Utility-Optimal Scheduling in Time-Varying Wireless Networks with Delay Constraints

I-Hong Hou
CSL and Department of Computer Science
University of Illinois
Urbana, IL, 61801, USA
ihou2@illinois.edu

P. R. Kumar
CSL and Department of ECE
University of Illinois
Urbana, IL 61801, USA
prkumar@illinois.edu

ABSTRACT

Clients in wireless networks may have per-packet delay constraints on their traffic. Further, in contrast to wireline networks, the wireless medium is subject to fading. In such a time-varying environment, we consider the system problem of maximizing the total utility of clients, where the utilities are determined by their long-term average rates of being served within their delay constraints. We also allow for the additional fairness requirement that each client may require a certain minimum service rate. This overall model can be applied to a wide range of applications, including delay-constrained networks, mobile cellular networks, and dynamic spectrum allocation.

We address this problem through convex programming. We propose an on-line scheduling policy and prove that it is utility-optimal. Surprisingly, this policy does not need to know the probability distribution of system states. We also design an auction mechanism where clients are scheduled and charged according to their bids. We prove that the auction mechanism restricts any selfish client from improving its utility by faking its utility function. We also show that the auction mechanism schedules clients in the same way as that done by the on-line scheduling policy. Thus, the auction mechanism is both truthful and utility-optimal. Finally, we design specific algorithms that implement the auction mechanism for a variety of applications.

Categories and Subject Descriptors

C.2.1 [**COMPUTER-COMMUNICATION NETWORKS**]: Network Architecture and Design —*Wireless communication*

General Terms

Theory

This material is based upon work partially supported by US-ARO under Contract Nos. W911NF-08-1-0238 and W-911-NF-0710287, AFOSR under Contract FA9550-09-0121, and NSF under Contract No. CNS-07- 21992.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
MobiHoc'10, September 20–24, 2010, Chicago, Illinois, USA.
Copyright 2010 ACM 978-1-4503-0183-1/10/09 ...$10.00.

Keywords

Scheduling, utility maximization, NUM, deadlines, delays, auction

1. INTRODUCTION

This paper studies the problem of network utility maximization (NUM) in time-varying wireless networks, when packets have delay constraints. It is motivated by two considerations. First, delay constraints are becoming important as wireless networks are increasingly used for serving real-time traffic such as VoIP and video streaming. Also, delay constraints are critical to applications such as networked control where inner control loops can be destabilized by excessive delay, and even outer control loops used for coordination are safety critical, e.g., vehicular traffic control. Second, unlike wireline networks, where the network topologies and link capacities are static, wireless networks are time-varying in that the available bandwidth and link qualities are all time-varying due to both node mobilities and channel fading.

We first propose a system model that allows us to pose and provide solutions that address the dynamics of different entities involved. The model characterizes the system state by the collection of subsets of clients that can be served under it, and makes no other assumptions about the network. Thus, the model is general enough to be applied to a wide range of applications, including delay-constrained wireless networks with rate adaptation, mobile cellular networks, and dynamic spectrum allocation; we will specifically focus on the problem of delay-constrained wireless networks. The performance of a client is defined by the long-term average rate that it is served, subject to per-packet delay constraints. The utility gained by a client is determined by its service rate through its utility function. Further, to impose a certain degree of fairness and avoid starving some clients, we assume that each client requires a certain lower bound on its service rate. The NUM problem under this model is to maximize the total long-term utility with respect to network dynamics, per-packet delay constraints, and minimum service requirements of clients.

To solve the foregoing NUM problem in time-varying wireless networks, we first formulate it as a convex programming problem in which network dynamics are considered. We then propose an on-line scheduling policy for the NUM problem that does not require knowledge of the probability distribution of system states. We prove that the policy converges to the optimal solution of the convex programming problem and thus solves the NUM problem.

In practice, utility functions may be known only to the clients.

Thus, clients may provide a fake utility function to gain more service. To ensure that clients reveal their true utility functions, we design an auction that is based on the Vickrey-Clarke-Groves (VCG) mechanism [3, 5, 20] for scheduling. In this auction, clients announce their bids in each instance, and the server schedules service and charges clients based on their bids as well as the system state. We prove that this auction is truthful, meaning that a selfish client cannot strictly increase its net utility by lying about its utility function. We also show that the schedule derived from the auction is the same as that from the on-line scheduling policy for solving the NUM. Thus, this auction mechanism also achieves maximum total utility.

We also discuss how to implement the auction mechanism for three possible applications: delay-constrained wireless networks, mobile cellular networks, and dynamic spectrum allocation. For each of the three applications, we derive specific algorithms for both scheduling clients and charging them.

Finally, we provide simulation results for the three applications. We compare our proposed policies against state-of-the-art policies for each application. The compared policies either only focus on satisfying the minimum service requirements of clients or consider utilities on a per-interval base rather than long-term average performance. Simulation results show that these policies can result in low utility and serious unfairness. This suggests that, where long-term average performance is concerned, these compared policies are not applicable. On the other hand, our proposed policies not only satisfy the minimum service requirements for all clients but also achieves the highest utilities in all three applications.

The rest of the paper is organized as follows. Section 2 summarizes related work. Section 3 describes a system model for time-varying wireless networks and defines the NUM problem. We demonstrate that several applications can be described by our system model in Section 4. Section 5 formulates the NUM problem as a convex programming problem and describes a simple on-line scheduling policy that solves it. Section 6 designs an auction mechanism that not only makes clients report their true utility functions but also achieves the maximum total long-term utility. Section 7 discusses algorithms for implementing the auction mechanism under several applications. Simulation results are presented in Section 8. Finally, Section 9 concludes this paper.

2. RELATED WORK

First, we note that there is no work other than [8], to our knowledge, that addresses utility maximization when packets have delay constraints. Rather utility maximization has been studied in the context of throughput only. Second, we note that even such work that studies the NUM problem mostly studies it in the context of static networks, and cannot be applied to time-varying wireless networks. The only work [8] that studies delay constraints in a utility maximization framework, also considers only a static network.

The work on network utility maximization was initiated by Kelly [10] and Kelly, Maulloo, and Tan [11], who considered utility-optimal rate control algorithms in wireline networks. Lin and Shroff [13] have considered the NUM problem in wireless networks with multi-path routing. These works assume the network topology is static. Liu, Chong, and Shroff [14] and O'Neill, Goldsmith, and Boyd [17] have both considered the NUM problem in a time-varying environments. However, they have evaluated the performance of clients on a per-interval base. Yi and Chiang [21] have summarized other existing work on the NUM problem.

Shakkottai and Srikant [19] and Raghunathan et al [18] have studied maximizing total throughput for delay-constrained traffic over unreliable wireless links. Their results, however, may result in serious unfairness. Hou and Kumar [7] have studied an analytical model for delay-constrained wireless networks and proposed feasibility-optimal scheduling policies that satisfy the minimum service requirements of clients. Their work has not considered utilities gained by clients. As noted above, the work [8] has proposed an utility-optimal scheduling policy for delay-constrained traffic over unreliable wireless links. This work only treats the case when link reliabilities are time-invariant and does not consider rate adaptation. Thus, it is not suitable for networks with fading channels or with rate adaptation.

Dynamic spectrum allocation has also attracted increasing research interest. Gandhi et al [4] has proposed a framework for spectrum auctions. Zhou et al [22] and Jia et al [9] have studied designing truthful spectrum auction mechanisms. These works have focused on the scenario where spectrum auctions are carried out infrequently.

