Alcom-FT Technical Report Series
ALCOMFT-TR-02-93

Parallel Scheduling Problems in Next Generation Wireless Networks

L. Becchetti*, S. Diggavi†‡, S. Leonardi*, A. Marchetti-Spaccamela*, S. Muthukrishnan‡, T. Nandagopal§, A. Vitaletti*

Abstract

Next generation 3G/4G wireless data networks allow multiple codes (or channels) to be allocated to a single user, where each code can support multiple data rates. Providing fine-grained QoS to users in such networks poses the two dimensional challenge of assigning *both* power (rate) and codes for every user. This gives rise to a new class of parallel scheduling problems. We abstract general downlink scheduling problems suitable for proposed next generation wireless data systems. This includes a communication-theoretic model for multirate wireless channels. In addition, while conventional focus has been on throughput maximization, we attempt to optimize the maximum response time of jobs, which is more suitable for stream of user requests. We present provable results on the algorithmic complexity of these scheduling problems. In particular, we are able to provide very simple, online algorithms for approximating the optimal maximum response time. This relies on resource augmented competitive analysis. We also perform an experimental study with realistic data of channel conditions and user requests to show that our algorithms are more accurate than our worst case analysis shows, and they provide fine-grained QoS to users effectively.

*Dipartimento di Informatica e Sistemistica Universita' di Roma "La Sapienza", Rome, Italy.

†Contact author, AT&T Shannon Labs, 180 Park Avenue, Bldg 103, Rm C287, Florham Park, NJ 07932, USA. Email: suhas@research.att.com., Tel: (973)-360-8492, Fax: (973)-360-8178.

‡AT&T Shannon Labs, Florham Park, New Jersey, USA.

§Department of Electrical and Computer Engineering, University of Illinois, Urbana-Champaign, Illinois, USA.

1 Introduction

There is tremendous momentum in the wireless industry towards next generation (3G and beyond)[1] systems. These systems will not only migrate the existing voice traffic to a higher bandwidth platform, but are also expected to jumpstart large scale data traffic. These emerging wireless systems[2] such as CDMA, wideband OFDM and multislot TDMA allow multiple codes (channels)[3] to be allocated to users, in each of which traffic can flow in one of multiple rates. This provides them more flexibility than is available in current systems to manage and modulate the traffic. This also gives rise to novel parallel scheduling problems that we study in this paper.

In a packet wireless cellular architecture each cell has a base station and is connected by a high-speed backbone to the Internet. Each base station handles all requests to and from mobile users within the cell, i.e., it handles both uplink (from mobile users) and downlink (to mobile users) requests.

Our focus in this paper is on the downlink channel performance, which is likely to be a major focus in emerging systems since data traffic is expected to dominate over time and data traffic typically tends to have asymmetrically large downlink demand. Existing wireline scheduling and resource allocation algorithms can not be directly applied to manage the downlink since wireless networks have unique characteristics, such as location dependent channel errors. Users in different regions of a cell experience different error rates, and hence they may get different data rates on the downlink. Unlike traditional scheduling scenarios, in a wireless environment, the scheduler must consider channel state in order to provide reasonable Quality of Service (QoS), and wireless systems have a variety of built-in capabilities to gather channel condition information for this purpose. In addition, random channel errors may result in poor performance of transport protocols such as TCP. Most solutions to this problem propose intercepting the connection at the base station, creating two logical connections [1]. The base station thus acts as a proxy, and interprets the packets up to the transport layer in order to address random channel errors. In many commercial wireless networks proxy servers store-and-forward data to mobile users, and thus have information regarding the various requests in the system, including user request sizes. We will assume this context and address the parallel scheduling problems that arise in this setting.

1.1 Contributions

Our contributions are threefold.

- We abstract a general downlink scheduling problem in next generation wireless data networks. This problem has many novelties. For example, we embody channel characteristics guided by communication theoretic considerations, and the properties of these channels get exploited in our scheduling algorithms. Second, we study QoS parameters related to per request behavior, in particular, we focus on optimizing response time per request. In contrast, prior work in wireless systems scheduling has typically focused on rate optimization (maximization) metrics.
- The scheduling problems that arise above are novel versions of multidimensional malleable task parallel scheduling. Here one can think of the number of channels as the number of processors and the malleability arises because the rates on each channel changes depending on power allocated. We show these problems to be NP-complete and that they are also hard to approximate. However, we use resource-augmented competitive analysis, to derive simple, online algorithms that provably have good performance in approximating the optimal maximum response time of a job. We also present results for other QoS metrics, such as average and weighted response times.
- We present a detailed experimental study of our algorithms. Using real web server request logs and realistic 3G/4G system parameters, we show experimentally that our online algorithms are practical, and perform significantly better than our worst-case analyses indicate. We also examine which resource is most important to augment in a wireless setting.

1.2 Related Work

There has been a significant amount of work on scheduling problems over wireless channels. We

[1] Current systems are 2G or 2.5G, where rudimentary data services are being deployed.

[2] CDMA: Code Division Multiple Access; OFDM: Orthogonal Frequency Division Multiplexing; TDMA: Time-Division Multiple Access. See Appendix J for more details.

[3] We use code and channel interchangeably since the terminology varies between CDMA, OFDM and TDMA systems

study the downlink scheduling problem. The uplink scheduling problem is a complementary problem where the fundamental issues are quite different. See [2] and references therein for more details.

Typically resource allocation problems study per-user rate throughput. The rate optimization problem has been extensively studied for various wireless system with focus varying from maximizing overall throughput to providing a minimum throughput guarantee for all users. A good discussion of related work about throughput optimization and fairness in wireless data networks can found in [3, 4].

Job scheduling is very popular in the context of processor scheduling, and various algorithms have been proposed for different QoS metrics such as completion time, maximum response time and, weighted response time [5]. In the parallel scheduling literature, there has been significant work on scheduling of *malleable* tasks, see for example [6, 7, 8, 9, 10]. Our multidimensional malleable scheduling problem has not been studied previously.

In wireless networks, job scheduling has been addressed in the context of downlink *broadcast* scheduling [11]. There a single transmission may satisfy multiple users or requests which is a model that is applicable in many special purpose systems. In contrast, we focus on unicast scheduling where each transmission is destined to a unique request which applies to general purpose wireless data networking. In a recent work, downlink unicast scheduling in CDMA systems was studied [12]; this is close to our work in spirit. However, they assume a linear rate model for the physical layer which is not accurate. Also, they do not have any upper bound on the number of available codes; hence, they study the problem of allocating power only. We have studied the nuances of allocating both power and codes, which is more suitable. Finally, we have provided a thorough competitive analysis of the online algorithms, in particular, using the resource augmented analysis; this is the first provable result known for any of the online scheduling problems, including the ones in [12].

Roadmap: We present the communication channel model, and abstract our scheduling problem in Section 2. In Section 3, we present a theoretical study showing the structure and the complexity of these problems. In Section 4, we present our main algorithmic results, namely, simple online algorithms and present our augmented-resource based analyses. In Section 5, we present our experimental results. We conclude the paper in Section 6. We have obtained a number of algorithmic results for optimizing other per request QoS metrics such as (weighted) average response times which we have included in the Appendix. Many of the other proofs are in the Appendix as well.

2 Problem Formulation

In this section, we will describe effects of the wireless communication medium on the transmission rates for each user, and also formulate parallel scheduling problems that arise in next generation wireless networks. (See Appendix J where we describe some details about next generation wireless proposals; these will fit our problem abstraction.)

2.1 Communication Channel Model

In wireless systems, channels have variable attenuation depending on the geographic location of the users. This is mainly due to multipath impairments and radio propagation losses. Say the base station (BS) is communicating with n mobile users. The physical channel attenuations of the users are denoted by $\bar{g}_1, \bar{g}_2, \cdots, \bar{g}_n$ respectively; each $\bar{g}_i$ is a scalar parameter called the *physical* gain. If the BS transmits power p_i to a user i, the signal-to-interference-plus-noise ratio (SINR) is given by $SINR = \frac{\bar{g}_i p_i}{\sigma^2}$, where σ^2 is the total noise power (including interference) [13]. SINR determines the rate of transmission of packets to the user. In particular, the rate $r_{bps}(\cdot)$ as a function of the SINR is a concave logarithmic function [14],

$$r_{bps}(x) = \bar{W} \log_2(1 + \frac{x}{\Gamma}) \tag{1}$$

where $r_{bps}(\cdot)$ is the rate in bits per second, $\bar{W}$ is the spectral bandwidth used, and Γ is dependent on the coding gain from the physical layer error-correcting code [14]. Both Γ and $\bar{W}$ are system parameters, which for our purposes will be constants. Therefore, for a particular user, the number of bits received over a period of time τ, obtained as a function of the power p_i allocated to the user on a single channel (code), is given by

$$r_i(p_i) = \bar{W}\tau \log_2(1 + \frac{\bar{g}_i p_i}{\sigma^2 \Gamma}) \tag{2}$$

The rate vs. SINR curves for next-generation wireless systems closely approximate the convex function described by this equation (see for example

[15]). This rate function already embodies the effect of variable rate error-correcting coding schemes in the physical layer, as is typical in next generation wireless systems [14, 15, 16]. Therefore, we will use this equation for rate calculations in our scheduling problems. For notational convenience we will denote $g_i = \frac{\bar{g}_i}{\Gamma\sigma^2}$ as the *channel gain* and we will set $W = \tau\bar{W}$ yielding $r_i(p_i) = W\log_2(1 + p_i g_i)$.

