-RPC for problem TMP$(\mathcal{J}, \mathcal{M}, \mathcal{R}, \mathcal{P})$ is at most $2 + \mathcal{I}$.

Proof: We prove the result by extending the arguments of [10] from the single-machine case. Let $V(X)$ be the sum of values of intervals in the set of intervals X. The proofs of Lemmata 1 and 3 of [10], which prove, respectively that the algorithm produces a valid, conflict-free solution and that the solution value (i.e., the total weight of the jobs scheduled in Phase Two) is at least $V(S)$, go through essentially unchanged. Our argument here follows and generalizes the proof of [10]'s Lemma 2.

By S we will refer both to the stack and the set of entries on it at the end of Phase One, when the algorithm is run on some fixed problem instance. Let O be some optimal solution (of value OPT).

Let S_x be the set of all non-$x.i$ job instances on S conflicting with x, and let $S_{x.i}$ be the set of all $x.i$ job instances on S.

Consider a particular job instance $x = (i, k, \pi, \rho, s, e, v) \in O$. S_x contains the non-$x.i$ job instances on the stack that would have conflicted with x at the time it was considered and possibly pushed onto the stack. (Recall that job instances are considered in order of ending.) Now consider $\sum_{x \in O} V(S_x)$. Each time a job instance $x \in O$ is considered and some instance $y \in S$ conflicts with it, the value of y is added to this sum. How many times can an instance y be counted? That is, for a particular y on S, how many intervals $x \in O$ can conflict with it? In the single-machine setting of [10], the intervals $[s, e)$ appearing in O are disjoint, and so the intervals on S each intersect with at most one interval of O—the intervals of O *partition* (a subset of) the intervals on the stack, and the answer is 1. In our setting, however, things are more complicated.

Because all the intervals $[s, e)$ appearing in O *for a given machine* k are disjoint, y's interval will intersect at most one member of O per machine. Less loosely, since $x.s < y.e \leq x.e$ in such cases, y conflicts with each such instance x at time step $y.e$ and yet the instances x do *not* conflict with one another, and so the largest possible number of such instances x for machines *other than* $x.i$ is $\mathcal{I}$. Of course, y could conflict with a (lower power) $y \in O$ on the same machine as y as well.

Therefore

$$\sum_{x \in O} V(S_x) \leq (1 + \mathcal{I}) V(S). \tag{12}$$

Consider again job instances $x \in O$ and the corresponding $S_{x.i}$. Since these are disjoint subsets of S, we have

$$\sum_{x \in O} V(S_{x.i}) \leq V(S). \tag{13}$$

We now prove that for each $x \in O$, we have

$$w_i \leq V(S_x) + V(S_{x.i}). \tag{14}$$

Let $\hat{S}$ be the set of job instances on the stack when $x = (i, k, \pi, \rho, s, e, v)$ is considered in Phase One. At that point we decide whether to push to the stack based on the value

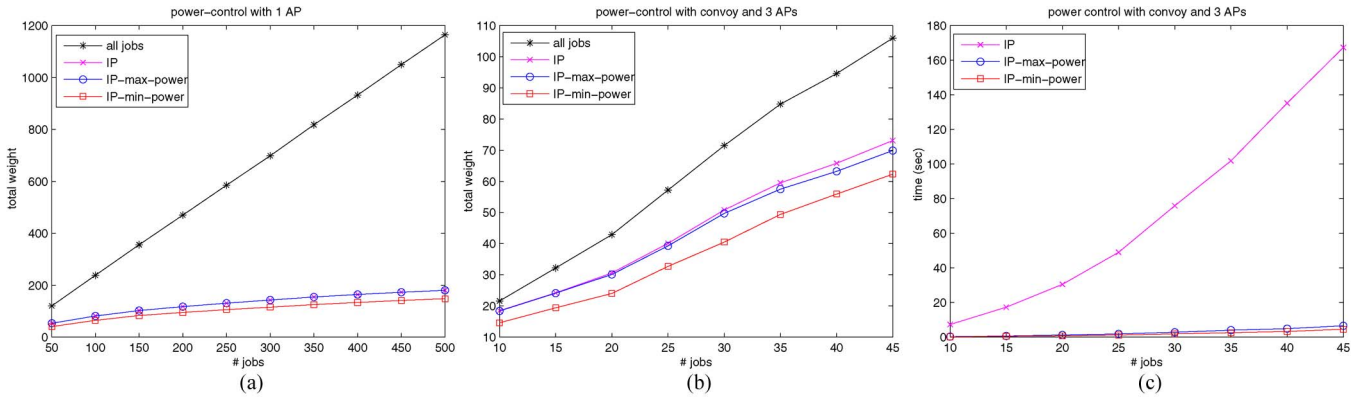

Fig. 4. Power-controlled experiments. (a) 1 AP. (b) 3 APs. (c) Runtime for 3 APs.