3. SYSTEM MODEL

Consider a wireless system with one server and N clients, numbered $\{1, 2, \ldots, N\}$. Time is divided into *time intervals*. Each client desires some service in each time interval. The service requirement within a time interval of each client is indivisible; that is, the server can only either fully meet the demand of a client or not serve it at all. At the beginning of each time interval, the server obtains the current channel condition. Both the demands of clients and the channel condition can be time-varying, and together we call them the *system state* in each time interval. The sever can learn the system state by either polling, probing, or estimating. Since these operations are costly and cannot be carried out too frequently, the server assumes that the system state does not change within an interval. Due to limited wireless resources, the server may be only able to serve some particular subsets of clients in each system state. To be more specific, we denote the system state in the k^{th} time interval by $c(k) \in C$, where C is a finite set, and $\{c(1), c(2), \ldots\}$ are i.i.d. random variables with $Prob\{c(k) = c\} =: p_c$. In practice, not only the system state but also the distribution of system states can be time-varying. However, the distribution of system states usually evolve on a much slower time scale compared to the length of a time interval and thus is assumed to be static.

A subset S of clients is said to be *feasible* under system state c if it is possible for the server to serve all clients in S. For simplicity, we represent a system state c by the collection of subsets S that are feasible under c. Thus, we have $S \in c$ if S is feasible under c, and $S \notin c$ otherwise. Since the constraints of feasible sets can be defined arbitrarily, this model can be applied to a wide range of applications. We will illustrate some examples of applications in Section 4. In particular, it can accommodate per-packet delay constraints and rate adaptation.

The server is in charge of choosing a feasible subset $S \in c(k)$ to serve in each time interval k. The server's choice is described by a *scheduling policy*.

DEFINITION 1. Let $h(k)$ be the system's history up to the

k^{th} time interval. A *scheduling policy* is a function $\eta : (h(k-1), c(k)) \mapsto 2^{\{1,2,\ldots,N\}}$, such that given history $h(k-1)$ and current system state $c(k)$, the server chooses the subset $\eta[h(k-1), c(k)] \in c(k)$ of clients to serve. All clients $n \in \eta[h(k-1), c(k)]$ are considered to be *served in the k^{th} time interval*.

As the system state is time-varying, it is less meaningful to discuss the performance of clients on a per-interval base. Rather, we measure the performance of a client through its average rate of being served. We define the *service rate* of a client n as follows:

DEFINITION 2. Let $q_n(k)$ denote the *service rate* of client n up to the k^{th} time interval, defined by the recursion:

$$q_n(k+1) = \begin{cases} (1-\alpha_k)q_n(k) + \alpha_k, & \text{if client } n \text{ is served at the } k^{th} \text{ interval,} \\ (1-\alpha_k)q_n(k), & \text{otherwise,} \end{cases}$$

where $0 \leq \alpha_k \leq 1$, for all k. The *long-term service rate* of client n is defined as $q_n := \liminf_{k\to\infty} q_n(k)$.

In the above definition, α_k is a system-wide variable that is assumed to be the same for all clients. For example, by setting $\alpha_k \equiv \frac{1}{k}$, $q_n(k)$ becomes the proportion of time intervals that client n is being served. On the other hand, setting all α_k to be a constant makes $q_n(k)$ a weighted-average of service where recent service is more important than service a long time ago.

We further assume that each client n has an *utility function* $U_n(\cdot)$. The utility functions are strictly increasing, strictly concave, and infinitely differentiable. At the k^{th} time interval, client n receives utility that is equivalent to an amount $\frac{1}{\alpha_k}U_n(q_n(k))$ of money. The scaling factor $\frac{1}{\alpha_k}$ of the amount of money received by client n is set to equalize the effects of events in each interval. Section 6.1 provides a more detailed explanation of this setting. The *long-term utility* of client n is defined as $\liminf_{k\to\infty} U_n(q_n(k))$, which equals $U_n(q_n)$ since $U_n(\cdot)$ is continuous.

Finally, to enforce some form of fairness among clients, we also assume that each client n has a requirement of *minimum long-term service rate*, $\underline{q}_n$; that is, it requires $q_n \geq \underline{q}_n$ with probability 1. We assume that the minimum long-term service rate requirements are strictly feasible, that is, there exists some scheduling policy that ensures $q_n > \underline{q}_n$, for all n.

We are interested in maximizing the *total long-term utility* of the network, $\sum_{n=1}^{N} U_n(q_n)$. The NUM problem of this framework can hence be expressed as:

$$\begin{aligned} &\text{Max } \textstyle\sum_{n=1}^{N} U_n(q_n) \\ &\text{s.t. Network dynamics and feasibility constraints,} \\ &\text{and } q_n \geq \underline{q}_n, \forall n. \end{aligned}$$

However, this formulation only considers the long-term behavior of the system. A solution to this NUM problem may not translate into an implementable scheduling policy, which would have to make decisions on a per-interval basis. Thus, we also wish to design *utility-optimal* scheduling policies.

DEFINITION 3. A scheduling policy η is said to be *utility-optimal* if, by applying η, $\sum_{n=1}^{N} U_n(q_n(k))$ converges to the optimal value of the NUM problem almost surely as $k \to \infty$.

4. EXAMPLES OF APPLICATIONS

We will first discuss several applications that can be described by our framework.

4.1 Delay-Constrained Wireless Networks with Rate Adaptation

We consider the model introduced in [6] that characterizes a system where clients generate real-time traffic, and which was extended to allow fading in [7]. Assume that there are N wireless clients and one access point (AP). Time is assumed to be slotted and divided into time intervals, each consisting of T consecutive time slots. At the beginning of each time interval, packets for each client arrive at the AP. Each client specifies a delay bound of τ_n time slots, with $\tau_n \leq T$. The packet for client n is to be delivered no later than the τ_n^{th} time slot in each time interval. Otherwise, the packet expires and is dropped from the system.

Due to channel fading, the link qualities between the AP and the client can be time-varying. We assume that the AP has full knowledge of the current channel state. The AP then applies rate adaptation for error-free transmissions. Thus, the transmission rates for different clients can be different, which in turn results in different transmission times. We define $t_{c,n}$ as the number of time slots required for an error-free transmission for client n under system state c. A scheduling policy is one which selects an ordered subset $S = \{s_1, s_2, \ldots, s_m\}$ of clients and transmits packets for clients in S according to the order. The ordered subset is considered feasible under system state c if packets for all clients in S can be delivered before their respective delay bounds, or, to be more specific, $\sum_{n=1}^{i} t_{c,s_n} \leq \tau_{s_i}$, for all $1 \leq i \leq m$. In this scenario, the service rate of each client reflects its *timely throughput*.

4.2 Mobile Cellular Network

Consider a mobile cellular network with a base station and N users. The system may have more than one channel, but each channel can be occupied by at most one user at any given time. We assume that time is slotted, where a *time slot* corresponds to a *time interval* in the system model. The length of a time slot is defined as the time needed for transmitting a packet plus any control overhead. Also, due to mobility, the link qualities between the base station and an user can be time-varying. We consider an ON/OFF model for links. The link between an user and the base station is considered ON if a packet can be transmitted between the two without errors, and considered OFF otherwise. We assume that the base station never transmits packets to users with OFF links. Thus, the system state at any time slot can be described as the set of users with ON links. A subset S of users is considered feasible under some system state c if for any user $n \in S$, the link between user n and the base station is ON, and the size of S is smaller than or equal to the number of channels. A scheduling policy is one which chooses, based on current system state and past history, a feasible subset of users and assigns channels to each of them. Finally, the service rate of each user is equal to its throughput.

4.3 Dynamic Spectrum Allocation

Consider a scenario with one primary user and N secondary users. The primary user holds licenses for several channels over a large geographical region. TV broadcasters are typical examples of primary users. The primary user only uses

a portion of its licensed channels and is willing to allocate unused channels to secondary users. The secondary users are scattered throughout the region and constrained to much smaller transmission powers compared to the primary user, which makes spatial reuse possible. Still, some secondary users may interfere with each other and thus cannot be allocated the same channel simultaneously. We use a conflict graph $G = (V, E)$ to represent the interference relations between secondary users, where V is the set of secondary users and there is an edge between two users if they interfere with each other.