2.2 Abstract Scheduling Problem

The base station has a total power P to transmit. Time is assumed to be partitioned into equal width windows called *time slots*[4] (or frames), whose width is τ. We also assume that there are a total of C codes which can be assigned to users in a time slot. If user u makes a request i, then we say that the gain of request i is the same as the gain of the user u, $g_i = g_u$. Requests [5] arrive in the system over time at the beginning of time slots; requests/jobs are for non-real time traffic such as browsing, downloads, etc., i.e., file transfers. The size s_i (in bits) and the channel gain g_i of the user who made the ith request are known when the request arrives at time a_i. The arrival time is also known as *release time*. We will assume that the channel conditions of the users are constant over the scheduling period. Although this is a simplification, it holds in realistic cases[6]. The scheduling problem is to determine an assignment of power and codes to each user in each time slot, for the required QoS criterion. Formally, we have online job arrivals which specifies job size, and users time of arrival of each job, with the scheduling algorithm producing the following output.

Output: The set $\mathcal{C}_u(t)$, the set of codes assigned to user u at time t, and $p_u^{(i)}(t)$, the power assigned to user u at time t to each code $i \in \mathcal{C}_u(t)$. The assignment must satisfy the following conditions.

- *Total Code and Power Constraints.* Obviously, for any time slot t, $\sum_u |\mathcal{C}_u(t)| \le C$ and $\sum_u \sum_{i \in \mathcal{C}_u(t)} p_u^{(i)}(t) \le P$.

- *Discrete Rate Set* : Only a discrete set of rates (equivalently, minimum power per discrete rate) is allowed. These rates denoted $R(1), R(2), \ldots$ have the property[7] that $\frac{R(i)}{R(i-1)} \le 2$.

- *Request Completion*: All requests get the requested data size, that is, we need

$$s_u = \sum_t R_u(t) = \sum_t \sum_{i \in \mathcal{C}_u(t)} r_i(p_i) \qquad (3)$$

where $\mathcal{C}_u(t)$ is the set of codes assigned to user u in time slot t, and $r_i(p_i)$ is calculated using (2), subject to the discrete rate set constraint above.

Our goal is to *minimize the maximum response time* or *max-flow* [17], where response time for request i is $c_i - a_i$, if request i is completed by time c_i.[8] We will focus on this goal for most of the paper. We do prove results for certain related metrics such as minimizing total weighted response time, but those results appear in Appendix I.

We assume requests may be served over several time slots with different sets of codes at each time slot. In standard scheduling terminology [17], this corresponds to requests being *preempted* (*i.e.*, stop processing a request, process other requests, and resume the original request) or being *migrated* (*i.e.*, assign sets of codes to a user that differ from one time slot to another).[9]

There are two basic variants of our problems, namely *offline* or *online*. In the offline version, all request arrivals are known ahead of time. The offline case is of theoretical interest and is mainly useful to quantify the benefit to be accrued from scheduling. In the online case, requests arrive over time and the scheduling algorithms have to take their decisions without knowledge of future requests. The performance of the online algorithms are measured in comparison to the offline case as in standard competitive analysis.

Our scheduling problem is a malleable multidimensional scheduling problem. In general, a *malleable resource* is one in which giving more of it (say,

[4] We will use time and time slot interchangeably when no confusion arises.

[5] The terms: requests, jobs and users, will be used interchangeably.

[6] If time scale of the scheduler is several seconds, and if the users do not have very high mobility, then the channel conditions will be static over this time scale [13].

[7] This relationship holds for existing next generation wireless data system proposals like cdma2000 and HDR, which has a rate set of {38.4, 76.8, 102.6, 153.6, 204.8, 307.2, 614.4, 921.6, 1228.8, 1843.2, 2457.6} kilobits per second (kbps). The factor 2 is not sacrosanct. If the discrete rates are more spread out, but bounded by some constant, all our results will apply with minor changes in the claimed bounds.

[8] This is also sometimes called flow time in literature.

[9] A more detailed model may distinguish some codes to be more preferable than the others from one time slot to another to insure intercell interference avoidance, an issue we do not consider in this paper.

more processors) improves performance in the other (runs faster). Prior work involves malleable scheduling of parallelizable job [6, 8] or scheduling a mixture of malleable and non-malleable jobs [?, 7]. In [?], performance of heuristics which reduce a multiple malleable resource problem to a single malleable resource allocation problem, was analyzed. The differences in our problem from previous work are the following. In our case, the processing time of a job depends on two variables, the number of codes and the assigned power per slot; both these resources are malleable, and their combined effect on the total time to satisfy a request is provided by Equation 2. Moreover, the dependence amongst resources is not linear. We are also not aware of prior work on scheduling multiple malleable resources with preemption and migration as we do, that minimizes the maximum response time under Equation 2. Finally, to the best of our knowledge, the resource augmented competitive analysis for scheduling with multiple malleable resources has not been previously presented.

3 Understanding the Scheduling Problems

3.1 Some Structural Observations

Here, we state and prove some properties of the communication channel. They will be invoked later in proving our main results.

Continuous Power (Rate) Case: First, we consider case where the rates are not discrete and are a monotonic function of power, as in Equation (2). Concavity of the rate with respect to p in (2) implies that if we assign $c \geq 1$ codes to a user u, then *it is optimal to divide the total power p allocated to that user equally among the codes assigned* as summarized below.

Lemma 3.1 ***(Equipartition of power)*** *Given c codes and power p to a user, the rate r is maximized for $p_i = p/c,\ i = 1, \ldots, c$.*

Using Lemma 3.1 we can write the rate obtained by a user given c codes and p total power as,

$$R(p, c) = Wc \log(1 + \frac{gp}{c}). \qquad (4)$$

Discrete Power (Rate) Case: When we have a discrete rate set, the actual rate obtained on each code is given by the highest discrete rate[10] which is below $W \log(1 + \frac{gp}{c})$. Therefore, the difference between the continuous rate and the discrete rate case, depends on the discrete rate set available. Therefore, we can easily see the following fact.

Fact 3.1 *If the discrete rate set $\{R(j)\}$ is such that $\frac{R(j)}{R(j-1)} \leq 2$ then for any continuous power allocation p there exists a discrete rate $R(l)$ such that $R(l) \geq \frac{1}{2} r_i(p)$, where $r_i(p)$ is given by (2).*

3.2 Computational Hardness

In order to understand the challenge of the problem further, let us consider the offline complexity of the scheduling problems. If power (rate) values are required to be drawn from a discrete set, the problem in its simplest instance is the bin packing problem and hence it is NP-complete [18]. We focus on the more challenging case when code is discrete, as usual, but the power (and hence rate) is allowed to take any continuous value. In order to prove the hardness of this problem, we will consider the version of the problem in which ith request has arrival (release) time a_i, deadline d_i and size s_i in bytes. Using this, we can state the following theorem whose proof is given in the Appendix A.

Theorem 3.1 *If the number of codes assigned to each user is integral, then it is NP-complete to compute a feasible schedule for the problem of meeting deadlines even if all users have the same channel gain, a common release time, a common deadline and the power assigned to each code is not restricted to a discrete set of values.*

The result above in fact shows the problem to be NP-complete in the strong sense (see [18] for definition and significance). Although we have shown this hardness result only for the deadlines problem, it is easy to see that this immediately gives the hardness of other the scheduling problem we have, namely, minimizing the maximum response times. Finally, notice that the result holds independently of how rates are affected by the use of multiple codes, since the hardness is proved even for the restrictive case

[10] Note that we make a regularity assumption that the user gains are such that the lowest discrete rate $R(1) \leq W \log(1 + gP)$, *i.e.* by allocating all the resources to the user there exists a feasible discrete rate. In practice, error-correcting codes can be used over a group of codes to increase the dynamic range of user gains that fall into the feasible region.

when each request gets only one code over all time slots.

We can also show that this problem is hard to approximate. This is summarized in the following theorem whose proof is given in Appendix B. Therefore, in order to prove approximation results we require to use the techniques of resource augmentation as done in Section 4.

Theorem 3.2 *For every $c > 0$, it is NP-hard to compute a c-approximation of the maximum response time.*

3.3 Offline Scheduling Problem

We study the offline version *i.e.*, when all arrival times are known apriori. Using this, we will get lower bounds on optimum values of certain QoS metrics which will be a benchmark to compare against online algorithms.

Deadlines Scheduling Problem: Here, each request j has an arrival time a_j as well as a deadline d_j. As before, for each request j, at time a_j we know its size s_j and the channel gain g_j The goal is to merely test feasibility, *i.e.,* determine if there is a valid schedule that meets all deadlines. This problem is the technical core of many other scheduling problems.

We can write the solution to the deadlines scheduling problem as a combinatorial optimization program as shown in Table 1. Here, $c(j,t)$ denotes the number of codes assigned to user j in time slot t and let $p(j,t)$ be the total power assigned to user j in time slot t over all the codes. This is called as the *time indexed program* in the table.

Since $c()$ and $p()$ take on only discrete values, even to check feasibility is an NP-complete problem as proved earlier. Hence, we relax the variables to be continuous by allowing $c(j,t)$ and $p(j,t)$ to be fractional, resulting in the *Fractional Time Indexed Program.* The following is proved in Appendix C.

