On Adaptive Transmission for Energy Efficiency in Wireless Data Networks

Elif Uysal-Biyikoglu, *Member, IEEE,* and Abbas El Gamal*, Fellow, IEEE*

***Abstract*—This paper investigates the problem of energy-efficient transmission of data packets in a wireless network by jointly adapting to backlog and channel condition. Specifically, we consider minimum-energy scheduling problems over multiple-access channels, broadcast channels, and channels with fading, when packets of all users need to be transmitted before a deadline T. Earlier work has considered a similar setup and demonstrated significant transmission energy saving by adapting to backlog for channels that are time invariant and when transmission is restricted to time-division. For concreteness, throughout the paper, rates and powers corresponding to optimal coding over discrete-time additive white Gaussian noise (AWGN) channels are assumed. The results, however, hold for more general channels and coding schemes where the total transmitted power is convex in the transmission rates. The offline scheduling problems for all the channels considered are shown to reduce to convex optimization problems with linear constraints. An iterative algorithm, referred to as FlowRight, that finds optimal offline schedules is presented. A heuristic online algorithm that we call look-ahead water-filling, which jointly adapts to both channel fading state and backlog is described. By the use of a small buffer which introduces an almost fixed delay, this algorithm achieves a considerable reduction in energy relative to water filling solely on channel states.**

***Index Terms*—Adaptive transmission, broadcast, energy-efficient transmission, iterative algorithm, multiple-access, power control, scheduling, time-division, wireless networks.**

I. Introduction

ENERGY efficiency in computation, signal processing, and information transmission has been a topic of increasing interest motivated by applications in mobile computing, wireless data, *ad hoc* and sensor networks. Among the various problems related to energy efficiency, optimizing transmission power is of special importance, since the scalability of a wireless network is fundamentally limited by transmission power.

Power control for the purpose of maximizing rate of reliable communication in the presence of interference and fading, both characteristic of wireless networks, is well studied. Optimal rate and power adaptation schemes for the single-user and multiple-access fading channels have been developed (e.g., [11], [15], [20]), and approximated by practical adaptive coding/modulation schemes ([12], [14]). These schemes, however, assume a continuous stream of data. If the average data generation rate is known, these schemes would keep transmission rate close to the data arrival rate and be energy efficient. However, in many wireless data applications, the rate at which data is generated and needs to be transmitted is unknown and varies with time (e.g., wireless web sessions or a sensor network where data gets generated at random times at each node). Ignoring this variability and adapting solely to the channel can be inefficient in terms of transmit power and bandwidth.

To understand how inefficiency may arise, consider the following generic data communication situation: the transmitter and receiver engage in a session which may be a video conference or a web session, or may alternate between the two. Different types of applications generate data at different rates, and the data packets are collected in the transmitter's buffer to be sent to the receiver. Let the rate at which packets arrive into the transmitter's buffer at time t be $\lambda(t)$ packets per second. These packets are transmitted to the receiver at a rate $\mu(t)$ packets per second. Now, assume we set $\mu(t) = \mu$, a constant that is large enough, say, for a high-rate streaming video session. When the required rate drops, for example because the user switches to a lower rate web session where $\lambda(t) \ll \mu$, the transmitter will idle a significant fraction of time and unnecessarily transmit at a high rate for the rest of the time.

Schemes that adapt solely to the channel state can maximize the throughput for a given energy constraint; but since they cannot track the value of $\lambda(t)$, they do not have control over delay. In order to guarantee finite average delay, they need to be set for the largest possible value of $\lambda(t)$, which causes them to be energy inefficient.

In order to achieve better performance in terms of energy and delay, one must also adapt to the variation in the data rate. To our knowledge, the earliest appearance of joint queue state/channel state adaptive power control is in [6]. A more comprehensive treatment appears in [2], where the objective is to obtain the optimal power–delay tradeoff curve and develop algorithms that minimize power while keeping the buffer size below a certain level. In [23], the minimum-delay power control problem (under a power constraint) is posed. It is shown that under the two extremes of fast fading and slow fading, the optimal power control policy goes to the two extremes of water-filling in time [20], and channel inversion, respectively. Médard *et al.* [16] showed that the capacity region of the time-slotted ALOHA system with power-constrained users is the same as the capacity region of the multiple-access channel.