The primary user allocates unused channels periodically. Since the network activity of the primary user can be time-varying, the number of unused channels can also be time-varying. A scheduling policy is one which chooses disjoint subsets of secondary users for each unused channel, with the constraint that two users that are assigned the same channel cannot share a link in the conflict graph.

5. A GENERAL METHOD FOR UTILITY MAXIMIZATION

In this section, we propose a general method for solving the NUM problem in time-varying wireless networks with minimum service requirements. We first show that the NUM problem can be formulated as a convex programming problem. Although the formulation requires explicit knowledge of the distribution of system states, i.e., the values of probability $[p_c]$, we will show the surprising result that there exists an on-line scheduling policy that does not need any information on the distribution of system states, and is, further, also utility-optimal. For simplicity, we assume that $\alpha_k := 1/k$, that is, $q_n(k)$ is the proportion of time intervals that client n has been served until the k^{th} time interval. We will discuss the case where α_k is a constant for all k at the end of this section.

5.1 Convex Programming Formulation

Define $p_c(k)$ and $f_{c,S}(k)$, for all system states c and subsets $S \in c$, recursively, as follows:

$$p_c(k+1) = \begin{cases} \frac{k-1}{k} p_c(k) + \frac{1}{k}, & \text{if } c(k) = c, \\ \frac{k-1}{k} p_c(k), & \text{otherwise,} \end{cases}$$

and

$$f_{c,S}(k+1) = \begin{cases} \frac{k-1}{k} f_{c,S}(k) + \frac{1}{k}, & \text{if } c(k) = c \text{ and} \\ & S \text{ is scheduled at the } k^{th} \text{ interval,} \\ \frac{k-1}{k} f_{c,S}(k), & \text{otherwise.} \end{cases}$$

These two variables can be thought of as the relative frequencies of occurrence of the system state c and the event that subset S is scheduled under system state c, respectively. Also, we have $\sum_{S \in c} f_{c,S}(k) = p_c(k)$ and $\sum_c \sum_{S:S \in c, n \in S} f_{c,S}(k) = q_n(k)$ for all c and k. For ease of discussion, we only consider scheduling policies where $f_{c,S} := \lim_{k \to \infty} f_{c,S}(k)$ exists for all system states c and subsets S. Thus, we have $\sum_{S \in c} f_{c,S} = p_c$ and $\sum_c \sum_{S:S \in c, n \in S} f_{c,S} = q_n$. The NUM problem can be described as the following convex programming problem:

$$\begin{aligned} & \text{Max } \textstyle\sum_{n=1}^N U_n(q_n) = \sum_{n=1}^N U_n(\sum_c \sum_{S:S \in c, n \in S} f_{c,S}) \\ \text{s.t.} \quad & \textstyle\sum_{S \in c} f_{c,S} = p_c, \forall c, \\ & q_n = \textstyle\sum_c \sum_{S:S \in c, n \in S} f_{c,S} \geq \underline{q}_n, \forall n, \\ \text{over} \quad & f_{c,S} \geq 0. \end{aligned}$$

While typical techniques for solving a convex programming problem can be applied to solve this NUM problem, such solutions may not be directly translatable into a scheduling policy for our time-varying network. Also, a solution based on solving the convex programming problem would require the knowledge of the probability distribution of system states. In practice, this knowledge may not always be available to the server. Thus, a scheduling policy that makes decisions based only on past history and current system state is needed.

5.2 An On-line Scheduling Policy

We now describe an on-line scheduling policy, and prove that it is utility-optimal. This scheduling policy only requires information on the past history and the current system state, and, surprisingly, does not need any knowledge of the actual probability distribution of system states. Thus, it is easily implementable. The scheduling policy is based on dual decomposition, which is similar to the approach used in Lin and Shroff [13], although they do not consider network dynamics.

We assign a Lagrange multiplier λ_n for each constraint $\sum_c \sum_{S:S \in c, n \in S} f_{c,S} \geq \underline{q}_n$. The resulting Lagrangian of the resulting convex programming problem is:

$$\begin{aligned} L(f, \lambda) = \quad & \textstyle\sum_{n=1}^N U_n(\sum_{c,S:S \in c, n \in S} f_{c,S}) \\ & + \textstyle\sum_{n=1}^N \lambda_n(\sum_{c,S:S \in c, n \in S} f_{c,S} - \underline{q}_n), \end{aligned}$$

where f denotes the vector consisting of $[f_{c,S}]$, for all c and S, and λ denotes the vector $[\lambda_n]$. The dual objective function is:

$$D(\lambda) = \max_{f: f_{c,S} \geq 0; \sum_{S \in c} f_{c,S} = p_c, \forall c} L(f, \lambda).$$

Since the minimum long-term service rate requirements, $[\underline{q}_n]$, are strictly feasible, there exist $[f_{c,S}]$ such that

$$\sum_{S \in c} f_{c,S} = p_c, \text{ and } \sum_c \sum_{S:S \in c, n \in S} f_{c,S} > \underline{q}_n,$$

for all n. By Slater's condition, $\min_\lambda D(\lambda)$ equals the maximum total utility.

Let $\lambda(k) = [\lambda_n(k)]$ denote Lagrange multipliers that are used in the k^{th} period. The maximum total utility can be achieved by solving two subproblems: maximizing

$$\lim_{k \to \infty} E[L(f(k), \lambda)],$$

for any given λ, and minimizing

$$\lim_{k \to \infty} E[D(\lambda(k))].$$

We will refer to these two subproblems as the *primal problem* and *dual problem*, respectively.

We first discuss how to solve the primal problem. Due to the constraint $\sum_{S \in c} f_{c,S} = p_c$, $[f_{c,S}]$ is an optimal solution if and only if $\frac{\partial L}{\partial f_{c,S}} := \sum_{n \in S} (U_n'(q_n) + \lambda_n) = \max_{S' \in c} \frac{\partial L}{\partial f_{c,S'}}$ for every c and S such that $f_{c,S} > 0$. Recall that $U_n(\cdot)$ is

strictly concave, and $U_n'(\cdot)$ is a strictly decreasing function. Suppose, at some time interval k with $c(k)=c$, there exists a subset S feasible under c such that $\sum_{n\in S}(U_n'(q_n(k))+\lambda_n) > \sum_{n\in S'}(U_n'(q_n(k))+\lambda_n)$ for all other subsets S' feasible under c. We wish to narrow the difference between S and all other S'. One obvious choice would be to schedule the subset S in the time interval, so as to increase $q_n(k+1)$ for all $n\in S$, and thus decrease $\sum_{n\in S}(U_n'(q_n(k))+\lambda_n)$. In fact, as we shall see in the lemma below, selecting the feasible subset S that maximizes $\sum_{n\in S}(U_n'(q_n(k))+\lambda_n)$ also points in the steepest ascent direction of L.

DEFINITION 4. Given λ and $f(k)$, a *max-weight scheduling policy* is one that schedules a feasible subset $S\in c(k)$ that maximizes $\sum_{n\in S}(U_n'(q_n(k))+\lambda_n)$ in each time interval k.