Theorem 3.3 *There exists a pseudo-polynomial time algorithm to solve the Fractional Time Indexed Program.*

Interval Indexed Program: The relaxation of the integer programming problem to the pseudo-polynomial time algorithm, still results in a problem size of $O(nT)$ variables, where n is the number of requests and T is the total length of the schedule. Next we consider decreasing the number of variables used in the convex program. We will define a new program below called the *interval indexed convex program.*

An *event* is either the arrival or the deadline of a request in the system. Consider the sorted list of the events $t_1, \dots, t_K$. We divide the time scale into *intervals* where an interval is the time period between any two consecutive events, that is interval I_k contains $[t_k, t_{k+1})$. For n requests, the total number of intervals is at most $2n$. We will look for *sliver* solutions, that is, ones in which for each interval I, each user j gets power $p(j,t)$ and $c(j,t)$ for $t \in I$ that remains constant for all $t \in I$, that is, $p(j,t_1) = p(j,t_2)$ for $t_1, t_2 \in I$ and likewise for $c()$. Let $c(j,k)$ be the fractional number of codes and $p(j,k)$ be the fractional power assigned to job j in interval k *per time slot.* Let a_j^{-1} denote the interval at the beginning of which job j arrives in the system, and d_j^{-1} be the interval at the end of which its deadline lies.

The convex programming formulation for solving the scheduling problem with slivers is given in the Table 1.

Theorem 3.4 *The time indexed convex program has a feasible solution if and only if the interval indexed convex program has a feasible solution. It can be solved in time polynomial in n, C using the convex program above.*

The proof is given in Appendix D. The result above exposes an interesting structural property of the interval indexed convex program, *i.e.*, the structure that sliver assignment of power and code to requests is optimal in the fractional case.

Using the Deadlines Scheduling Problem: Given that the feasibility of the (relaxed) time-indexed program can be solved efficiently, we can use it to solve the offline maximum response time problem nearly optimally. We do this by guessing a target response time F, and checking the feasibility of a deadline scheduling problem with deadlines $a_i + F$. By doing a binary search on the target response time value, we get an efficient (polynomial time) algorithm to optimize the maximum response time. Indeed the same approach works for optimizing quality of service criteria such as $\max_i f(c_i - a_i)$ for any monotonic increasing function f.

4 Online Heuristics

In this section we present our main algorithmic results, namely, a set of online algorithms for optimizing the metrics of our interest. As is standard, we measure the performance of an online algorithm using the ratio of the value of the objective function (here, *max-flow*) achieved by the online algorithm and the optimal value, which can be computed offline.

In our analysis we also use *resource augmentation* [19]. That is, we compare the optimum (i.e. the solution found by adversary) with the value of the solution found by the online algorithm when it is provided with more resources than an adversary who can serve it optimally. Formally, we say that algorithm A for our scheduling problem is an $(\alpha, \beta, \gamma, \delta)$ approximation if it provides a β approximation of the optimum when the sizes of user requests are scaled down by a factor α and the number of codes (the power) used by the algorithm is at most γ (δ, respectively) times the number of codes (the power, respectively) used by the optimum solution to serve the original input sequence.

Before proceeding with our results we provide a general rule that, given an $(\alpha, \beta, \gamma, \delta)$ approximation algorithm A, provides a $(1, \beta, \alpha\gamma, \alpha\delta)$ approximation algorithm A'. Algorithm A' is as follows: first apply A to the given instance and let S be the obtained solution that allocates power and codes to users. For each slot $x \in S$, A' uses α copies of x and allocates these slots in the same way as x. Clearly the set of new slots allows to answer α times the demand satisfied by x. Notice that A' uses α times more codes and total power than A. This implies that all results stated for algorithms working on requests of reduced size can be transformed into results for algorithms working on the original instance of the problem at the expense of some extra codes and power allocated by the system. Therefore, we can translate our results for reduced size requests to extra resource augmentation with no reduction in request sizes. This is the case with Theorem 4.3 and Theorem 4.5.

4.1 Minimizing the Maximum Response Time

We first develop our analysis for the "continuous" rate case (Theorem 4.2) and then analyze the case when only discrete rates per code is allowed (Theorem 4.4). It is well known in processor scheduling literature [17] that the online algorithm Earliest Release Time (ERT) or First In First Out (FIFO) is optimal for minimizing the maximum response time on a single machine and is a 3 approximation algorithm on parallel machines. Therefore, it is natural to ask how it would perform in our case. The simple FIFO strategy in our case allocates all the codes and power to one user at a time till the user completes the job. Using the "Equipartition of power" lemma (Lemma 3.1), this translates into giving the user a power per code of P/C in consecutive time slots that the user completely occupies. Therefore, we study the online scheme where each user is given a power P/C per code and is served by a FIFO scheduling discipline. We call this scheduling discipline FIFO($\frac{P}{C}$). Given an instance $\mathcal{I}$ of continuous job arrivals, we denote the maximum response time of the scheduling discipline FIFO($\frac{P}{C}$) by $\mathbf{f^{FIFO}}(\mathcal{I})$ and the optimal discipline has a maximum response time of $\mathbf{f^{OPT}}(\mathcal{I})$. We first we show a negative result (with proof in Appendix E), which demonstrates that such a strategy could be arbitrarily worse than the optimal strategy.

Theorem 4.1 *Given an instance $\mathcal{I}$ of continuous job arrivals,* $\max_{\mathcal{I}} \left(f^{FIFO}(\mathcal{I}) / f^{OPT}(\mathcal{I}) \right) > M$ *for any* M.

In spite of the negative results above, we can show that FIFO($\frac{P}{C}$) is able to achieve the optimum if every request is reduced to 50% of its original size. To do this we need the following lemma whose proof is given in Appendix F.

Lemma 4.1 *Let the optimal discipline on an online job arrivals instance $\mathcal{I}$ give a power assignment $p_j^{OPT}(\mathcal{I})$ and code assignment $k_j^{OPT}(\mathcal{I})$ to the jth job. Let us denote by $\mathcal{I}'$ the instance where each job size s_j in instance $\mathcal{I}$ is reduced to $s_j/2$, and denote by $k_j(\mathcal{I}')$ the code assignment to job j by the scheduling discipline FIFO($\frac{P}{C}$) applied to $\mathcal{I}'$. Then*

$$\frac{k_j(\mathcal{I}')}{C} \leq max \left\{ \frac{p_j^{OPT}(\mathcal{I})}{P}, \frac{k_j^{OPT}(\mathcal{I})}{C} \right\} \tag{5}$$

The algorithm schedules the jobs in order of release time, assigning a number of codes k_j^{red} with power allocation P/C until 50% of the original demand is met. Its operation can be summarized as follows:

1. When a request j is presented, find the number of codes k_j^{red} needed to complete demand s_j^r reduced by 50% with $\frac{P}{C}$ power per code.
2. When a job is completed select the pending user request, if any, with earliest release time a_j.

Theorem 4.2 *Denote by $\mathcal{I}'$ the instance where each job size s_i in instance $\mathcal{I}$ is reduced to $s_i/2$, then*

$$f^{FIFO}(\mathcal{I}') \leq f^{OPT}(\mathcal{I}) + 2, \ \forall \mathcal{I}. \tag{6}$$

Proof: Consider the request r achieving the maximum flow time for the scheduling discipline FIFO($\frac{P}{C}$) applied to instance $\mathcal{I}'$. W.l.o.g., we assume that request r is the last request presented to the algorithm. Request r has been released in slot t_r and completed in slot $C_r(\mathcal{I}')$ in the algorithm's solution. Denote by t (the renewal time) the last slot in which all requests that have been presented before time t have been completed by slot t. We can restrict our attention to the subset of user requests, denoted by $\mathcal{J}' = \{j \in \mathcal{J} | a_j \geq t\}$, that have been presented at or after slot t, since these are the only requests that contribute to the flow time of request r. The completion time for request r using the FIFO($\frac{P}{C}$) discipline on instance $\mathcal{I}'$ is at most $C_r(\mathcal{I}') \leq t + \left\lceil \frac{\sum_j k_j(\mathcal{I}')}{C} \right\rceil$, where $k_j(\mathcal{I}')$ is the number of codes required to complete job j on instance $\mathcal{I}'$. Denote by s the user request completed last in the solution of the optimum on instance $\mathcal{I}$. Request s has been released at some time $t_s \leq t_r$ and therefore, $C_s^{OPT}(\mathcal{I}) - t_s \geq t - t_s + \Psi$, where

$$\Psi = max \left\{ \left\lceil \frac{\sum_j p_j^{OPT}(\mathcal{I})}{P} \right\rceil, \left\lceil \frac{\sum_j k_j^{OPT}(\mathcal{I})}{C} \right\rceil \right\}.$$

Now, we have

$$\begin{aligned} f^{FIFO}(\mathcal{I}') &= C_r(\mathcal{I}') - t_r \qquad (7)\\ &\leq t - t_r + \left\lceil \frac{\sum_j k_j(\mathcal{I}')}{C} \right\rceil \overset{(a)}{\leq} t - t_s + \Psi \\ &\leq C_s^{OPT}(\mathcal{I}) - t_s + 2 \leq f^{OPT}(\mathcal{I}) + 2, \end{aligned}$$

where (a) follows from Lemma 4.1 giving us the result.

■

This result shows that by reducing the demand, we can prove a positive result on FIFO($\frac{P}{C}$). As seen in the following theorem (whose proof is in Appendix G), if we allow for resource augmentation, a positive result can be shown on FIFO($\frac{P'}{C'}$).

Theorem 4.3 *Denote by $f^{FIFO_{Aug}}(\mathcal{I})$ the maximum response time of the scheduling discipline* FIFO$_{Aug}$($\frac{P'}{C'}$) *applied to $\mathcal{I}$ with power $P' \leq 2P$ and $C' \leq 2C$ codes. Then*

$$f^{FIFO_{Aug}}(\mathcal{I}) \leq f^{OPT}(\mathcal{I}), \ \forall \mathcal{I}. \qquad (8)$$

We now show how to transform our algorithms for the continuous case into algorithms for the discrete case. Recall that in the continuous case we assign power P/C to every code, but this may correspond to a non-feasible transmission rate at the receiver for some specific user. To move from the continuous to the discrete case, we need to round the power assignment to a value that sustains one of the discrete transmission rates[11].