$v = w_i - TOT(x) - tot(i)$. By definition, we have $tot(i) = V(\hat{S}_{x.i})$ and $TOT(x) = V(\hat{S}_x)$. Therefore, if $v \leq 0$, we have

$$\begin{aligned} w_i &\leq tot(i) + TOT(x) \\ &= V(\hat{S}_{x.i}) + V(\hat{S}_x) \\ &\leq V(S_{x.i}) + V(S_x). \end{aligned}$$

On the other hand, if $v > 0$ then we push (increasing the stack value by v), and so we have

$$\begin{aligned} V(S_i) &\geq V(\hat{S}_{x.i}) + v \\ &= V(\hat{S}_{x.i}) + w_i - TOT(x) - tot(i) \\ &= V(\hat{S}_{x.i}) + w_i - V(\hat{S}_x) - V(\hat{S}_{x.i}) \\ &= w_i - V(\hat{S}_x) \\ &\geq w_i - V(S_x). \end{aligned}$$

Thus Ineq. (14) again follows.

Now we sum the LHS and the two terms of the RHS of Ineq. (14) over all $x \in O$ and apply Ineqs. (12) and (13)

$$\begin{aligned} \sum_x w_i &\leq \sum_x V(S_x) + \sum_x V(S_{x.i}) \\ OPT &\leq (1 + \mathcal{I})V(S) + V(S) \\ &\leq (2 + \mathcal{I})V(S). \end{aligned}$$

□

V. Simulation and Results

We conducted a series of experiments on simulated data, evaluating the performance of the integer programs and our algorithms on:

- the power-controlled problem formulated in Table III (see Fig. 4);
- the rate-controlled problem formulated in Table IV (see Fig. 5);
- the rate/power-controlled problem formulated in Table II (see Fig. 6).

In each experiment, we evaluate how either a) *performance* or b) *running time* changes as we vary a problem parameter: either the number of jobs (indexed by j in the IPs) or the number of machines (indexed by k). Recall from above that the performance measure we are optimizing for in our problem setting is the total weighted throughput of jobs. That is, we want to maximize the sum of the weights of all jobs that are successfully scheduled.

Our running time measures represent the times we empirically observed for the various algorithms to compute and return their solutions. Our code implementing the algorithms was written in Python and CPLEX was used to solve integer programs. Our experiments were performed on an HP Pavilion dv6t with a 2.2GHz Intel Core 2 Duo processor and 4 GB of RAM, running Windows 7.

In each experiment we average the results (i.e., performance or running time) over 50 randomly generated problem instances. In each such experiment, the same 50 instances are used for all algorithms evaluated.

Several AP deployment scenarios are considered (recall that APs are indexed by k in the IP):

- a single AP,
- multiple APs deployed on a straight line and a convoy of clients (e.g., cars) traveling by at varying speeds,
- multiple APs deployed in a grid formation.

The density of the APs network is controlled by varying the d which is defined to be the distance between horizontal or vertical lines of a grid with each AP located on an intersection of horizonal and vertical lines of a grid. To generate jobs, we use the Random Waypoint model described in [22] and used in [12]. In this model a job (car) selects a random destination in the simulated region and a random speed in the range of [5, 10] m/sec.

IP Constants and Experiment Parameters: We will now overview the values chosen in the experiments for the problem parameters (refer to Table V), which correspond to constants in the integer programs given above. The weight w_j of a job is randomly generated using a Zipf distribution (see [23] for an overview) with $\alpha = 2$, clipped with a minimum and maximum weights of 1 and 10, respectively. The algorithms can adaptively choose rates ρ from 1 to 11 Mb/s $(K = 11)$ in discrete steps of 1 Mb/s. The bandwidth of APs is set to 20 Mb/s. Each job has a *data size*, which is uniformly distributed in [1, 10]. These data sizes, together with transmission rates, determine the processing times.