LEMMA 1. *Let $\Delta f(k)$ be the vector consisting of the elements $\Delta f_{c,S}(k) := f_{c,S}(k+1)-f_{c,S}(k)$ for all c and S. Given λ and $f(k)$, the max-weight scheduling policy also maximizes $E[\nabla L(f,\lambda)\cdot\Delta f(k)|f_{c,S}(k)]$.*

PROOF. Recall that we have:

$$f_{c,S}(k+1)=\begin{cases}\frac{k-1}{k}f_{c,S}(k)+\frac{1}{k}, & \text{if } c(k)=c \text{ and}\\ & S \text{ is scheduled at the } k^{th} \text{ interval},\\ \frac{k-1}{k}f_{c,S}(k), & \text{otherwise.}\end{cases}$$

Thus, $\Delta f_{c,S}(k)=\frac{1}{k}(1-f_{c,S}(k))$ if $c(k)=c$ and S is scheduled, and $\Delta f_{c,S}(k)=-\frac{1}{k}f_{c,S}(k)$, otherwise. Let $\hat{f}_{c,S}(k)$ be the probability that $c(k)=c$ and S is scheduled under the max-weight scheduling policy. We then have:

$$\begin{aligned}&E[\nabla L(f,\lambda)\Delta f(k)|f_{c,S}(k)]=\textstyle\sum_{c,S}E[\frac{\partial L}{\partial f_{c,S}}\Delta f_{c,S}(k)|f_{c,S}(k)]\\ &=\tfrac{1}{k}\{\textstyle\sum_{c,S}[\hat{f}_{c,S}(k)-f_{c,S}(k)][\sum_{n\in S}(U_n'(q_n(k))+\lambda_n)]\}.\end{aligned}\tag{1}$$

Since $Prob\{c(k)=c\}=p_c$, $\sum_S\hat{f}_{c,S}(k)=p_c$. The term $E[\nabla L(f,\lambda)\Delta f(k)|f_{c,S}(k)]$ is maximized by setting:

$$\hat{f}_{c,S}(k)=\begin{cases}p_c, & \text{if } S=\arg\max_{S\in c}\sum_{n\in S}U_n'(q_n(k))+\lambda_n,\\ 0, & \text{otherwise.}\end{cases}\tag{2}$$

This is achieved by selecting the feasible subset S that maximizes $\sum_{n\in S}(U_n'(q_n(k))+\lambda_n)$ for every system state c. □

Next, we prove that the max-weight scheduling policy solves the primal problem.

THEOREM 1. *Under the max-weight scheduling policy,*

$$L(f(k),\lambda)\to D(\lambda), \text{ as } k\to\infty,$$

for any given λ.

PROOF. Since the utility functions are infinitely differentiable, $L(f,\lambda)$ is also infinitely differentiable. By Taylor's theorem, we have that for any f, Δf, and fixed λ,

$$L(f+\Delta f,\lambda)=L(f,\lambda)+\nabla L(f,\lambda)\Delta f+r(f,\Delta f,\lambda),$$

where $|r(f,\Delta f,\lambda)|<a(\lambda)|\Delta f|^2$, for some constant $a(\lambda)$. Now we have,

$$\begin{aligned}&E[L(f(k+1),\lambda)|f(k)]\\ \geq\ & L(f(k),\lambda)+E[\nabla L(f(k),\lambda)\Delta f(k)-a(\lambda)|\Delta f(k)|^2|f(k)]\\ \geq\ & L(f(k),\lambda)+E[\nabla L(f(k),\lambda)\Delta f(k)|f(k)]-\tilde{a}/k^2,\end{aligned}\tag{3}$$

where $\Delta f(k)$ is defined as in Lemma 1 and $\tilde{a}$ is some constant. The last inequality follows because $|\Delta f_{c,S}(k)|\leq\frac{1}{k}$ for all c,S.

Let $p_c(k):=\sum_{S\in c}f_{c,S}(k)$, which is the empirical frequency that system state c occurs, and let $\hat{f}_{c,S}(k)$ be defined as in the proof of Lemma 1. The values of $\hat{f}_{c,S}(k)$ under the max-weight scheduling policy are given as in (2). Further, let $\hat{\mu}_c(k):=\max_{S\in c}\sum_{n\in S}(U_n'(q_n(k))+\lambda_n)$, for all c. Using (1) and (2),

$$E[\nabla L(f(k),\lambda)\Delta f(k)|f(k)]\geq\frac{1}{k}\sum_c(p_c-p_c(k))\hat{\mu}_c(k).$$

Since $kp_c(k+1)$ is the number of occurrences of system state c until the k^{th} time interval, and the system state in each time interval is i.i.d. distributed, by the law of iterated logarithm [2], there exists some positive constant b such that $\limsup_{k\to\infty}\frac{k(p_c(k+1)-p_c)}{k^{1/2}(\log\log k)^{1/2}}\leq b$. Thus, for large enough k, there exists constant $\tilde{b}$ such that

$$E[\nabla L(f(k),\lambda)\Delta f(k)|f(k)]\geq-\frac{(\log\log k)^{1/2}}{k^{3/2}}\tilde{b}.$$

For large enough k, (3) can hence be bounded by

$$E[L(f(k+1),\lambda)|f(k)]\geq L(f(k),\lambda)-\frac{(\log\log k)^{1/2}}{k^{3/2}}\tilde{b}-\frac{\tilde{a}}{k^2}.\tag{4}$$

As we can see in the above, $E[L(f(k+1),\lambda)|f(k)]$ is "almost" larger than $L(f(k),\lambda)$ except for two diminishing terms. For large enough constant d, $-L(f,\lambda)+d$ is also nonnegative for all f, and by (4) it is therefore a "near positive submartingale" as in [15]. Since $\sum_{k=1}^{\infty}[\frac{(\log\log k)^{1/2}}{k^{3/2}}\tilde{b}+\frac{\tilde{a}}{k^2}]<\infty$, Exercise II-4 in [15] shows that $L(f(k),\lambda)$ converges almost surely.

Next, we need to show that $\lim_{k\to\infty}L(f(k),\lambda)=D(\lambda)$. We prove this by contradiction. Recall that the necessary and sufficient condition for $L(f,\lambda)=D(\lambda)$ is that $\sum_{n\in S}(U_n'(q_n)+\lambda_n)=\max_{S'\in c}\sum_{n\in S'}(U_n'(q_n)+\lambda_n)$, for all c,S such that $f_{c,S}>0$. Suppose $L(f(k),\lambda)$ does not converge to $D(\lambda)$. Then, there exists $\delta>0$, $\epsilon>0$ such that for all large enough k, there exist (c_k,S_k) so that $f_{c_k,S_k}>\delta$ and $\sum_{n\in S_k}(U_n'(q_n)+\lambda_n)<\max_{S'\in c}\sum_{n\in S'}(U_n'(q_n)+\lambda_n)-\epsilon$. Evaluating the term $E[\nabla L(f(k),\lambda)\Delta f(k)|f(k)]$ under this condition shows that $E[\nabla L(f(k),\lambda)\Delta f(k)|f(k)]>\frac{1}{k}\delta\epsilon$. Since there exists some constant K such that for all $k>K$, $\frac{\tilde{a}}{k^2}<\frac{1}{k}\delta\epsilon/2$, we obtain $E[L(f(k+1),\lambda)|f(k)]>L(f(k),\lambda)+\frac{1}{k}\delta\epsilon-\frac{\tilde{a}}{k^2}>L(f(k),\lambda)+\frac{1}{k}\delta\epsilon/2$. Since $\sum_{k=1}^{\infty}\frac{1}{k}=\infty$, we also have

$$\lim_{k\to\infty}E[L(f(k),\lambda)]=\infty,$$

which is a contradiction. Thus, $\lim_{k\to\infty}L(f(k),\lambda)=D(\lambda)$. □

Next we discuss how to solve the dual problem: $\min_\lambda D(\lambda)$. We use the subgradient method to solve it. We first find a subgradient for $D(\lambda)$.