We will implement a rounding scheme that allows us to turn a solution for the continuous case into a solution for the discrete case. We perform two different kinds of rounding of a power z assigned to a code:

1. *Round up*: If there exists a power $\overline{z}_1 \in (z, 2z]$ corresponding to a discrete rate, then assign power $\overline{z}_1$ to the code.
2. *Round down*: If there exists a power $\overline{z}_2 \leq z$ corresponding to a discrete rate, then assign power $\overline{z}_2$ per code.

Then for each user we choose the rounding that gives the higher rate, if the user were given all the resources *i.e.,* all the power and codes. We denote by FIFO$_{disc}$($\frac{P}{C}$), the scheduling discipline obtained by taking the discipline FIFO($\frac{P}{C}$) and applying the above rounding procedure. We also define by $\mathbf{f^{FIFO_{disc}}}(\mathcal{I})$, the maximum response time of FIFO$_{disc}$($\frac{P}{C}$) on a online job arrival instance $\mathcal{I}$. It is easy to see that the following result holds (its proof is given in Appendix H).

Lemma 4.2 *The allocation scheme for the discrete case satisfies all users demands.*

Now, we show that the approximations shown for the continuous case can be translated to the discrete rate by additional resource augmentation. The proof idea is to show that additional power and codes needed, for the rounding procedure above to be as good as the continuous case, is bounded.

Theorem 4.4 *Let us denote by $\mathcal{I}'$ the instance where each job size s_i in instance $\mathcal{I}$ is reduced to $s_i/2$. Let* FIFO$_{disc}$($\frac{P'}{C'}$) *be applied to $\mathcal{I}'$, where the power and number of codes per time slot have been augmented to P' and C' respectively. Then there exist $P' \leq 2P$, $C' \leq 2C$ such that,*

$$f^{FIFO_{disc}}(\mathcal{I}') \leq f^{OPT}(\mathcal{I}) + 2, \ \forall \mathcal{I}. \qquad (9)$$

[11] For the P/C allocation we impose a further regularity condition that the lowest discrete rate $R(1) \leq W \log(1 + \frac{gP}{C})$, *i.e.* there is a feasible discrete rate below this power allocation. Though this is perhaps a little more stringent than required, it makes the analysis simpler. As before this restriction can be removed in practice by using error-correcting codes on a group of codes, so that the combined rate is a feasible discrete rate.

Proof: Let $\mathcal{K}_1$ ($\mathcal{K}_2$) denote the number of codes whose associated power was rounded up (respectively down) in a particular time slot l. Clearly, $|\mathcal{K}_1| + |\mathcal{K}_2| \leq C$. Let the power allocated on each code i after rounding be denoted by $p^{rnd}(i)$, and clearly $p^{rnd}(i) \leq 2P/C, i \in \mathcal{K}_1$ and $p^{rnd}(i) \leq P/C, i \in \mathcal{K}_2$. Now, suppose in each time-slot, for every code $i \in \mathcal{K}_2$ we assign two codes with power $p^{rnd}(i)$, and for each code $i \in \mathcal{K}_1$ (whose power was rounded up) we assign one code. Clearly such an allocation will meet the same demand as the continuous rate FIFO($\frac{P}{C}$) schedule. Hence the schedule is equivalent to the the continuous rate FIFO($\frac{P}{C}$) schedule. Therefore using this and Theorem 4.2 the result (9) can be obtained. The only question that remains is how much resource augmentation was done to obtain this. In the new allocation we have used $P' = \sum_{i\in\mathcal{K}_1} p^{rnd}(i) + 2\sum_{i\in\mathcal{K}_2} p^{rnd}(i)$ total power and $C' = |\mathcal{K}_1| + 2|\mathcal{K}_2|$ total codes. But we have,

$$P' \leq 2\frac{P}{C}|\mathcal{K}_1| + \frac{P}{C}2|\mathcal{K}_2| \leq 2P \qquad (10)$$
$$C' = |\mathcal{K}_1| + 2|\mathcal{K}_2| \leq 2C.$$

Hence the new allocation uses at most a power $P' \leq 2P$ and a number of codes $C' \leq 2C$. ■

We can also extend the result in Theorem 4.3 to the discrete rate with more resource augmentation, giving us the following result.

Theorem 4.5 *If the scheduling discipline* FIFO$_{disc}$($\frac{P'}{C'}$) *is applied to* $\mathcal{I}$ *with power* $P' \leq 4P$ *and* $C' \leq 4C$ *codes, then*

$$f^{FIFO_{disc}}(\mathcal{I}) \leq f^{OPT}(\mathcal{I}),\ \forall\mathcal{I}. \qquad (11)$$

4.2 Other QoS crieria

Although we focused our attention on minimizing the maximum response time in this paper, we can extend our results to other optimization criteria such as *minimize total weighted response time*, $\sum_i w_i(c_1 - a_i)$, where arbitrary weights w_i are specified for each request i. In particular we consider arbitrary weights w_i and for average flow time (where $w_i = \frac{1}{n}$, for n jobs). For average flow time, we analyze the Shortest Remaining Processing Time (SRPT) which is an optimal algorithm for the scheduling problem of minimizing the average flow time on a single machine [20]. For weighted response time, we analyze Highest Density first (HDF), which at any time t schedules the pending request with maximum ratio w_j/s_j, which is defined as the *density*. The following results are proved in Appendix I.

Theorem 4.6 *Algorithm SRPT achieves the optimum average flow time if every user demand is reduced to* $\frac{1}{2(1+\epsilon)}$ *of the original demand.*

Theorem 4.7 *For any* $\epsilon > 0$, *Highest Density first is an* $\frac{1+\epsilon}{\epsilon}$ *approximation for minimizing the weighted flow time if every request is guaranteed for a fraction* $\frac{1}{2(1+\epsilon)}$ *of the original demand.*

5 Simulation Study

In this section, we study the performance of our online and offline algorithms experimentally.

5.1 Online Algorithms

The $FIFO(\frac{P}{C})$ algorithm was described in Section 4. We call this algorithm *FIFO-continuous*. Essentially, this algorithm allots P/C power to each code, and job requests are then scheduled in the order of their arrival.

The rounding procedure for converting the continuous power (rate) algorithm to a discrete power (rate) algorithm was described in Section 4.1. Since rounding the rates might result in some power and/or codes to be unused in a slot, we design discrete-rate online algorithms to minimize this potential waste of resources in order to reduce the maximum response time. We have developed three online discrete-rate algorithms, which we call *FIFO*, *2D-FIFO*, and *2D-PIKI*. Given a job, the power per code corresponding to the discrete bit rate is the same for all of these algorithms. They differ only in the way the jobs are selected for receiving service.

FIFO: This is the traditional FIFO algorithm. The request i currently in the system that has the earliest release time a_i is always selected.

2D-FIFO: The request i currently in the system that has the earliest release time a_i has higher priority over other job requests. However, if this job i leaves power/codes unused in that time-slot, other jobs j in the system are considered in the non-decreasing order of their release times a_j.

2D-PIKI: The request j currently in the system that has the highest value of power per code p_j

is selected. If this job leaves power/codes unused in that time-slot, other jobs j in the system are considered in the non-increasing order of the power per code p_j. This scheme aims to achieve a better packing in each time slot, in order to reduce the completion time.

Due to discrete nature of the rate set, in certain slots the scheduler may have some codes k^{extra} and some power p^{extra} that cannot be assigned to any job in the system, since the power per code $p_j > p^{extra}$ for all jobs j. In such a situation, the scheduler will choose the first job that received service in the slot and give it the best possible discrete rate with the remaining power and codes. This is applicable to all the three algorithms described above.

Note that no algorithm guarantees that all the power and codes will be used in every slot. Therefore, we expect to see differences between the *FIFO-continuous* algorithm and the three discrete-rate algorithms. In the remainder of this section, we will quantify the differences through simulations.

5.2 Channel Specifications

We adopt the channel specifications similar to 3G system proposals [15, 16] for our channel model.[12]

We perform experiments on a single cell and abstract the effect of out-of-cell interferers into a decrease in SINR values. The peak power available at the base station was chosen to be $P = 40W$, while the maximum number of channels was chosen as $C = 16$. The power attenuation factor $\bar{g}_u$ for user u is modeled with two components: (a) shadow loss component S, which is a log-normal shadowing variable, and (b) path loss components $P = 1/d^{\alpha}$, where d is the distance between the base station and the user and α is the distance loss exponent. We chose $\alpha = 3$, giving $\bar{g}_u \propto S/d_u^3$.

The parameters to be used for the rate calculation given in Equation (2) were chosen as follows:

$\tau = 1.67$ milliseconds, $\bar{W} = 76.8KHz$ and $\Gamma = 4.7dB$. We operated over an SINR range from -15dB/Hz to 15dB/Hz. The discrete rate set used is a set of 15 rates : {2.4, 4.8, 9.6, 19.2, 38.4, 76.8, 102.6, 153.6, 204.8, 307.2, 614.4, 921.6, 1228.8, 1843.2, 2457.6} Kbps. Under these restrictions, the maximum data rate for a mobile user in the cell will range from 10 Kbps to 2 Mbps.