The power levels π (measured in mW) can be controlled and adaptively chosen from a set $\mathcal{P} = \{100, 125, 150, 175, 200\}$. For each job-AP pair the contact windows (which are related

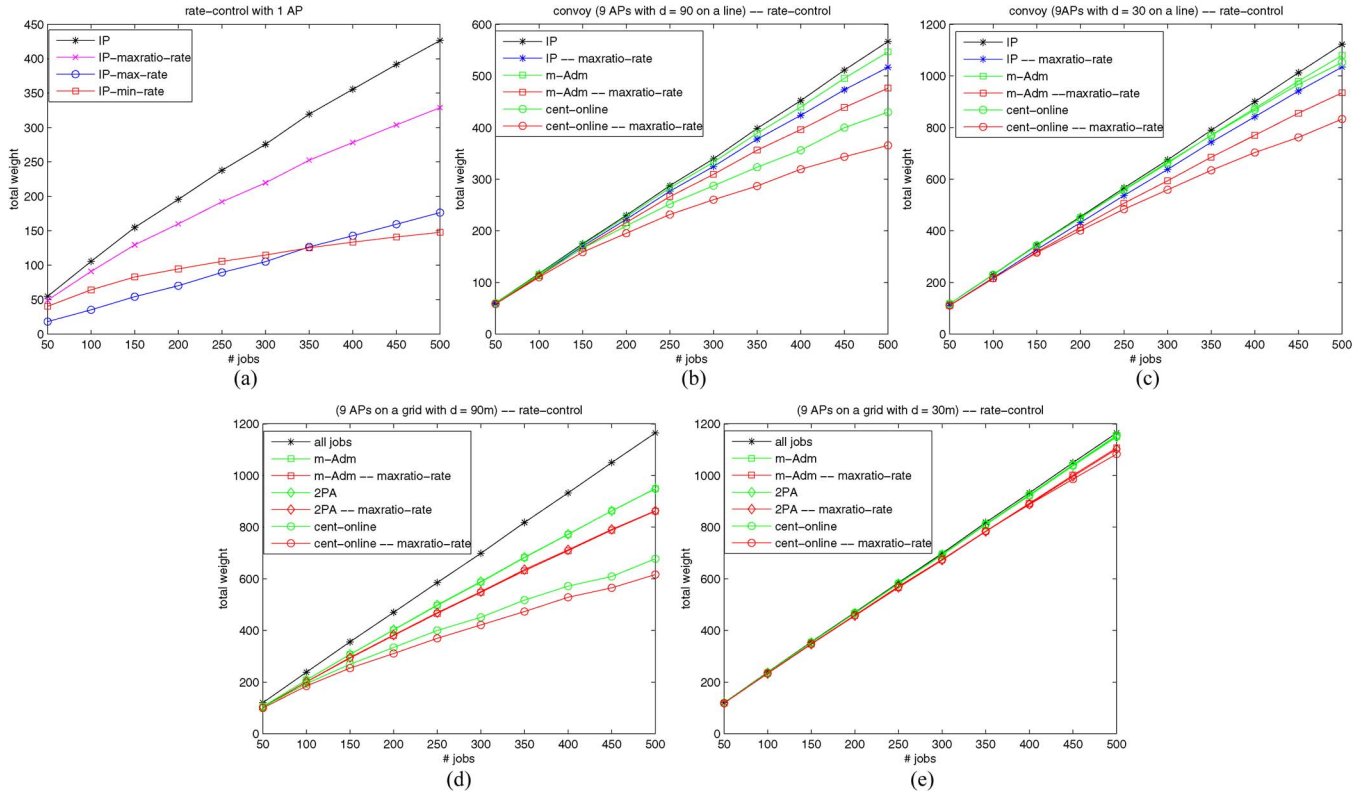

Fig. 5. Rate-controlled experiments. (a) 1 AP. (b) Convoy with separation 90. (c) Convoy with separation 30. (d) Grid with separation 90. (e) Grid with separation 30.