LEMMA 2. *Let $v_n:=[\sum_{c,S:S\in c,n\in S}f_{c,S}^*-\underline{g}_n]$, where $[f_{c,S}^*]$ maximizes $L(f,\lambda)$. Then v is a subgradient of $D(\lambda)$.*

PROOF. Let λ' be an arbitrary vector. We have:

$$\begin{aligned}D(\lambda')=&\max_{f:\sum_{S\in c}f_{c,S}=p_c,\forall c}L(f,\lambda')\\ \geq&L(f^*,\lambda')=L(f^*,\lambda)+(\lambda'-\lambda)^{\mathrm{T}}vD(\lambda)\\ =&D(\lambda)+(\lambda'-\lambda)^{\mathrm{T}}v.\end{aligned}$$

Thus, v is a subgradient of $D(\lambda)$. □

The following theorem then follows from Theorem 8.9.2 in [1]:

THEOREM 2. *Let $\{\beta_k\}$ be a sequence of nonnegative numbers with $\sum_{k=1}^{\infty}\beta_k = \infty$ and $\lim_{k\to\infty}\beta_k = 0$. Update $\lambda(k)$ by:*

$$\lambda_n(k+1) = \{\lambda_n(k) - \beta_k[\sum_{c,S:S\in c, n\in S} f^*_{c,S}(k) - \underline{q}_n]\}^+,$$

where $[f^*_{c,S}(k)]$ maximizes $L(f,\lambda(k))$. Then,

$$\lim_{k\to\infty} D(\lambda(k)) = \min_{\lambda\geq 0} D(\lambda).$$

In practice, the max-weight scheduling policy may converge slowly. Thus, the values of $\lambda(k)$ should be updated at a slower time scale only after the max-weight scheduling policy converges. As $q_n(k) = \sum_{c,S:S\in c,n\in S} f^*_{c,S}(k)$ when the max-weight scheduling policy converges, the subgradient method can be further simplified as

$$\lambda_n(k+1) = \{\lambda_n(k) - \beta_k[q_n(k) - \underline{q}_n]\}^+ \quad (5)$$

when updating $\lambda(k)$. In practice, we will update the values of $\lambda(k)$ infrequently. A scheduling policy that jointly applies the max-weight scheduling policy and updates $\lambda(k)$ according to the subgradient method is utility-optimal. This policy is an on-line policy in the sense that it schedules clients and updates $\lambda(k+1)$ only based on $[U_n'(q_n(k))]$, $[q_n(k)]$ and $\lambda(k)$.

We close this section by discussing the case where α_k is a constant for all k. Since all the discussions in this section are based on the assumption that $\alpha_k = \frac{1}{k}$, the scheduling policy that uses the max-weight scheduling policy while updating $\lambda(k)$ may no longer be utility optimal when α_k is a constant since the system will be too sensitive to events in the current interval. For example, consider the extreme case where $\alpha_k = 1$ for all k. The total utility of the system at interval k, $\sum_{n\in S} U_n(q_n(k))$, solely depends on events in interval k and thus will never converge under any scheduling policy. However, in practice, the constant value of α_k is usually small. Under such a case, the scheduling policy that jointly applies the max-weight scheduling policy and updates $\lambda(k)$ according to the subgradient method still offers good performance. In Section 8, we will show by simulations that, when the constant value of α_k is around the order of 10^{-2}, the performance of the proposed policy is better than all other compared policies in all evaluated scenarios. We refer the reader to [12] for a a broader discussion of issues related to constant step sizes and vanishing step sizes and technical issues related to convergence analysis in the two cases.

6. A TRUTHFUL AUCTION DESIGN

We have proposed an on-line scheduling policy and proved that it is utility-optimal in Section 5. However, this policy requires knowledge of the utility functions of all clients. In practice, utility functions may be known only to their clients. A strategic client may hence improve its own utility by faking its utility function. In this section, we will propose an auction design that prevents clients from benefiting by faking their utility functions. In this auction, clients offer their bids for service in each time interval. The server selects a subset of clients to serve and charges them based on their bids. The decision of selecting clients and charging them is derived from an auction design similar to the VCG auction. We prove that this design restricts clients from faking their utility functions. We also show that the auction design schedules the same clients as those scheduled by the on-line scheduling policy introduced in Section 5. Thus, this auction is not only truthful but also utility-optimal.

6.1 Basic Mechanism and Truthful Property

We first describe the procedure and terminology of an auction. We then propose a VCG-based auction. At the beginning of each time interval k, every client n announces a bid $b_n(k)$ to the server. Based on the bids $[b_n(k)]$, along with the past history of the system and the current system state, the server schedules a subset S in the time interval k and charges each client n an amount $e_n(k+1)$ of money. Thus, an auction design can be specified by its decisions on selecting clients to schedule and charging them.

Recall that client n receives an utility that is equivalent to an amount $\frac{1}{\alpha_{k+1}}U_n(q_n(k+1))$ of money. The factor $\frac{1}{\alpha_{k+1}}$ is defined so as to equalize the importance of events in each interval k because whether client n is scheduled in interval k influences $q_n(k+1)$ by α_k. The *net utility* of a client can be defined as $\frac{1}{\alpha_{k+1}}U_n(q_n(k+1)) - e_n(k+1)$. We can also define the *marginal utility* of client n from being scheduled in the k^{th} time interval as follows:

DEFINITION 5. Let $q_n^+(k+1)$ and $q_n^-(k+1)$ be the service rates of client n if it is scheduled, and if it is not scheduled, in the k^{th} time interval, respectively. The *marginal utility* of client n in the k^{th} time interval is defined as $\frac{1}{\alpha_{k+1}}[U_n(q_n^+(k+1)) - U_n(q_n^-(k+1))]$.

Suppose the goal of every client is to selfishly maximize its own net utility $\frac{1}{\alpha_{k+1}}U_n(q_n(k+1)) - e_n(k+1)$, in each time interval k. An auction design is considered *truthful* if all clients bid their marginal utilities.

DEFINITION 6. An auction design is *truthful* if choosing

$$b_n(k) = \frac{1}{\alpha_{k+1}}[U_n(q_n^+(k+1)) - U_n(q_n^-(k+1))]$$

yields the highest net utility for client n in time interval k.

Now we propose a VCG-based auction. The server assigns a non-negative *discount*, $d_n(k)$, to each client n in time interval k. The values of discounts are announced before clients offer their bids and are thus not influenced by the bids of clients. After gathering the bids from clients, the server then schedules a feasible subset S that maximizes $\sum_{n\in S}(b_n(k)+d_n(k))$, with ties broken arbitrarily. The server does not charge anything to those clients that are not scheduled. For each scheduled client $m \in S$, the server charges it the minimum bid $e_m(k+1)$ it should have offered to be scheduled, specifically

$$\begin{aligned} e_m(k+1) = \quad & \max_{S':m\notin S'}[\textstyle\sum_{n\in S'}(b_n(k)+d_n(k))] \\ & - \textstyle\sum_{n\in S, n\neq m}(b_n(k)+d_n(k)) - d_m(k). \end{aligned} \quad (6)$$

This auction mechanism is usually referred to as a *weighted VCG* mechanism. The proof that such an auction mechanism is truthful can be found in Section 9.5.3 of [16].

THEOREM 3. *The proposed auction is truthful.*

6.2 Proof of Optimality

We now prove that the scheduling policy derived from the proposed auction design is consistent with the max-weight scheduling policy. Thus, this auction also achieves the maximum total long-term utility.

THEOREM 4. *Let $d_n(k) \equiv \lambda_n(k)$. Assume all clients bid their marginal utility. Then, there exists $\epsilon > 0$ such that for all $\alpha_k < \epsilon$, a feasible subset $S \in c(k)$ that maximizes $\sum_{n\in S}(b_n(k)+d_n(k))$ also maximizes $\sum_{n\in S}(U_n'(q_n(k))+\lambda_n(k))$ over all feasible subsets.*

PROOF. Let $M := \max_{S\in c(k)} \sum_{n\in S}(U_n'(k)+\lambda_n(k))$, and let $\mathbb{S}_M$ be the collection of all feasible subsets S with

$$\sum_{n\in S}(U_n'(k)+\lambda_n(k)) = M.$$

Also, let $M^- := \max_{S:S\in c(k),S\notin\mathbb{S}_M} \sum_{n\in S}(U_n'(k)+\lambda_n(k))$ and let $\delta := M - M^-$.