[12] We would like to emphasize that our algorithms are applicable to all systems that support multiple channels and multiple rates. Such systems include the various next-generation wireless data networks.

5.3 Data Sets and Simulation Tools

The traces used in the experiments are derived from a single web-proxy server. For comparing the online algorithms with the offline optimum, we used traces that consist of up to 100 jobs that arrive over a period of 1 minute[13]. To evaluate the performance of various online algorithms under heavy demand, we use traces consisting of 4000 jobs arriving over a period of 35-40 minutes, where the requests are generated by 100 users in the cell. In all of the traces used, the minimum request size was 40 bytes, the maximum request size was 500 kilobytes, with mean request sizes ranging from 20 - 34 kilobytes. The average inter-arrival time of requests in the traces is 500 - 600 milliseconds.

In order to compute the optimum flow in the offline case, we used an optimization tool called *LOQO* [21]. Our online algorithms were evaluated using a custom-built simulator.

5.4 Experiments

We performed two kinds of experiments to evaluate our algorithms. The first set of experiments validate our theoretical results and demonstrate some interesting properties of the online algorithms, while the second experiment was designed to measure the average-case performance of our algorithms.

5.4.1 Online Heuristics

In this section, we evaluate the different online algorithms and compare their performance against the offline optimal algorithm. We used small web traces, with 100 jobs arriving over a period of 1 minute, for computing the convex programming lower bound for max-flow (see Section 3.3), which we denote by OPT. The job requests are for users who are distributed uniformly in the cell. We present the max-flow results for four such traces along with the results for the online heuristics in Table 2: all max-flow values are in terms of slots.

It can be seen that the online algorithms perform very close to the optimal, on the average. From the table, we also see that *2D-FIFO* performs the best among the three discrete-rate algorithms and also that *2D-PIKI* performs the worst. In addition, the discrete algorithms always appear to perform worse than the continuous version.

[13] The small size of these traces was primarily due to the computational restrictions on finding the optimal flow.

These inferences continue to hold true in most instances, as we will show in the subsequent examples. We simulated the three algorithms *FIFO-continuous*, *2D-FIFO* and *2D-PIKI* on 36 traces of 4000 requests each generated by 100 users distributed over the cell. Figure 1 shows the values of the max-flow computed for all the 36 traces by the three algorithms. One can still observe that *2D-FIFO* is very close to *FIFO-continuous* on all traces, while *2D-PIKI* performs the worst.

Though these results hold on the average, we have designed a synthetic trace of 8 jobs where *2D-PIKI* performs as good as the optimal algorithm, and *2D-FIFO* and *2D-PIKI* perform much better than the *FIFO-continuous* algorithm.

5.4.2 Resource Augmentation

In this set of experiments, we will examine the amount of resource augmentation needed for a discrete-rate algorithm to achieve the same max-flow as *FIFO-continuous* and compare it to theoretical bounds given in Theorems 4.4 and 4.5.

Using the same set of 36 web traces described in Section 5.4.1, the scheduling algorithms were provided with augmented power in steps $\{P, 1.25P, 1.5P, 1.75P, 2P\}$ and augmented number of codes in steps $\{C, 1.5C, 2C, 3C\}$.

In the first experiment we measure the worst case ratio between the max flow-time of *2D-FIFO* and *FIFO-continuous*. We tested the *2D-FIFO* algorithm, since it outperforms the other algorithms in the average case. The results of the first experiment are summarized in the 3-D graphic of Figure 2. We observe on all traces that *2D-FIFO* obtains a max-flow time lower than *FIFO-continuous* with power augmentation factor 1.5. We also found that a code augmentation factor of 4 is needed (without power augmentation) to obtain max-flow time lower than *FIFO-continuous*. Therefore augmenting codes is not as efficient as augmenting power. Another way to observe this is by examining (4) where the rate is $R(p,c) = Wc\log(1+\frac{gp}{c})$. Here one can see that $R(p,c) \leq Wgp, \forall c$, hence even with a large number of codes, the rate is bounded above. In contrast, the rate is an unbounded function of p.

In a second experiment, for each combination of augmented power and codes we reduce the demand in steps $\{s_i, 0.95s_i, 0.9s_i, 0.85s_i, 0.8s_i, 0.75s_i, 0.7s_i, 0.6s_i\}$ until the Max-flow for the reduced demand in the discrete case was lesser or equal than the Max-flow computed by *FIFO-continuous* with power P and codes C at 100% of the demand. The lower hull over all traces of the reduced demand for each combination of augmented code and power was taken. This represents the maximum demand reduction for a given combination of augmented resources, as shown in Figure 3.

Two observations can be made here:

(a) *The average case is better than the worst-case.* In Figure 3 we observe that if power is augmented 1.25 P , then the Max-flow in the discrete case with 95% of the original demand equals that of the continuous case with power P . Thus, a system designed for meeting the QoS needs in the average case needs to over-provision its resources by an amount much lesser than that implied by the theoretical bounds.

(b) *Code augmentation and power augmentation are not the same.* As can be seen from Figures 2 and 3, the asymmetric nature of the graphs tells us that over-provisioning codes is not very efficient compared to over-provisioning of power.

In conclusion, our experimental results demonstrate that the discrete-rate algorithms proposed in this section perform closer to the continuous rate case than the worst case analyses indicate. We also show that resource augmentation, in particular, power augmentation, will considerably enhance the performance of discrete algorithms.

6 Discussion

We have formulated and studied new parallel scheduling problems in emerging 3G/4G wireless systems. We proposed simple, online algorithms that utilize the multicode, multirate feature of 3G/4G wireless networks for scheduling user requests. Our analyses involves resource-augmentation. Our experimental results show that in practice, the proposed algorithms perform much better than our worst case analyses predict. Therefore, we expect our results to find uses in providing fine-grained QoS to the users in next generation wireless data networks. This work should serve to initiate study of multidimensional malleable parallel scheduling problems under different metrics, some of which may be relevant in next generation wireless systems. We are also not aware of other experimental studies of scheduling algorithms that explore resource-augmentation needed in the analysis as done in this paper.

References

[1] Hari Balakrishnan, Venkata N. Padmanabhan, Srinivasan Seshan, and Randy H. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 756–769, 1997.

[2] Nicholas Bambos, "Toward power-sensitive network architectures in wireless communications: concepts, issues, and design aspects," *IEEE Personal Communications*, vol. 5, no. 3, pp. 50–59, June 1998.

[3] Yuming Lu and R W. Brodersen, "Integrating power control, error correction coding, and scheduling for a CDMA downlink system," *IEEE Selected Areas in Communications*, vol. 17, no. 5, pp. 978–989, May 1999.

[4] T. Nandagopal, S. Lu, and V. Bharghavan, "A unified architecture for the design and evaluation of wireless fair queueing algorithms," in *ACM Mobicom*, 1999, pp. 132–142.

[5] D. Karger, C. Stein, and J. Wein, "Scheduling algorithms," in *Algorithms and Theory of Computation Handbook, CRC Press*, 1999.

[6] J. Turek, J. Wolf, and P. Yu, "Approximate algorithms for scheduling parallel tasks," in *In Proc. of the 4th Annual Symposium on Parallel Algorithms and Architectures*, 1992, pp. 323–332.

[7] W. Ludwig and P. Tiwari, "Scheduling malleable and nonmalleable parallel tasks," in *In Proc. of the 5th ACM SIAM Symposium on Discrete Algorithms and Architectures*, 1994, pp. 167–176.

[8] K. Jansen and L. Porkolab, "Linear-time approximation schemes for scheduling malleable parallel tasks," in *In Proc. of the 10th ACM SIAM Symposium on Discrete Algorithms*, 1999, pp. 490–498.

[9] J. Turek, W. Ludwig, J. L. Wolf, L. Fleischer, P. Tiwari, J. Glasgow, U. Schweigelshohn, and P. S. Yu, "Scheduling parallelizable tasks to minimize average response time," in *In Proc. of the 6th Annual Symposium on Parallel Algorithms and Architectures*, 1994, pp. 200–209.

[10] R. Lepère, D. Trystram, and G. Woeginger, "Approximation algorithms for scheduling malleable tasks under precedence constraints," in *Proc. of the 9th Annual European Symposium on Algorithms, LNCS 2161*, 2001, pp. 146–157.

[11] S. Acharya and S. Muthukrishnan, "Scheduling on-demand broadcasts for heterogenous: new metrics and algorithms," in *ACM MobiCom*, 1998, pp. 43–54.

[12] Niranjan Joshi, Srinivas R. Kadaba, Sarvar Patel, and Ganapathy Sundaram, "Downlink scheduling in CDMA data networks," in *ACM MobiCom*, 2000, pp. 179–190.

[13] Theodore S. Rappaport, *Wireless Communications, Principles & Practice*, Prentice Hall, Inc., Piscataway, New Jersey, 1996.

[14] J M. Cioffi, G P. Dudevoir, M V. Eyuboglu, and G D. Forney, "MMSE decision-feedback equalizers and coding: Parts I & II," *IEEE Transactions on Communications*, vol. 43, no. 10, pp. 2582–2604, Oct. 1995.

[15] Paul Bender, Peter Black, Matthew Grob, Roberto Padovani, Nagabhushana Sindhushayana, and Andrew Viterbi, "CDMA/HDR; A bandwith-efficient high-speed wireless data service for nomadic users," *IEEE Communications Magazine*, vol. 38, no. 7, pp. 70–77, July 2000.

[16] Sanjiv Nanda, Krishna Balachandran, and Sarath Kumar, "Adaptation techniques in wireless packet data services," *IEEE Communications Magazine*, vol. 38, no. 1, pp. 54–65, January 2000.