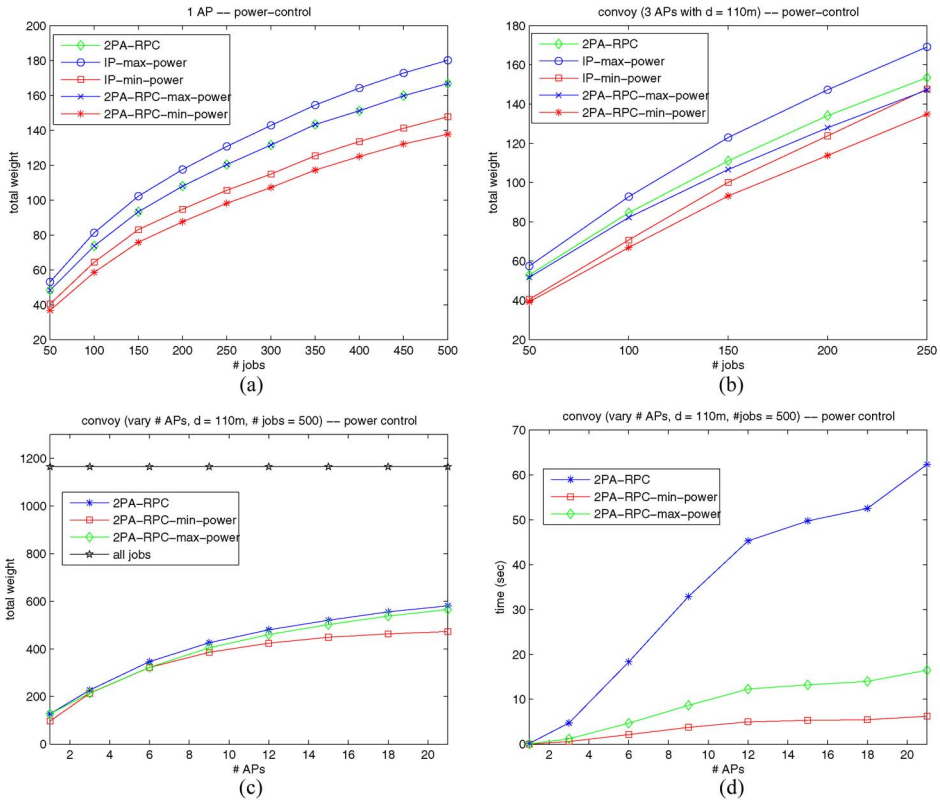

Fig. 6. Rate/power-controlled experiments. (a) 1 AP. (b) Convoy. (c) Varying # APs. (d) Varying # APs: time.

TABLE V
SIMULATION PARAMETERS

machines	access points (APs)
jobs	mobile users ("cars")
release time on M_k	time of entry into AP k's range
deadline time on M_k	time of exit from AP k's range
$x_{jk\pi\rho s} = 1$	car j downloads from AP k with power π and rate ρ starting at time s
number of APs $m \in \{1, 3, 9\}$ or varying	
number of jobs $n = 500$ or varying	
weights w_j chosen from a Zipf distribution [23]	
power levels $\pi_j \in \{100, 125, 150, 175, 200\}$, in mW	
communication rates $\rho_j \in \{1, 2, ...11\}$, in Mbps	
AP separation distance $d \in \{30, 90, 110\}$, in m	

to transmission ranges) are calculated for different transmission rates and different transmission powers using formula

$$C = B \cdot \log\left(1 + \frac{P/(1\text{ mW})}{d^2}\right).$$

The interference matrix $\mathcal{I}_{(M_k,M_\ell)}[\mathcal{P} \times \mathcal{P} \times \mathcal{R} \times \mathcal{R}]$ is determined according to the protocol interference model. The entry is zero if the disks corresponding to transmission ranges of APs with radii given by the above formula, do not overlap for given transmission power levels and transmission data rates. The entry is one otherwise.

We calculate the contact windows $[r_{jk\rho\pi}, d_{jk\rho\pi})$ as follows. For each AP we use the above formula to compute the transmission ranges yielded by different power levels and transmission rates. Given a car's speed, we can compute when the car enters and leaves the circle representing the AP's transmission range. We can thus compute the release times $r_{jk\rho\pi}$ and deadline $d_{jk\rho\pi}$ of a job j, for each (machine k, power level π, transmission rate ρ) triple. We round up to nearest integer the release times and round down to the nearest integer the deadlines. The processing time $p_{jk\rho}$ of each job instance is computed by dividing the data size of job j by the transmission rate ρ that AP k uses to transmit that data. We then round up to the nearest integer the processing time. Then for job j and for each (k, π, ρ) triple, this yields job instances for all start times $s \in \{r_{jk\rho\pi}, \ldots, d_{jk\rho\pi} - p_{jk\rho}\}$.

Power-Controlled Experiments: In the first experiment [see Fig. 4(a)] we use one AP and vary the number of jobs. The power levels can be controlled and are chosen from a set $\mathcal{P} = \{100, 125, 150, 175, 200\}$. The transmission rate is fixed at 1 Mb/s. We solve the IP with all possible power levels, with only maximum power level of 200 mW, and with only minimum power level of 100 mW. We also performed an experiment [see Fig. 4(b) and (c)] for a scenario with three APs deployed on a line, separated by a distance of 110m, and a convoy of cars traveling past them at different speeds. As in the previous experiment, power levels can be controlled while the transmission rate is fixed at 1 Mb/s. The AP separation distance $d = 110$ m so that neighboring APs do not interfere when transmitting at the lowest power but *do* interfere otherwise.