Recall that $q_n^+(k+1) = (1-\alpha_k)q_n(k)+\alpha_k$ and $q_n^-(k+1) = (1-\alpha_k)q_n(k)$. Since utility functions are infinitely differentiable, by using Taylor's series,

$$\begin{aligned} b_n(k) &= \frac{1}{\alpha_k}[U_n(q_n^+(k+1)) - U_n(q_n^-(k+1))] \\ &= U_n'((1-\alpha_k)q_n(k)) + O(\alpha_k). \end{aligned}$$

There exists some ϵ such that for all $\alpha_k < \epsilon$,

$$|b_n(k) - U_n'(q_n(k))| < \delta/2N.$$

Now we have that $|\sum_{n\in S}(b_n(k)+d_n(k)) - \sum_{n\in S}(U_n'(k)+\lambda_n(k))| < \frac{\delta}{2}$ for all subsets S. Thus, for all feasible subsets $S \in \mathbb{S}_M$ and $S' \notin \mathbb{S}_M$,

$$\sum_{n\in S}(b_n(k)+d_n(k)) > \sum_{n\in S'}(b_n(k)+d_n(k)).$$

Therefore, a feasible subset that maximizes $\sum_{n\in S}(b_n(k)+d_n(k))$ also maximizes $\sum_{n\in S}(U_n'(q_n(k))+\lambda_n(k))$ □

We have proved that the max-weight scheduling policy and the auction mechanism schedule the same set of clients if $\alpha_k < \epsilon$. Thus, when α_k is $\frac{1}{k}$, since $\alpha_k \to 0$ as $k \to \infty$, the two policies will converge eventually. For the case where α_k is set to a constant for all k, the behavior of the auction mechanism is still approximately the same as that of the max-weight policy if the constant value of α_k is small. We again refer the reader to [12] for a discussion of such issues. In Section 8, we will show by simulation that when the constant value of α_k is around the order of 10^{-2}, the performances of the two policies are indistinguishable.

In Section 5.2, we have shown that by applying the max-weight scheduling policy and updating $[\lambda_n(k)]$ according to (5), the total long-term utility is maximized. Thus, we can also maximize the total long-term utility by applying the proposed auction and setting discounts $[d_n(k)] \equiv [\lambda_n(k)]$.

6.3 Remarks on Implementation

In practice, we implement a protocol that jointly applies the proposed auction design and updates discounts by $[d_n(k)] = [\lambda_n(k)]$. To reduce overhead, all clients announce their utility functions to the server upon joining the system. In each time interval, the server computes their bids as

$$b_n(k) = \frac{1}{\alpha_{k+1}}[U_n(q_n^+(k+1)) - U_n(q_n^-(k+1))].$$

The server then schedules the feasible subset $S \in c(k)$ that maximizes $\sum_{n\in S}[b_n(k)+d_n(k)]$ and charges all clients $n \in S$ according to (6). As a consequence of Theorem 3, a client that aims to greedily maximize its own net utility in each time interval k would not benefit by lying about its utility function. Also, this protocol is utility-optimal.

In addition to being truthful and utility-optimal, this protocol also provides incentives for servers to offer service by charging clients. For example, consider the scenario where a TV broadcast company holds several licenses for channels. Even though the company may not fully utilize its channels, it would not be willing to allocate unused bandwidth to unaffiliated users unless doing so can increase its own revenue. Thus, generating revenues for the server is also an important property for a protocol.

7. ALGORITHMS FOR SPECIFIC APPLICATIONS

We have shown that the protocol described in Section 6.3 is both truthful and utility-optimal. Given the bids $[b_n(k)]$ from clients, the protocol needs to select a subset $S \in c(k)$ that maximizes $\sum_{n\in S}[b_n(k)+d_n(k)]$ and charge them. In this section, we discuss how to design algorithms for scheduling and charging explicitly for each of the three applications discussed in Section 4.

7.1 Delay-Constrained Wireless Networks with Rate Adaptation

We consider first the scenario described in Section 4.1. There are N wireless clients and one AP. Each time interval contains T time slots. At the beginning of each time interval, there is one arrived packet at the AP for each client n. The packet for client n is to be delivered before the τ_n^{th} time slot, with $\tau_n \le T$, or else the packet expires and is dropped from the system. The transmission time for client n under system state c is $t_{c,n}$.

A feasible subset $S \in c$ can be described as an ordered subset $S = \{s_1, s_2, \ldots, s_m\}$, such that transmitting packets for clients in S according to the order will meet their respective delay bounds; that is, $\sum_{j=1}^{i} t_{c,s_j} \le \tau_{s_i}$, for all $1 \le j \le m$. To find the feasible subset $S \in c(k)$ that maximizes $\sum_{n\in S}(b_n(k)+d_n(k))$ in the k^{th} time interval, we can assign a value $v_n(k) := b_n(k)+d_n(k)$ to each client k. Now, if we have $\tau_n \equiv T$, for all n, then this is simply a knapsack problem. To deal with the case when different clients may require different delay bounds, we note that for any feasible ordered subset S, reordering clients in S in ascending order of their delay bounds, i.e., serving clients in an "earliest deadline first" fashion, is also feasible. Thus, an analog to the modified knapsack algorithm described in [7] finds a feasible subset $S \in c(k)$ that maximizes $\sum_{n\in S}(b_n(k)+d_n(k))$ in the k^{th} time interval. The complete algorithm is shown in Algorithm 1. The complexity of this algorithm is $O(NT)$. To compute the charge for a client $n \in S$, we need to determine $\max_{S':S'\in c(k),n\notin S'}(b_n(k)+d_n(k))$, which can be obtained by eliminating client n and rerunning Algorithm 1. Thus, the complexity of computing charges for all scheduled clients is $O(N^2T)$.

7.2 Mobile Cellular Networks

Consider the scenario described in Section 4.2 with one

Algorithm 1 Delay-Constrained Networks

for $n = 1$ to N **do**
  $v_n(k) \leftarrow b_n(k) + d_n(k)$
Sort clients such that $\tau_1 \leq \tau_2 \leq \cdots \leq \tau_N$
$S[0,0] \leftarrow \phi$
$M[0,0] \leftarrow 0$
for $n = 1$ to N **do**
  for $t = 1$ to T **do**
    if $t > \tau_n$ **then**
      $M[n,t] \leftarrow M[n,t-1]$
      $S[n,t] \leftarrow S[n,t-1]$
    else if $v_n(k) + M[n-1, t-t_{c(k),n}] > M[n-1,t]$ **then**
      $M[n,t] \leftarrow v_n(k) + M[n-1, t-t_{c(k),n}]$
      $S[n,t] \leftarrow S[n-1, t-t_{c(k),n}] + \{n\}$
    else
      $M[n,t] \leftarrow M[n-1,t]$
      $S[n,t] = S[n-1,t]$
$\max_{S:S\in c(k)} \sum_{n\in S}(b_n(k)+d_n(k)) \leftarrow M(N,T)$
schedule according to $S[N,T]$

base station holding $\mathbb{C}$ channels and N mobile users. The links between the users and the base station can either be ON or OFF, and the base station can only schedule transmissions to users with ON links. Recall that a subset S is feasible under system state c if every client n in S has an ON link, and the size of S is smaller or equal to the number of channels, $\mathbb{C}$. Thus, to implement the protocol, we can simply sort all clients with ON links in descending order of $b_n(k) + d_n(k)$, and schedule the first $\mathbb{C}$ clients. Also, let client m be the $(\mathbb{C}+1)^{th}$ client with an ON link. To be scheduled, a scheduled client n would have to outbid client m, that is, $b_n(k) + d_n(k) \geq b_m(k) + d_m(k)$. Thus, the price paid by each scheduled client n is $b_m(k) + d_m(k) - d_n(k)$.