[17] M. Bender, S Chakrabarti, and S. Muthukrishnan, "Flow and stretch metrics for scheduling continuous job streams," in *In Proc. of Annual Symposium on Discrete Algorithms (SODA '98)*, 1998, pp. 270–279.

[18] Michael R. Garey and David S. Johnson, *Computers and intractability : a guide to the theory of NP-completeness*, W H. Freeman., San Francisco, 1979.

[19] B. Kalyanasundaram and K.Pruhs, "Speed is as powerful as clairvoyance," in *IEEE Symposium on Foundations of Computer Sci.*, 1995, pp. 214–221.

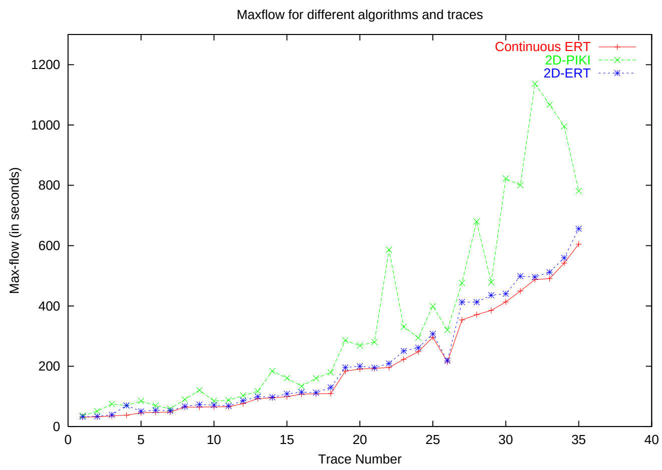

Figure 1: On-line Heuristics: Performance

[20] K.R. Baker, *Introduction to Sequencing and Scheduling*, Wiley, 1974.

[21] "LOQO Optimization Toolkit," http://www.orfe.princeton.edu/ loqo, 2000.

[22] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi, *Complexity and Approximation: Combinatorial optimization problems and their approximability properties*, Springer - Verlag, Berlin, 1999.

[23] David G. Luenberger, *Linear and Non-linear programming*, Addison-Wesley, Reading, Mass., 2nd edition, 1984.

[24] Thomas M. Cover and Joy A. Thomas, *Elements of Information Theory*, John Wiley and Sons, New York, 1991.

[25] Justin Chuang and Nelson Sollenberger, "Beyond 3G: Wideband data access based on OFDM and dynamic packet assignment," *IEEE Communications Magazine*, vol. 38, no. 7, pp. 78–87, July 2000.

[26] Thomas J. J. Starr and John Cioffi, *Understanding digital subscriber line technology*, Prentice Hall, Inc., Upper Saddle River, New Jersey, 1999.

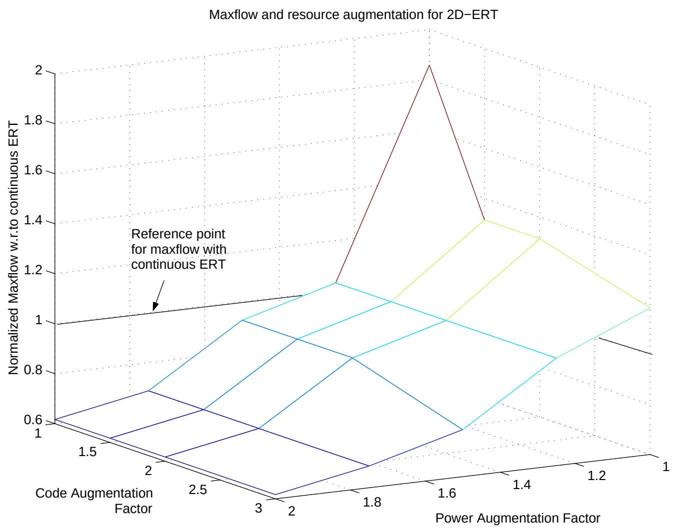

Figure 2: On-line Heuristics: Max-Flow with Resource Augmentation

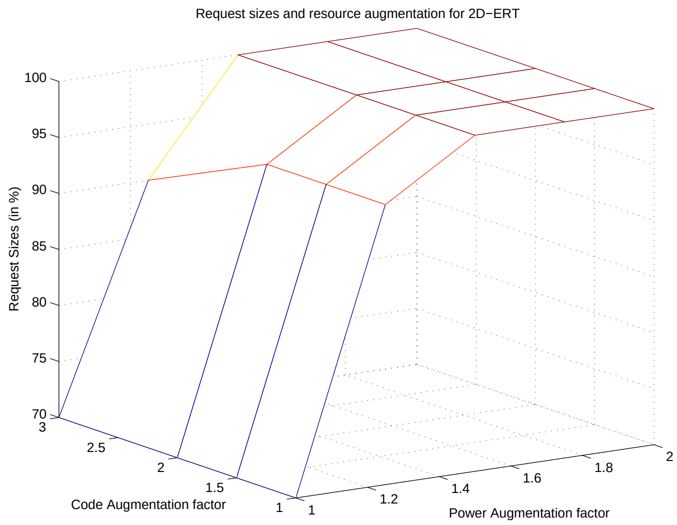

Figure 3: On-line Heuristics: Reduced Demand and Resource Augmentation

Appendices

A Proof of Theorem 3.1

Proof: We reduce the NP-complete problem *3-partition* defined as follows [18]:
INSTANCE: A set A of $3m$ elements, $a_1, a_2, \ldots, a_{3m}$, a bound B and a size $s(a_j)$, $a_j \in A$, such that $\sum_{j=1}^{3m} s(a_j) = mB$.
QUESTION: Can A be partitioned into m disjoint sets $A_1, A_2, \ldots, A_m$ such that for $1 \le i \le m$, A_i has three elements and $\sum_{a_j \in A_i} s(a_j) = B$?

The reduction is as follows: given an instance $\mathcal{I}$ of 3-partition we define an instance $\mathcal{J}$ of the combinatorial problem of meeting deadlines with $3m$

Time Indexed Program (integral)	Interval Indexed Program (fractional)
maximize 1	maximize 1
subject to	subject to
$\sum_{j=1}^{n} c(j,t) \le C \; \forall t \quad \sum_{j=1}^{n} p(j,t) \le P, \quad \forall t$	$\sum_{j=1}^{n} c(j,k) \le C \; \forall k \quad \sum_{j=1}^{n} p(j,k) \le P, \quad \forall k$
$\sum_{t=a_j}^{d_j-1} W c(j,t) \log\left(1+\frac{p(j,t)g_j}{c(j,t)}\right) \ge s_j, \; \forall j$	$\sum_{t=a_j}^{d_j-1} W \tau_k c(j,k) \log\left(1+\frac{p(j,k)g_j}{c(j,k)}\right) \ge s_j, \; \forall j$
$\sum_{t<a_j, t\ge d_j} c(j,t) = 0, \; \forall j$	$\sum_{k<a_j^{-1}, k>d_j^{-1}} c(j,k) = 0, \; \forall j$
$\sum_{t<a_j, t\ge d_j} p(j,t) = 0, \; \forall j$	$\sum_{k<a_j^{-1}, k>d_j^{-1}} p(j,k) = 0, \; \forall j$
$c(j,t), p(j,t)$ discrete values	$c(j,k), p(j,k) \ge 0, \; \forall j,k$

Table 1: Offline scheduling programs

Trace	OPT (in slots)	Continuous	Discrete		
		FIFO	FIFO	2D-FIFO	2D-PIKI
config1	109257	109891	120682	114054	114587
config2	50249	50637	55281	52263	59467
config3	36460	36725	40325	38540	46432
config4	16224	16254	17280	17210	24711

Table 2: Online Heuristics: Performance

users. The request of user j, $1 \le j \le 3m$, has size $s_j = s(a_j)$; all requests are released at time 0 and have a common deadline $D = m$. All users have the same channel gain g and, therefore, the power that users need to get desired rates depends only on the number of codes they are assigned and on the size of the request. Let p_j denote the power assigned to user j if only one code is assigned to j over all frames. We assume that there are 3 codes per frame and that the maximum power of the base station is P where $\sum_{j=1}^{3m} p_j = mP$.

We now show that there is an assignment of users to frames that meets the common deadline D if and only if $\mathcal{I}$ has a feasible solution.

Assume that $A_1, A_2, \ldots, A_m$ is a feasible solution of $\mathcal{I}$. We define a solution of $\mathcal{J}$ as follows: if element a_j, $j = 1, 2, \ldots 3m$, is assigned to set A_i then user j is assigned to frame i with the minimum power that is needed to satisfy the user's request in one code in any one frame. It is easy to see that this is a feasible solution since $\sum_{a_j \in A_i} s(a_j) = B$ implies that P is the total power required by users assigned to frame i.

Similarly it is possible to show that given a feasible solution of $\mathcal{J}$ that satisfy all users' request within m frames, it is possible to obtain a feasible solution of $\mathcal{I}$. Namely, it is sufficient to assign to set A_i, $i = 1, 2, \ldots m$, the elements that correspond to users assigned to frame i; since the total power assigned to frame i is P it follows that $\sum_{a_j \in A_i} s(a_j) = B$. That completes the proof. ■

B Proof of Theorem 3.2

Proof:

We apply the gap technique for proving inapproximability results ([22]) showing a reduction from 3-Dimensional Matching to the problem of minimizing the maximum response time in the discrete case. 3-Dimensional Matching is defined as follows:
INSTANCE: a set A of $3n$ rationals x_i, $0 < x_i < 1$, $i = 1, 2, \ldots, 3n$ such that $\sum_{i=1}^{3n} x_i = n$.
QUESTION: Can A be partitioned into n sets F_j, $j = 1, 2, \ldots, n$, such that for each j, $j = 1, 2, \ldots, n$, $|F_j| = 3$ and $\sum_{\forall x_i \in F_j} x_i = 1$?