Rate-Controlled Experiments: In the next experiment we investigate effects of rate control. First we performed an experiment [see Fig. 5(a)] with one AP with a fixed transmission power but adjustable rates in Mb/s chosen from the set $\mathcal{R} = \{1, 2, \ldots 11\}$. We ran IP with all rate levels, with maximum rate level, with minimum rate level, and with rate level for each job-AP pair selected in such a way so that the ratio of contact window size over job size is maximized.

We conducted experiments to test the performance of our algorithms for both offline and online settings. We performed experiments with a convoy passing through 9 APs on a line with $d = 90$ m and $d = 30$ m [see Fig. 5(b) and (c), respectively], and with 9 APs placed on a grid of 3 rows and 3 columns with distance of separation of rows and columns $d = 90$ m and $d = 30$ m [see Fig. 5(d) and (e), respectively]. The client paths are generated by the Random Waypoint model. The algorithms evaluated are m-Admission, two-phase, centralized-online, with a) all rates and b) the preselected best rate, based on the maximum ratio of contact window size to job size.

Rate/Power-Controlled Experiments: Finally, we conducted experiments to test the 2PA-RPC algorithm for the case of adjustable rates/powers. We tested under two scenarios: one AP with a fixed transmission rate but adjustable powers in mW chosen from a set $\mathcal{P} = \{100, 125, 150, 175, 200\}$ [see Fig. 6(a)], and a convoy passing through 3 APs on a line [see Fig. 6(b)]. The AP separation distance was set to $d = 110$ m, so that, as before, neighboring APs do not interfere when transmitting at the lowest power but *do* interfere otherwise.

To study how the performance of 2PA-RPC algorithm scale as a function of the number of APs we performed an experiment with a convoy of 500 jobs passing through varying number of APs on a line with $d = 110$ m [see Fig. 6(c) and (d)]. Again a fixed transmission rate is used while power levels can be adjusted and chosen from a set $\mathcal{P} = \{100, 125, 150, 175, 200\}$.

The next section reports results and insights from simulation.

A. Results and Insights

Power-Controlled Experiments: The case of power control on one AP is depicted on Fig. 4(a). The curves for IP with all power levels and for IP with only maximum power level coincide and are higher than the curve for IP with only minimum power level. This clearly shows that, since there are no other APs to interfere with, the optimal solution with power control is to always select the highest possible power for scheduling jobs. This is not true in a case of 3 APs deployed on a line. Since APs may interfere with one another the maximum power is not always optimal anymore. This can be seen from Fig. 4(b) where the curve of IP with maximum power is lower than the curve of IP. Nonetheless, the curve of IP with maximum power is still higher than the curve of IP with minimum power. This shows that by increasing power we increase contact window size that may have sometimes more positive effect such as increased throughput than negative effect such as interference. Fig. 4(c) shows runtimes in seconds for running IPs with all power levels or just maximum or minimum power level.

Observe that running IP in this power-controlled setting becomes prohibitively expensive very quickly [see Fig. 4(c)], even with small number of jobs: the running time to solve the IP grows much faster than the times for the heuristic algorithms, because the number of job instances, and hence number of

IP variables, grows with number of jobs and also on number of APs, which in this case is 3. For IPs with only a single power level (i.e., max or min) the running time is far more tractable.

Rate-Controlled Experiments: The case of rate control on one AP is depicted on Fig. 5(a). From the Figure we see that unlike with power control, selecting maximum rate is inefficient. This is due to the double effect that the rate has on scheduling, i.e., increasing rate not only shrinks the job size but also shrinks the contact window size. With lower rate, the contact window size increases, but in expense of increased job size. In fact employing only one fixed rate, whether min or max, gives very poor performance as is seen in the Figure. Preselecting a rate where the ratio of contact window size to job size is maximum gives better throughput than using any single fixed rate for all jobs; however, it is still sub-optimal to a case where all rate levels are considered.

Performance of algorithms for a convoy passing through 9 APs on a line separated by a distance of $d = 90$ m and $d = 30$ m is depicted in Fig. 5(b) and (c), respectively. As seen from the Fig. 5(b), the solution based on m-Admission algorithm gives a near optimal throughput. Even m-Admission with preselected rates has better performance than centralized-online algorithm that operates on all possible rates. However, when the separation between the APs decreases to $d = 30$ m, as is seen from Fig. 5(c), the centralized online algorithm gives a much better throughput, even outperforming the IP solution with preselected rates.