Manuscript received June 26, 2002; revised June 29, 2004. The work was supported in part under SNRC Multilayer Mobile Networking Project and by a Stanford Graduate Fellowship.

E. Uysal-Biyikoglu is with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: elif@mit.edu).

A. El Gamal is with the Information Systems Laboratory, Department of Electrical Engineering, Stanford University, Stanford, CA 94305 USA (e-mail: abbas@isl.stanford.edu).

Communicated by L. Tassiulas, Associate Editor for Communication Networks.

Digital Object Identifier 10.1109/TIT.2004.838355

Delay-optimal scheduling of data packets from different queues is studied in the context of satellite transmission systems in [17], [9]. In [17], it is shown, using Lyapunov theory, that the throughput-optimal schedule for a system of several queues is a maximum weight matching where the weight of each queue is the backlog multiplied by the rate at which it can transmit under the current channel conditions, for a certain power constraint. A related study, [9], considers optimal energy allocation and admission control for communication satellites in earth orbit. The goal is to choose which data requests to serve at a given time, in order to maximize expected total reward while the energy that can be stored is finite and is replenished at regular intervals. An optimal policy is obtained using dynamic programming.

In this paper, we build a simple framework that captures the discrete nature of packet arrivals, on which we are able to study scheduling (assigning rates and transmission times to users) in a multiuser setting with respect to energy and delay. The insights obtained are then used to propose good scheduling algorithms. The work here extends [19] and [8], where it was shown that adaptation to the packet arrival process can result in significant transmission energy savings. In [19] and [8] (see Section II for a review), the channel was assumed to be time invariant such that the energy as a function of transmission duration does not depend on when a packet is transmitted. Here, we consider the more realistic wireless communication scenario where the channels (hence, the energy functions) are time varying due to interference and fading. Specifically, we consider minimum-energy scheduling problems over multiple-access channels, broadcast channels, and channels with fading. For concreteness, throughout the paper, we assume rates and powers corresponding to optimal coding over discrete-time additive white Gaussian noise (AWGN) channels. Our results, however, hold for more general channels and coding schemes where the total transmitted power is convex in the transmission rates.

The key results in the paper are: i) showing that for each of these channels, offline scheduling reduces to a convex optimization problem with linear constraints, ii) devising an algorithm, FlowRight, that iteratively finds the optimal offline schedules (Section III-B), and iii) devising a heuristic online algorithm, look-ahead water-filling, that by jointly adapting to both channel fading state and backlog achieves energy efficiency close to the optimal offline schedule (Section IV-B). The following example illustrates the potential energy saving achieved by such joint adaptation.

Example

Data packets of size $B = 10^3$ bits arrive at a transmitter's buffer at a rate known to be at most $\lambda_{\max} = 1$ packets/time unit.[1] The packets are transmitted over an AWGN channel with noise power σ^2 and slow ergodic fading, where the transmitter and the receiver have perfect channel state information (CSI). Fig. 1 shows the channel gain and the arrival instants, and the rate (bits per transmission) used by three scheduling algorithms. The first is water-filling in time [11], which adapts optimally to the channel to achieve a time average rate of $\lambda_{\max}$ packets/time unit. The second is the look-ahead water-filling algorithm. The third is the optimal offline algorithm, which has perfect information of all future packet arrival times and channel states at time 0. Notice how the water-filling algorithm uses a high rate and idles for a major fraction of the time, while the second online algorithm, which adapts its rate to its current backlog, spreads the rate more uniformly across time, imitating the optimal offline schedule. As a result, the online algorithm reduces the average transmission energy per packet by a factor of 100, to a value which is almost within a factor of 1.5 of the offline optimal.