Given c and an instance $\mathcal{I} = \{x_1, x_2, \ldots, x_{3n}\}$ of 3-Dimensional Matching we define an instance $\mathcal{I}'$ of the scheduling problem of minimizing the max response time and we show that if $\mathcal{I}$ has a solution then the max response time of $\mathcal{I}'$ is n; otherwise the max response time of $\mathcal{I}'$ is bigger than cn. Therefore, by the gap technique, if there exists a polynomial time algorithm for solving the problem of minimizing the maximum response time within a factor c approximation of the optimum, then it is possible to solve 3-Dimensional Matching in polynomial time. Since 3-Dimensional Matching is NP-complete the theorem follows.

Given c and $\mathcal{I} = \{x_1, x_2, \ldots, x_{3n}\}$ we define an instance of the scheduling problem with $(cn+1)n(c+3)$ jobs and we assume that each slot has total power equal to 10 and that there are three codes per slot.

The jobs are released in $cn+1$ phases; in each phase $n(c+3)$ jobs are released. Phase $k = 0, \ldots., cn$

is formed by two stages as follows:

Stage 1 : At time $t = kn(c+1)$, $3n$ requests $J_1, J_2, \ldots, J_{3n}$ arrive. For each k and i, the gain function of request J_i is such that by assigning power greater or equal to $3+x_i$ one code is sufficient to complete the job in a slot and $3+x_i$ is the minimum power requirement to process request J_i. Jobs released in Stage 1 of a phase are called J-jobs in the following.

Stage 2 : At time $t = kn(c+1)+n+j$, $j = 0, 1, \ldots, nc-1$ a job A_j is released. For each k and j, the gain function of this job is such that by assigning power equal or greater to 8 one code is sufficient to complete it in a slot and 8 is the minimum power requirement to process the job. Jobs released in Stage 2 of a phase are called A-jobs in the following.

The easy proof of the following Fact is omitted.

Fact B.1 *All requests released in stage 1 of a phase can be completed using n slots if and only if $\mathcal{I}$ has a solution; otherwise there are at least $n+1$ slots containing requests released in stage 1 of the phase.*

We first prove that if instance $\mathcal{I}$ of 3-Dimensional Matching has a solution then the maximum response time is equal to n. This follows since the $3n$ jobs released at time $kn(c+1)$ can be scheduled in n slots and therefore completed by time $kn(c+1)+n$. Every job of stage 2, released at time $t = kn(c+1)+n+j$, $j = 0, 1, \ldots, nc-1$, is scheduled at time t itself.

We are left to prove that if instance $\mathcal{I}$ of the 3-dimensional matching problem has not a solution then the maximum response time is bigger than cn. This follows from the following induction on the number k of phases:

Fact B.2 *Assume that instance $\mathcal{I}$ has not a solution. For every $k = 1, 2, .., cn$, if no J-job released in a phase $j < k$ has response time bigger that cn then $k-1$ A-jobs released in phases previous to k are not scheduled by time $kn(c+1)$.*

Proof:

We prove the claim by induction. For the basis of the induction we prove the claim for $k = 1$. If instance $\mathcal{I}$ has not a solution then, by Fact B.1, at least one J-job is not completed by slot $n-1$. If the response time of this job is less than $n(c+1)$then it is scheduled in some time slot in $[n, n(c+1)-1]$ and therefore one A-job will be scheduled after time $n(c+1)$. Assume the claim is true until phase $k-1$. We therefore have $k-1$ A-jobs released before time $(k-1)n(c+1)$ not completed by time slot $kn(c+1)$. All J-jobs released in Stage 1 of phase k are scheduled in at least $n+1$ different slots before time $(k+1)n(c+1)$. Since a time slot cannot accomodate an A-job together with a J-job, it then follows that at least k of the A-jobs released by time $(k+1)n(c+1)$ will be scheduled not earlier than $(k+1)n(c+1)$.

■

We therefore have $cn+1$ A-jobs released by time $(cn+1)n(c+1)$ scheduled after this time. The max response time of one A-job is therefore at least $cn+1$.

■

C Proof of Theorem 3.3

Proof Outline: We first need to show that the constraint set is convex. The Hessian $\mathbf{H}$ for the rate function $R_u(p,c) = Wc\log(1+\frac{gp}{c})$ is negative semidefinite, and therefore the function is concave in (p,c), (albeit not *strictly* concave) [23]. The other constraints are linear and hence the program is convex. There exist polynomial time solutions for convex programming problems that test feasibility [23]. For this program, the running time is polynomial not in input size ($n, \log s_i$ etc) but rather is polynomial in the number of variables which is bounded by the size of the numbers in the input (i.e., s_j).

■

D Proof of Theorem 3.4

Proof: We will show that if the time indexed convex program has a feasible solution, so does the interval indexed convex program; the other direction is trivial.

Say $\overline{p(j,t)}$ and $\overline{c(j,t)}$ be the power and code assignments respectively at time frame t to user j in the time indexed convex program. Therefore, these values satisfy all the constraints of the time indexed convex program. We now claim that $p(j,k) = \frac{\sum_{t\in k}\overline{p(j,t)}}{\tau_k}$ and $c(j,k) = \frac{\sum_{t\in k}\overline{c(j,t)}}{\tau_k}$ are feasible values in the interval indexed convex program for user j in interval k, for all users and intervals. That is, these values would satisfy the constraints of the interval indexed convex program. Clearly we can bound $\sum_j p(j,k)$ as

$$\sum_j \frac{\sum_{t\in k}\overline{p(j,t)}}{\tau_k} = \sum_{t\in k}\frac{\sum_j \overline{p(j,t)}}{\tau_k} \leq \sum_{t\in k}\frac{P}{\tau_k} \leq P.$$

So the power constraint is satisfied; similarly, the code constraint is satisfied too. We have

$$\frac{1}{\tau_k}\sum_{t\in k}\overline{c(j,t)}\log(1+\frac{g_j\overline{p(j,t)}}{\overline{c(j,t)}}) \leq c(j,k)\log(1+\frac{g_j p(j,k)}{c(j,k)})$$

due to Jensen's inequality for multidimensional functions which states that $\mathbb{E}[f(\mathbf{X})] \leq f(\mathbb{E}[\mathbf{X}])$ for a concave function $f(\cdot)$ [24]. Therefore, the demand constraint is satisfied which completes the proof. ■

E Proof of Theorem 4.1

Proof: Let $C-1$ jobs arrive in a batch every time slot so that they need $\frac{P+\delta}{C}$ power each and one code to complete and such that $0<\delta\leq\frac{P}{C-1}$, which implies that the jobs can be scheduled in one time slot.[14] Such a sequence would be scheduled in two time-slots by FIFO($\frac{P}{C}$), since it assigns two codes for each job, and therefore, would need $2C-2$ codes for every $C-1$ jobs. Hence, the job batch that arrives at the Mth time-slot is scheduled in the same time slot by the optimum and therefore has a response time of 1, whereas the online FIFO($\frac{P}{C}$) serves this job set only after $2(M-1)$ time slots have elapsed. Hence, this job set has a response time $2+2(M-1)-M=M$, for any M. ■

F Proof of Lemma 4.1

Proof: For each job j of size s_j on the original instance $\mathcal{I}$, let the pair p_j^{GS}, k_j^{GS} be such that

$$\left(p_j^{GS},k_j^{GS}\right) = \text{argmin}_{(p_j^c,k_j^c):s_j\leq Wk_j^c\log(1+\frac{g_jp_j^c}{k_j^c})}\left[\frac{p_j^c}{P}+\frac{k_j^c}{C}\right] \tag{12}$$

where p_j^{GS}, k_j^{GS} are the total power and codes assigned to user j. This solution allocates a power per code $p_j^s = p_j^{GS}/k_j^{GS}$ to the jth job/user. We now show that allocating power P/C per code is not much worse in terms of this criterion. For each code which uses power p_j^s we allocate $r_j^s = \left\lceil\frac{p_j^s}{P/C}\right\rceil$ codes with power $\frac{P}{C}$ in each code. Since $r_j^s \geq \frac{p_j^s}{P/C}$,

$$\begin{aligned} s_j &\overset{(a)}{\leq} Wk_j^{GS}\log(1+p_j^s g_j) \\ &\leq Wk_j^{GS}\log\left(1+r_j^s\frac{P}{C}g_j\right) \\ &\leq Wk_j^{GS}r_j^s\log\left(1+\frac{P}{C}g_j\right). \end{aligned} \tag{13}$$

where (a) is due to (12). Hence using this allocation the demand s_j of each user j is satisfied. As a result we can give $k_j^{alloc} \overset{def}{=} k_j^{GS}\lceil\frac{p_j^{GS}/k_j^{GS}}{P/C}\rceil$ codes of power $\frac{P}{C}$ and still complete the job. Therefore, for the P/C allocation, if we use k_j^{alloc} codes for the job j, we obtain,

$$\frac{k_j^{alloc}}{C} = \frac{k_j^{GS}}{C}\left\lceil\frac{p_j^{GS}/k_j^{GS}}{P/C}\right\rceil \leq \left(\frac{p_j^{GS}}{P}+\frac{k_j^{GS}}{C}\right) \tag{14}$$