7.3 Dynamic Spectrum Allocation

Finally, we discuss the scenario of dynamic spectrum allocation. Suppose there is a primary user holding licenses to several channels , and there are N secondary users. The interference relations between secondary users are represented by a conflict graph $G = \{V, E\}$, where V is the set of all secondary users, and two users correspond to an edge in G if they interfere with each other. Suppose the primary user is going to allocate $\mathbb{C}(k)$ unused channels in the k^{th} time interval. A feasible subset of secondary users can be represented as a coloring by $\mathbb{C}(k)$ colors on some nodes in V, with the restriction that any two colored nodes with the same color cannot share an edge in G. To find a subset $S \in c(k)$ with the maximum $\sum_{n\in S}(b_n(k) + d_n(k))$, we can associate a value $v(k) = b_n(k) + d_n(k)$ to each node in V and find the maximum-weight coloring with $\mathbb{C}(k)$ colors. In particular, when $\mathbb{C}(k) = 1$, this is equivalent to finding the maximum-weight independent set. While finding the maximum-weight independent set is NP-hard, there exist heuristics. Also, for a mid-sized network with about 20 secondary users, the computational overhead for finding an optimal schedule is reasonably small.

8. SIMULATION RESULTS

We now present simulation results for the three discussed applications: delay-constrained wireless networks with rate adaptation, mobile cellular networks, and dynamic spectrum allocation. In each of the three applications, we evaluate the max-weight scheduling policy, the VCG-based auction mechanism, and a policy that randomly orders all clients and greedily schedules a maximal feasible subset in each time interval. In addition, we also compare our policies against a state-of-art policy in each of the three applications.

In all applications, the utility function of client n is set to be $U_n(q_n) := w_n \frac{q_n^{a_n}-1}{a_n}$, where w_n is a positive integer and $a_n \in (0,1)$. We set α_k to a small positive constant α, for all k. We update $\lambda_n(k)$ and $d_n(k)$ every T time intervals. To ensure our policies nearly converge within T time intervals, the value of T is chosen so that $(1-\alpha)^T$ is small. When updating $\lambda_n(k)$ and $d_n(k)$, we also set β_k to a constant β. The choice of β involves a tradeoff between two factors. On the one hand, if β is too small, it takes a long time for $\lambda_n(k)$ and $d_n(k)$ to be incremented to a large enough value. On the other hand, if β is too big, then $\lambda_n(k)$ and $d_n(k)$ may dominate the max-weight scheduling policy and the VCG-based auction mechanism. As a result, $\lambda_n(k)$ and $d_n(k)$ may fluctuate instead of converging to an optimal value. In practice, we choose β so that $\beta \underline{q}_n$ is about one-tenth the marginal utilities of clients.

In all simulations, we evaluate two metrics: the total utility of all clients, and the total penalty, $\sum_n [\underline{q}_n - q_n(k)]^+$, defined as the sum of shortfalls between the required throughputs and the actual throughputs of clients. All results presented are the average over 20 runs.

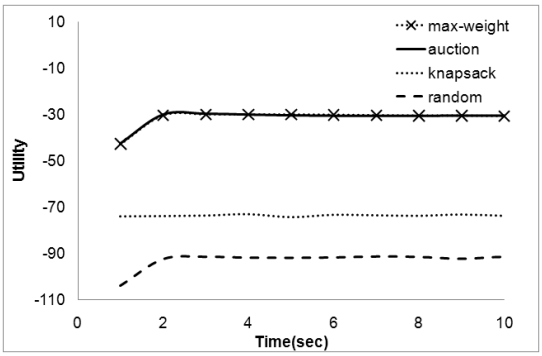
8.1 Delay-Constrained Wireless Networks with Rate Adaptation

In this scenario, we compare our policies against the modified-knapsack policy proposed in [7], which is feasibility-optimal in the sense that it satisfies the minimum service requirements for any sets of clients as long as they are feasible.

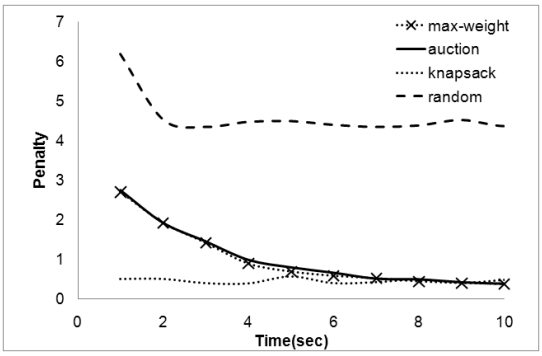
We use a similar set-up as the one in [7] for this scenario. Suppose that there are 45 clients that generate VoIP traffic. Each client generates one packet of size 160 Bytes every 20 ms, which is the length of a time interval. This results in a 64 kbit/sec rate. The highest data rate that can be supported by the link between the AP and client n alternates between 11 Mb/sec and 5.5 Mb/sec. The times needed for a transmission under the two data rates, including header and ACK, are 480 μs and 610 μs, respectively. Thus, by setting the duration of a time slot to 160 μs, a transmission takes 3 time slots under 11 Mb/sec, and 4 time slots under 5.5 Mb/sec. A time interval contains 125 time slots.

The utility function of client n is defined by $w_n = 3 + (n \mod 3)$ and $a_n = 0.05 + 2n$. The minimum service requirement of client n is $0.5+0.01(20n \mod 300)$ packets per interval. The delay bound of client n equals the length of a time interval if n is odd, and equals two-third of the length of a time interval if n is even. Further, we set $\alpha = 0.05$, $\beta = 1$, and $T = 50$.

Simulation results are shown in Figure 1. The max-weight scheduling policy and the VCG-based auction mechanism achieve the highest utility and near-zero penalty. The penalties resulting from the two policies converge to zero slower than the modified-knapsack policy. This is because it takes some time for the two policies to build up their discounts. On the other hand, the modified-knapsack policy has worse utility than the

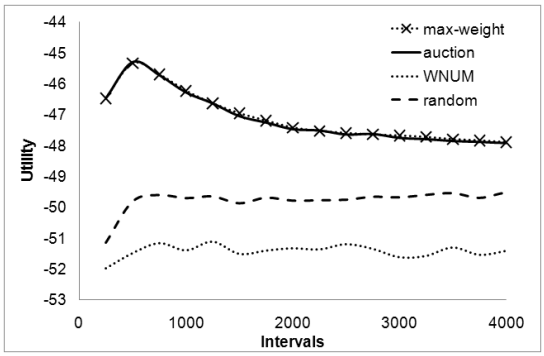


(a) Utility

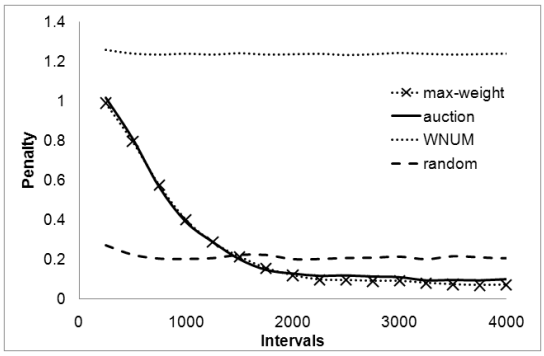


(b) Penalty

Figure 1: Delay-Constrained Networks



(a) Utility



(b) Penalty

Figure 2: Mobile Cellular Networks

two proposed policies because it only concerns itself with satisfying the minimum service requirements of clients and does not consider utilities.

8.2 Mobile Cellular Networks

We consider a system with one base station with three non-interfering channels, and 20 users. The utility function of the n^{th} client is defined by $w_n = 1 + (n \mod 3)$ and $a_n = 0.2+0.1(n \mod 7)$. A time interval is the time for one packet transmission. The minimum service requirement of client n is set to be $\underline{q}_n = 0.05(n \mod 5)$ packets per interval. The probability that the link between client n and the base station is ON is $0.6 + 0.02(n \mod 10)$. Finally, we set $\alpha = 0.01$, $T = 250$, and $\beta = 10$.