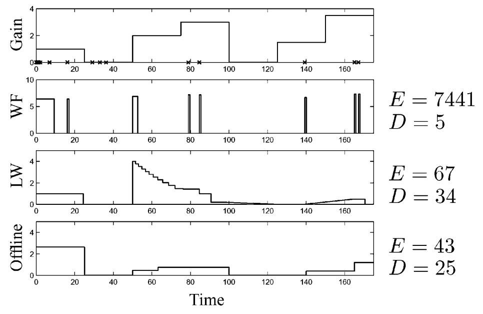

Fig. 1. The top plot shows a sequence of packet arrivals ("$\times$") and channel gains. The lower three plots show the instantaneous rates used by the online algorithms water-filling (WF) and look-ahead water-filling (LW), and the optimal offline schedule, respectively, as they run on this sequence of packet arrivals. The average energy per packet values (E) are normalized for a noise power of unity. Average delay (in time units) per packet (D) is also given.

The rest of the paper is organized as follows. First we consider transmission schedules for the multiple-access channel. The MoveRight algorithm (proposed in [8]) can find the best *time-division* solution to the offline multiple-access problem. However, to find the *optimal* solution, we need to consider general multiple-access coding schemes, where the users can simultaneously transmit. In the following section, we define the multiple-access offline scheduling problem and show that it can be cast as a convex optimization problem with linear constraints. In Section III-B, we present FlowRight, which solves this problem. In Section III-E, we present an online scheduling algorithm that uses a look-ahead buffer and compare its performance in terms of energy and average delay per packet to time-division and to the optimal offline schedule. FlowRight is also shown to optimally solve the offline scheduling problem for the broadcast channel. In Section IV, we turn to channels with fading, and show that FlowRight can optimally solve the offline scheduling problem in a slow-fading channel with perfect CSI at the transmitter and the receiver. We then present the look-ahead water-filling algorithm, which adapts jointly to the channel state and data arrival rate, and observe that it is significantly more energy-efficient than the well-known water-filling algorithm that adapts solely to the channel state. The look-ahead water-filling algorithm is shown (through simulations) to also perform closely to the offline optimal benchmark provided by FlowRight. The results are shown to generalize to multiple-access and broadcast channels. We also show that scheduling in a fast-fading channel reduces to the single-user problem of [19].

[1]The arrivals in this example are a realization of a bursty arrival process at average rate $\lambda = 0.2$.

Since this paper extends the work in [22] and [8], in the following section we first briefly review the key results in these papers.

II. Previous Work

The work in [22] and [8] is based on the observation that with many channel coding schemes, the energy required to transmit a packet over τ units of time, $w(\tau)$, is monotonically decreasing and strictly convex in τ. The minimum-energy packet scheduling problem for a single transmitter–receiver pair in a wireless network is formulated in [19]. The setup is as follows. Suppose that m packets arrive at the transmitter's buffer in the interval $[0,T)$ at times $0=t_1,t_2,\dots,t_m<T$. The node is required to transmit all m packets within the interval $[0,T]$. [2] The question is, how should the packet transmissions be scheduled to minimize the total energy required to transmit the packets. If we let τ_i be the transmission time for packet i, $1\le i\le m$, the offline version of the problem can be stated as follows.

Problem 1: Single-Transmitter Single-Receiver Offline Scheduling [19]: Given a vector of packet arrival times $\{t_i, i=1,\dots,m\}$, where $t_1=0$, $t_i<t_{i+1}$, and $t_m<T$, and an energy function $w(\tau)$ that is strictly monotonically decreasing and convex, find a schedule $\{\tau_i\}_{i=1}^m$ so as to minimize the total transmission energy: $\sum_{i=1}^m w(\tau_i)$ subject to causality[3] and deadline constraints.

Note that this is a convex optimization problem with linear constraints. The following explicit solution was found in [19].