If p_j^{alloc} is the overall power allocated to j, we have $p_j^{alloc} = k_j^{alloc}P/C$, and hence,

$$\frac{1}{2}\left(\frac{p_j^{alloc}}{P}+\frac{k_j^{alloc}}{C}\right) = \frac{k_j^{alloc}}{C} \leq \left(\frac{p_j^{GS}}{P}+\frac{k_j^{GS}}{C}\right) \tag{15}$$

We now relate this to the optimal solution on the instance $\mathcal{I}$. Denote by $p_j(\mathcal{I})$ and $k_j(\mathcal{I})$ respectively the total power and the number of codes allocated to j when the P/C allocation is applied to instance $\mathcal{I}$. Therefore[15],

$$\begin{aligned} \frac{k_j(\mathcal{I}')}{C} &\leq \frac{1}{2}\frac{k_j(\mathcal{I})}{C} \\ &\leq \frac{1}{2}\left(\frac{p_j^{GS}(\mathcal{I})}{P}+\frac{k_j^{GS}(\mathcal{I})}{C}\right) \\ &\leq \frac{1}{2}\left(\frac{p_j^{OPT}(\mathcal{I})}{P}+\frac{k_j^{OPT}(\mathcal{I})}{C}\right) \\ &\leq max\left\{\frac{p_j^{OPT}(\mathcal{I})}{P},\frac{k_j^{OPT}(\mathcal{I})}{C}\right\} \end{aligned} \tag{16}$$

■

[14]Recall that the maximum power and codes available in a slot are P and C respectively.

[15]Note that the optimal assignment need not necessarily assign equal power per code across the time slots where it schedules the jth user. However, due to the joint concavity of the rate in terms of power and code assignment (see in proof of Theorem 3.3) the rate for a given user can only increase by giving equal power assignment per code across time-slots, provided the power constraint is satisfied. Therefore, the optimal solution has a tighter constraint than the minimization in (12) and hence the third inequality in (16) is satisfied.

G Proof of Theorem 4.3

Proof: Let the optimal scheme on instance $\mathcal{I}$ allocate power $p^l_j(i)$ to user j, on the ith code in time slot l. Let us assign $r^l_j(i) = \lceil \frac{p^l_j(i)}{P/C} \rceil$ codes with power P/C per code. Due to the power constraint we have for the lth time slot: $\sum_j \sum_{i=1}^{C} p^l_j(i) \leq P$. For every slot l, we can now easily give an upper bound to the total number of codes needed with the P/C allocation:

$$\sum_j \sum_{i=1}^{C} r^l_j(i) \leq \sum_j \sum_{i=1}^{C} \left[\frac{p^l_j(i)}{P/C}\right] + C \leq 2C, \qquad (17)$$

where the second inequality is due to the power constraint. Therefore, there exists a P/C allocation which achieves the same schedule as the optimal but using power $P' \leq 2P$ and codes $C' \leq 2C$. The problem of scheduling jobs with P/C power per code is like a single processor scheduling problem [17], and FIFO is optimal for this problem with respect to maximum response time, proving the result. ∎

H Proof of Lemma 4.2

Proof: User j is allocated with power per code x. For every code allocated with exactly x, rounding up will increase the transmission rate achieved on a code by a user. Rounding down will result, by Fact 3.1, in a transmission rate that is at least half of the transmission rate in the continuous case. Therefore, by assigning two codes, the transmission rate for the user does not decrease. Hence, the demand is satisfied by the rounding scheme. ∎

I Other optimization criteria

I.1 Proof of Theorem 4.6

Proof: Consider user j and denote by f^{red}_j and by f^{OPT}_j the number of frames used by the algorithm working with reduced demands for user j and a lower bound on the number of frames used by the optimum.

From (16) it follows

$$f^{red}_j \leq \frac{p^r_j}{P} + \frac{k^r_j}{C} \quad \leq \quad max\{\frac{p^{OPT}_j}{P}, \frac{k^{OPT}_j}{C}\} \leq f^{OPT}_j$$

We know that SRPT on allocation f^{OPT}_j gives an optimal solution.

Consider the schedule produced by SRPT on f^{OPT}_j and stop processing request j when request j is allocated for f^{red}_j frames. This new schedule has an average flow time certainly smaller than SRPT on f^{OPT}_j. On the other hand, this schedule has an average flow time that is certainly higher than the result of applying SRPT on f^{red}_j for which the theorem follows. ∎

I.2 Proof of Theorem 4.7

Proof: We denote by f^{red}_j and by f^{OPT}_j respectively the maximum number of frames used by the algorithm working with reduced demands and a lower bound on the number of frames used by the optimum. Since demand is reduced to $\frac{1}{2(1+\epsilon)}$ of its original size, equation (16) implies that $f^{red}_j \leq \frac{1}{1+\epsilon} f^{OPT}_j$. The optimum has to allocate at least f^{OPT}_j frames to request j in order to meet its demand.

For the sake of analysis, we compare HDF with a lower bound on the optimum given by a fractional version of HDF, denoted in the following by FHDF. HDF and FHDF work the same way, but FHDF is able to reduce the weight of a job fractionally, as the job is processed. In particular, if an amount s of job j has already been scheduled by frame t, then the weight of job j is reduced to $w^{FHDF}_j(t) = w_j(s_j - s)/s_j$. The size of job j at time t is analogously reduced to $s_j(t) = s_j - s$. Observe $w_j(t)/s_j(t) = w_j/s_j$, hence the density of a job remains constant along the execution of FHDF. This implies that FHDF schedules jobs in the same order as HDF.

The maximum ratio between the weights not completed by HDF and FHDF at any time t is an upper bound on the approximation ratio of the two algorithms. Since $f^{red}_j \leq \frac{1}{1+\epsilon} f^{OPT}_j$ and both FHDF and HDF schedule jobs in the same order, the number of frames allocated by FHDF to any request j by time t is never larger than the number of frames allocated to request j by HDF. It follows that the jobs that are completed by $FHDF$ at any time t are also completed by HDF within the same time. As a consequence, the ratio between the uncompleted weight of the two algorithms is given by the maximum over all jobs j of $\frac{w_j}{w^{FHDF}_j(t)}$. Assume the worst case in which both HDF and FHDF started job j at the same time. When HDF is about to complete request j, FHDF is still left with at least $\frac{\epsilon}{1+\epsilon} f^{OPT}_j$ frames to be allocated to request j, therefore with a fraction at least $\frac{\epsilon}{1+\epsilon}$ of the original weight of request

j. The ratio between the remaining weight of HDF and FHDF is then bounded by

$$\frac{w_j}{w_j^{FHDF}(t)} \leq \frac{1+\epsilon}{\epsilon},$$

for which the claim of the theorem holds. ■

The Highest Density first heuristic, when specialized to average stretch and average response time, becomes the Shortest Processing Time first heuristic (SPT), that at any time schedules the pending request that has shortest processing time. (Note that the selection policy still applies to the SPT algorithm.)

Corollary I.1 *For any $\epsilon > 0$, Shortest Processing Time first is an $\frac{1+\epsilon}{\epsilon}$ approximation for minimizing the average stretch (and response time) if every request is guaranteed for a fraction $\frac{1}{2(1+\epsilon)}$ of the original demand.*

J Overview of Next Generation Wireless Multirate Data Networks

Our scheduling model abstracts multirate scheduling in many next generation wireless data systems. Here, we will briefly review two examples, namely, CDMA and and OFDM systems, and leave other examples such as multislot TDMA (EDGE) out of the discussion.

CDMA In *code division multiple access* (CDMA) systems, all the users share the entire transmission bandwidth and users are distinguished by the use of *signatures* (also called *codes*) assigned to them. There are two main CDMA proposals competing for the next generation standards. One is *cdma2000* which is an evolution of Qualcomm's IS-95 second generation CDMA system and is designed to use a frequency spectral bandwidth of 1.25MHz. The other is the *wideband CDMA* (WCDMA) which uses a larger frequency spectral bandwidth of 5 MHz and is being specified in Europe and Japan [16]. Both cdma2000 and WCDMA envisage higher rates through assigning multiple codes to users (*code-aggregation*) and variable rates through coding techniques. These two proposals are intended for both data and voice applications. Another system of significant interest is a purely data system called the *High Data Rate* (HDR) system being designed by Qualcomm [15]. This system is designed with a large range of available data rates, using sophisticated error-correcting coding schemes with higher latency, making it suitable for non-real time data traffic. In all CDMA proposals, there is a pilot signal in the broadcast control channel that enables the access terminal to measure the link channel conditions and this is reported back to the base-station. Therefore, the base-station has access to transmission conditions for users typically at a time-scale of a few milliseconds per measurement.

Wideband OFDM Another air-interface which is being actively studied is based on wideband *Orthogonal Frequency Division Multiplexing* (OFDM) [25]. Here the wideband channel is divided into narrow frequency *tones* in a manner similar to traditional frequency-division multiplexing. However, there are important differences since the transmission frame is assembled using a Fast Fourier Transform (FFT) in OFDMs. As a result, transmissions become less susceptible to multipath propagation effects [25]. Also, the use of FFT allows one to dynamically assign different sets of tones to different user. Moreover, using powerful error-correcting codes variable rates can allocated on the tones. Though this is not among the third generation wireless standard proposals, it is receiving significant research and industry attention. The idea of OFDM is already a part of European digital audio broadcast standards the US wireline DSL standards [26], the HiperLan wireless LAN standard and several wireless local loop systems. Systems with number of tones ranging from 16 to 256 are being currently studied [25]. Our algorithms will be applicable to OFDM systems with codes interpreted as tones.