In addition to the three policies discussed above, we also evaluate the WNUM policy proposed in [17], which considers and maximizes utilities on a per-interval base. Simulation results are shown in Figure 2. The max-weight scheduling policy and the VCG-based auction mechanism both achieve the highest utility and near-zero penalty, suggesting that they not only satisfy the minimum service requirements of clients but are also utility-optimal. On the other hand, the performance of the WNUM policy is even worse than the random policy. This shows that, as far as long-term average performance is concerned, a policy that optimizes total utility based on per-interval performance is not adequate.

8.3 Dynamic Spectrum Allocation

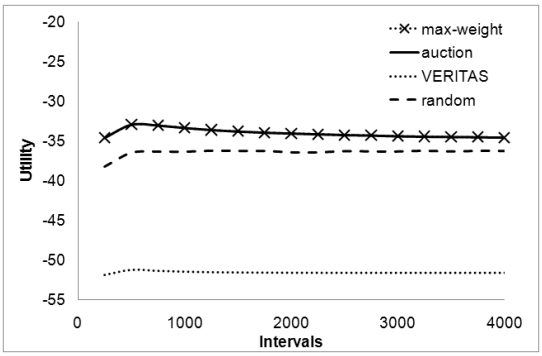
Finally, we present the simulation results for dynamic spectrum allocation. We compare our policies against VERITAS as proposed in [22]. VERITAS is developed for the scenario where the primary only makes allocation decisions once, or at most very infrequently.

We use a similar setting as that in [22]. We assume there are 20 secondary users, randomly spaced in a 1×1 square. Two users interfere with each other if their distance is smaller than 0.3. The average node degrees of the resulting conflict graph in each simulation range from 2.9 to 6.5. We assume that the primary user always has a channel to allocate. The utility function and minimum service requirement of client n are given by $w_n = 1+(n \mod 3)$, $a_n = 0.2+0.1(n \mod 7)$, and $\underline{q}_n = 0.05(n \mod 8)$. Finally, we set $\alpha = 0.01$, $\beta = 5$, and $T = 250$.

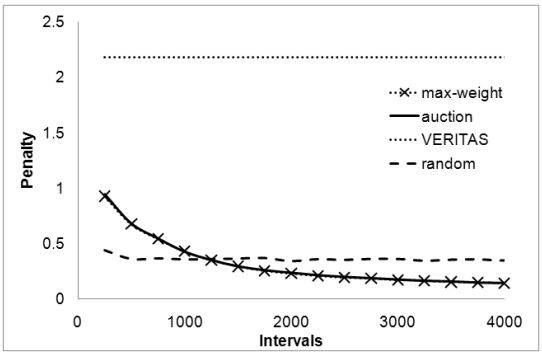
Simulation results are shown in Figure 3. As in the previous simulations, both the max-weight scheduling policy and the VCG-based auction have the highest utilities and near-zero penalties. Also, it can be shown that VERITAS has poor performance under this setting. This suggests that a protocol developed for spectrum allocation in a one-shot fashion is not applicable to scenarios where the primary user allocates spectrum frequently.

9. CONCLUSIONS

We have studied the problem of network utility maximization in a time-varying wireless networks where packets have delay constraints. We have considered a system with time-varying states and clients with minimum service rate requirements. We have proposed a max-weight scheduling policy and proved that it is utility-optimal. Further, to ensure that clients do not benefit by lying about their utility functions, we have developed a VCG-based auction mechanism where the server schedules and charges clients according to their bids.



(a) Utility



(b) Penalty

Figure 3: Dynamic Spectrum Allocation

We have shown that this auction mechanism is not only truthful but also consistent with the utility-optimal scheduling policy. Thus, the auction mechanism is also utility-optimal. We have discussed designing specific algorithms for three applications that are accommodated by our system model: delay-constrained wireless networks with rate adaptation, mobile cellular networks, and dynamic spectrum allocation. Simulation results show that both the max-weight scheduling policy and the VCG-based auction outperform state-of-the-art policies for the three applications.

While this work can be applied to a wide range of applications, we have mainly focused on discussing delay-constrained wireless networks with rate adaptation. The discussions on other applications are limited to simple scenarios. How to extend this work to other applications with more complicated and realistic settings is an interesting problem for future research.

10. REFERENCES

[1] BAZARAA, M., SHERALI, H., AND SHETTY, C. *Nonlinear programming: theory and algorithms*. Wiley-Interscience, 2006.

[2] CHOW, Y., AND H.TEICHER. *Probability theory, independence, interchangeability, martingales*. Springer-Verlag, 1988.

[3] CLARKE, E. Multipart pricing of public goods. *Public Choice 11*, 1 (Sep. 1971), 17–33.

[4] GANDHI, S., BURAGOHAIN, C., CAO, L., ZHENG, H., AND SURI, S. A general framework for wireless spectrum auctions. In *Proc. of IEEE DySPAN* (2007).

[5] GROVES, T. Incentives in teams. *Econometrica 41*, 4 (Jul. 1973), 617–631.

[6] HOU, I.-H., BORKAR, V., AND KUMAR, P. A theory of QoS in wireless. In *Proc. of IEEE INFOCOM* (2009), pp. 486–494.

[7] HOU, I.-H., AND KUMAR, P. Scheduling heterogeneous real-time traffic over fading wireless channels. In *Proc. of IEEE INFOCOM* (2010).

[8] HOU, I.-H., AND KUMAR, P. Utility maximization for delay constrained qos in wireless. In *Proc. of IEEE INFOCOM* (2010).

[9] JIA, J., ZHANG, Q., ZHANG, Q., AND LIU, M. Revenue generation for truthful spectrum auction in dynamic spectrum access. In *Proc. of ACM MobiHoc* (2009), pp. 3–12.

[10] KELLY, F. Charging and rate control for elastic traffic. *Euro. Trans. Telecomms 8* (1997), 33–37.

[11] KELLY, F., MAULLOO, A., AND TAN, D. Rate control in communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society 49* (1998), 237–252.

[12] KISHNER, H. J., AND YIN, G. G. *Stochastic Approximation Algorithms and Applications*. Springer-Verlag, 1997.

[13] LIN, X., AND SHROFF, N. Joint rate control and scheduling in multihop wireless networks. In *Proc. of IEEE CDC* (2004), pp. 1484–1489.

[14] LIU, X., CHONG, E., AND SHROFF, N. Opportunisitic transmission scheduling with resource-sharing constraints in wrieless networks. *J-SAC 19*, 10 (Oct. 2001), 2053–2064.

[15] NEVEU, J. *Discrete parameter maringales*. North-Holland, 1975.

[16] NISAN, N., ROUGHGARDEN, T., TARDOS, E., AND VAZIRANI, V. *Algorithmic game theory*. Cambridge University Press, 2007.

[17] O'NEILL, D., GOLDSMITH, A., AND BOYD, S. Wireless network utility maximization. In *Proc. of IEEE Milcom* (2008).

[18] RAGHUNATHAN, V., BORKAR, V., CAO, M., AND KUMAR, P. Index policies for real-time multicast scheduling for wireless broadcast systems. In *Proc. of IEEE INFOCOM* (2008).

[19] SHAKKOTTAI, S., AND SRIKANT, R. Scheduling real-time traffic with deadlines over a wireless channel. *Wireless Networks 8*, 1 (Jan. 2002).

[20] VICKERY, W. Counterspeculation, auctions and competitive sealed tenders. *J. of Finance 16* (1961), 8–37.

[21] YI, Y., AND CHIANG, M. Stochastic network utility maximization. *Euro. Trans. Telecomms* (2008), 1 – 22.

[22] ZHOU, X., GANDHI, S., SURI, S., AND ZHENG, H. eBay in the sky: strategy-proof wireless spectrum auctions. In *Proc. of ACM MobiCom* (2008), pp. 2–13.