The case of rate control with 9 APs on a grid is depicted on Fig. 5(d) and (e). On Fig. 5(d) we see that the lines for m-Admission and two-phase algorithms almost coincide. The centralized-online algorithms gives a slightly lower throughput. Performance of all three algorithms with preselected rates based on maximum ratio of contact window size to job size evidently give a lower throughput. However, we have observed that the runtime drops by an order of magnitude when best rates are preselected. The runtime is very crucial especially for the online setting when the schedule needs to be created online. When the APs are located closer on a grid all algorithms give near optimal solution as is seen from Fig. 5(e).

Rate/Power-Controlled Experiments: Performance of the 2PA-RPC algorithm for power/rate-controlled throughput is shown on Fig. 6(a) for one AP and Fig. 6(b) for multiple APs. In Fig. 6(a) we have IP solutions for both using maximum transmission power and minimum transmission power. For one AP it is always optimal to use the maximum transmission power and, thus, the IP solution using maximum transmission power is optimal even when using all possible power levels. From the Figure we see that the curve for 2PA-RPC algorithm is higher than IP-min-power curve, which means that 2PA-RPC algorithms gives better throughput than the best possible solution with only one lowest transmission power level. When running 2PA-RPC algorithm with only max power level it gives the same throughput as when running 2PA-RPC with all possible power levels. However, 2PA-RPC with min power level gives the lowest throughput, as expected. For the case of multiple APs using always the maximum power level is not always optimal but still better than using the minimum power level. In Fig. 6(b) we see that our 2PA-RPC still gives better solution than just using lowest power level. We also compare the throughput of 2PA-RPC algorithm with the ones when running 2PA-RPC using only max or only min power levels and observe that 2PA-RPC gives better throughput than if using just max or min power levels.

Fig. 6(c) and (d) depict the performance of the 2PA-RPC algorithm on a varying number of APs. From Fig. 6(d) we see that 2PA-RPC scales well with number of APs: the running time shown in the Figure appears to be nearly linear. In fact the complexity of the algorithm as implemented is determined by the time to sort all job instances: $O(N \log N)$ where N is the number of job instances. Note that N will grow linearly with the number of APs (with identical characteristics). With 500 jobs, 5 power levels, and up to 21 APs, the solving the integer programs is intractable almost immediately and so omitted.

VI. Conclusion and Future Work

In this paper, we have studied a variant of TMP problem with adaptive transmission power and rate control. We have formulated the problem for joint scheduling with either power control or rate control or both. We have adopted existing and proposed new algorithms with performance guarantees.

An interesting open problem is raised by our work. We have considered that when two APs transmit with such a power that creates an overlap between transmission circles, then APs interfere and cannot transmit at the same time. However, it is a liberal assumption since the jobs do not have to be within an overlap region. If the two jobs receiving transmission from two APs are outside the overlap region then such overlap may still be considered. The solution to such a problem should take into account not just the duration of contact windows but also point to point location of jobs within a region. In such a case the performance guarantee of the 2PA-RPC algorithm may be improved.

Acknowledgment

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence, or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

[1] P. Brucker, *Scheduling Algorithms*. Berlin, Germany: Springer-Verlag, 2007.

[2] C. Shannon, N. Petigara, and S. Seshasai, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, Jul. 1948.

[3] S. Rangwala, R. Gummadi, R. Govindan, and K. Psounis, "Interference-aware fair rate control in wireless sensor networks," in *Proce. ACM SIGCOMM*, 2006, pp. 63–74.

[4] J. Paek and R. Govindan, "RCRT: Rate-controlled reliable transport protocol for wireless sensor networks," *ACM Trans. Sensor Netw.*, vol. 7, no. 3, pp. 20:1–20:45, Oct. 2010. [Online]. Available: http://doi.acm.org/10.1145/1807048.1807049

[5] J. Chuzhoy, R. Ostrovsky, and Y. Rabani, "Approximation algorithms for the job interval selection problem and related scheduling problems," *Math. Oper. Res.*, vol. 31, no. 4, pp. 730–738, Nov. 2006. [Online]. Available: http://dl.acm.org/citation.cfm?id=1235273.1235278
[6] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. R. Kan, "Optimization and approximation in deterministic sequencing and scheduling: A survey," *Ann. Discr. Math.*, vol. 5, pp. 287–326, 1979.
[7] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Colmpleteness*. San Francisco, CA, USA: Freeman, 1979.
[8] B. B. Simons and M. K. Warmuth, "A fast algorithm for multiprocessor scheduling of unit-length jobs," *SIAM J. Comput.*, vol. 18, no. 4, pp. 690–710, Aug. 1989.
[9] N. G. Hall and M. J. Magazine, "Maximizing the value of a space mission," *Eur. J. Oper. Res.*, vol. 78, no. 2, pp. 224–241, Oct. 1994.
[10] P. Berman and B. DasGupta, "Multi-phase algorithms for throughput maximization for real-time scheduling," *J. Combinat. Optim.*, vol. 4, no. 3, pp. 307–323, Sep. 2000.
[11] A. Bar-Noy, S. Guha, J. S. Naor, and B. Schieber, "Approximating the throughput of multiple machines in real-time scheduling," *SIAM J. Comput.*, vol. 31, no. 2, pp. 331–352, 2001.
[12] F. Chen, M. P. Johnson, Y. Alayev, A. Bar-Noy, and T. F. L. Porta, "Who, when, where: Timeslot assignment to mobile clients," *IEEE Trans. Mobile Comput.*, vol. 11, no. 1, pp. 73–85, Jan. 2012.
[13] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor, and B. Schieber, "A unified approach to approximating resource allocation and scheduling," *J. ACM*, vol. 48, no. 5, pp. 1069–1090, Sep. 2001.
[14] Y. Lee and H. D. Sherali, "Unrelated machine scheduling with time-window and machine downtime constraints: An application to a naval battle-group problem," *Ann. Oper. Res.*, vol. 50, no. 1, pp. 339–364, 1994.
[15] C. Peng *et al.*, "Impact of power and rate selection on the throughput of *ad hoc* networks," in *Proc. IEEE ICC*, 2006, vol. 9, pp. 3898–3902.
[16] Y. Song, C. Zhang, and Y. Fang, "Throughput maximization in multi-channel wireless mesh access networks," in *Proc. IEEE ICNP*, 2007, pp. 11–20.
[17] D. K. Y. Yau and S. S. Lam, "Adaptive rate-controlled scheduling for multimedia applications," *IEEE/ACM Trans. Netw.*, vol. 5, no. 4, pp. 475–488, Aug. 1997.
[18] H. Liu and M. E. Zarki, "Adaptive source rate control for real-time wireless video transmission," *Mobile Netw. Appl.*, vol. 3, no. 1, pp. 49–60, Jun. 1998.
[19] P. Berman and B. Dasgupta, "Improvements in throughput maximization for real-time scheduling," in *Proc. 32nd Annu. ACM Symp. Theory Comput.*, 1999, pp. 680–687.
[20] E. Y.-H. Lin, "A bibliographical survey on some well-known non-standard knapsack problems," *INFOR*, vol. 36, no. 4, pp. 274–317, Nov. 1998.
[21] Y. Shi, T. Y. Hou, J. Liu, and S. Kompella, "How to correctly use the protocol interference model for multi-hop wireless networks," in *Proc. 10th ACM Int. Symp. MobiHoc Netw. Comput.*, 2009, pp. 239–248. [Online]. Available: http://doi.acm.org/10.1145/1530748.1530782
[22] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, "A performance comparison of multi-hop wireless *ad hoc* network routing protocols," in *Proc. 4th Annual ACM/IEEE Int. Conf. MobiCom Netw.*, 1998, pp. 85–97. [Online]. Available: http://doi.acm.org/10.1145/288235.288256
[23] M. Newman, "Power laws, pareto distributions and zipf's law," *Contemp. Phys.*, vol. 46, no. 5, pp. 323–351, Sep. 2005. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/00107510500052444

Yosef Alayev received the B.S. degree in computer science and B.A. degree in applied mathematics in 2002 from Queens College of the City University of New York, Flushing, NY, USA. He received the M.Phil. and Ph.D. degrees from the Department of Computer Science at the Graduate Center of the City University of New York, New York, NY, USA, in 2009 and 2014, respectively. His graduate research was in scheduling and resource allocation algorithms in wireless sensor network environment. His interests include combinatorial optimization, mathematical programming, and algorithm analysis and implementation.

Fangfei Chen received the B.S. degree in electronic engineering from Tsinghua University, Beijing, China, in 2006 and the Ph.D. degree in computer science from the Pennsylvania State University, University Park, PA, USA, in 2012. His Ph.D. research is in resource allocation problems in network systems. He is currently with Akamai Technologies.

Yun Hou was born in Kunming, China. She received the B.Eng. degree in telecommunications from Beijing University of Posts and Telecommunications, Beijing, China, in 2004, and the M.Sc. and Ph.D. degrees in electrical and electronic engineering from Imperial College London, London, U.K., in 2005 and 2009, respectively. In the summers of 2007 and 2008, he had internships at the Bell Labs, Alcatel-Lucent, UK, and the IBM T. J. Watson Research Center, NY, USA, respectively. After her Ph.D. graduation, she was with the Department of Electrical and Electronic Engineering, Imperial College London, as a Post-doctoral Research Associate. She then joined Shenzhen University, Shenzhen, China, as a Lecturer, in 2010, and since September 2011, she has been a Senior Engineer at the Communications Technologies Group of Hong Kong Applied Science and Technology Research Institute (ASTRI), Hong Kong. She is responsible for the research and development of the Digital Pre-Distortion (DPD) platform for GSM, WCDMA and LTE networks. In general, her research interests include resource allocation, scheduling and power control, network utility maximization, cloud computing and the design and development of embedded systems.

Matthew P. Johnson studied philosophy and computer science at Columbia and Lawrence Universities and received the Ph.D. degree in computer science from the City University of New York, New York, NY, USA, in 2010. He is currently an Assistant Professor in the Department of Mathematics and Computer Science at Lehman College, CUNY.

Amotz Bar-Noy received the B.Sc. degree in mathematics and computer science in 1981 and the Ph.D. degree in computer science in 1987, both from the Hebrew University, Jerusalem, Israel. From October 1987 to September 1989 he was a post-doc fellow in the Computer Science Department at Stanford University, Stanford, CA, USA. From October 1989 to August 1996 he was a Research Staff Member of the Communication and Networking Department at IBM T. J. Watson Research Center, New York, NY, USA. From February 1995 to September 2001 he was an Associate Professor in the Electrical Engineering—Systems Department of Tel Aviv University, Tel Aviv, Israel. From September 1999 to December 2001 he was a Principal Technical Staff Member of the Internet and Networking Systems Research Center at AT&T Research Labs, NJ. Since February 2002 he is a Professor in the Computer and Information Science Department at Brooklyn College and in the Computer Science Department at the Graduate Center of CUNY, New York, NY, USA.

Thomas F. La Porta (F'02) received the B.S.E.E. and M.S.E.E. degrees from The Cooper Union, New York, NY, USA, and the Ph.D. degree in electrical engineering from Columbia University, New York, NY, USA. He joined Penn State in 2002. He is the Director of the Institute of Networking and Security Research at Penn State. Prior to joining Penn State, he was with Bell Laboratories since 1986. He was the Director of the Mobile Networking Research Department in Bell Laboratories, Lucent Technologies where he led various projects in wireless and mobile networking. Currently, he is the William E. Leonhard Chair Professor in the Computer Science and Engineering Department at Pennsylvania State University, University Park, PA, USA. He is an IEEE Fellow, Bell Labs Fellow, received the Bell Labs Distinguished Technical Staff Award in 1996, and an Eta Kappa Nu Outstanding Young Electrical Engineer Award in 1996. He also won a Thomas Alva Edison Patent Awards in 2005 and 2009. Dr. La Porta was the founding Editor-in-Chief of the IEEE TRANSACTIONS ON MOBILE COMPUTING. He also served as Editor-in-Chief of IEEE Personal Communications Magazine. He was the Director of Magazines for the IEEE Communications Society and was on its Board of Governors for three years.

Kin K. Leung received the B.S. degree from The Chinese University of Hong Kong, Shatin, Hong Kong, in 1980, and the M.S. and Ph.D. degrees from University of California, Los Angeles, CA, USA, in 1982 and 1985, respectively.

He joined AT&T Bell Labs in Murray Hill, NJ, USA in 1986 and worked at its successor companies, AT&T Labs and Bell Labs of Lucent Technologies, until 2004. Since then, he has been the Tanaka Chair Professor in the Electrical and Electronic Engineering (EEE), and Computing Departments at Imperial College London in London, U.K. He serves as the Head of Communications and Signal Processing Group in the EEE Department at Imperial. His research interests focus on networking, protocols, optimization and modeling issues of wireless broadband, sensor and ad-hoc networks. He also works on multi-antenna and cross-layer designs for the physical layer of these networks.

Dr. Leung received the Distinguished Member of Technical Staff Award from AT&T Bell Labs in 1994, and was a co-recipient of the 1997 Lanchester Prize Honorable Mention Award. He was elected as an IEEE Fellow in 2001. He received the Royal Society Wolfson Research Merits Award from 2004 to 2009 and became a member of Academia Europaea in 2012. He has actively served on many conference committees. He serves as a member (2009–2011) and the chairman (2012–2015) of the IEEE Fellow Evaluation Committee for Communications Society. He was a guest editor for the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS (JSAC), IEEE Wireless Communications and the MONET journal, and as an editor for the JSAC: Wireless Series, TRANSACTIONS ON WIRELESS COMMUNICATIONS and IEEE TRANSACTIONS ON COMMUNICATIONS. Currently, he is an editor for the ACM Computing Survey and International Journal on Sensor Networks.