Let ξ_i's be the packet inter-arrival times. Define $k_0^*=0$, and

$$k_l^* = \arg\max_{k_{l-1}^*+1\le k\le m}\left\{\frac{1}{k-k_{l-1}^*}\sum_{i=k_{l-1}^*+1}^{k}\xi_i\right\},\quad l=1,2,\dots,\min\{l: k_l^*=m\}.$$

Note that k_1^* is the index at which the running average of the ξ_i's is maximized for the first time. We set $\tau_1^*=\tau_2^*=\cdots=\tau_{k_1^*}^*$ to this average value $\frac{1}{k_1^*}\sum_{i=1}^{k_1^*}\xi_i$. Next, we set

$$\tau_{k_1^*+1}^*=\tau_{k_1^*+2}^*=\cdots=\tau_{k_2^*}^*=\frac{1}{k_2^*-k_1^*}\sum_{i=k_1^*+1}^{k_2^*}\xi_i. \tag{1}$$

In general, τ_j^* is set to

$$\tau_j^*=\frac{1}{k_{(l+1)}^*-k_l^*}\sum_{i=k_l^*+1}^{k_{(l+1)}^*}\xi_i \tag{2}$$

where l is the largest integer such that $k_l^*\le j\le k_{(l+1)}^*$. This is the optimal schedule. Note that the schedule contains bands of equal transmission times between breakpoints at positions k_l. The fact that the optimal solution is in the form of bands of equal transmission durations is quite intuitive. Consider the following example. Suppose all m packets were available at time $t=0$. In this case, it would be optimal to set $\tau_i=T/m$ for $1\le i\le m$ (as a consequence of the monotonicity and convexity of the energy function $w(\cdot)$.) Now, suppose the mth packet arrives at time $T-\delta$, where $\delta<T/m$, while all previous packets arrive at $t=0$. Setting $\tau_m=T/m$ would violate causality (the mth packet would be starting transmission *before* it arrived.) Clearly, the solution is to set $\tau_m=\delta$, while

$$\tau_1=\tau_2=\cdots=\tau_{m-1}=(T-\delta)/(m-1).$$

As in this simple example, the optimal algorithm tries to equate arrival times and make them as large as possible, within the constraints of causality. This solution was further explored in [22], [18] and the formulation was combined with other constraints.

In [8], the minimum energy scheduling problem for a multiple-user channel, e.g., uplink and downlink, involving several transmitters and receivers where time-division is used is investigated. The goal is to minimize the total energy for all users, and this results in the setup being identical to that of the previous problem except that packets can have different energy functions. Again, energy functions are convex and decreasing in the transmission duration, and this is essentially all that is assumed about the channel, transmitters, and receivers. The offline time-division scheduling problem is formulated as follows.

Problem 2: Multiple-User Offline Time-Division Scheduling [8]: Given a vector of packet arrival times $\{t_i, i=1,\dots,m\}$, where $t_1=0$, $t_i<t_{i+1}$, and $t_m<T$, and energy functions $w_i(\tau)$ that are strictly monotonically decreasing and convex, find a schedule that minimizes the total transmission energy: $\sum_{i=1}^m w_i(\tau_i)$ subject to causality and deadline constraints.

This is also a convex optimization problem but does not in general admit a simple closed-form solution. By exploiting the special features of the problem, an algorithm, MoveRight, which finds the global optimal schedule efficiently, is developed. MoveRight iteratively moves the start times of packet transmissions one at a time, so that each move locally optimizes the energy function. The algorithm was shown to solve other scheduling problems, such as when packets have individual deadlines, and when the transmit buffer is finite. MoveRight also leads to an online algorithm that uses a simple look-ahead buffer. The transmitter buffers the packets for a specified length of time L (the look-ahead window). At the end of the look-ahead window, the packets in the buffer are scheduled using a faster version of the MoveRight algorithm for transmission from L to $2L$. Meanwhile, the arrivals from L to $2L$ are buffered, to be transmitted in the following time window. Hence, at the expense of incurring a delay of $\approx L$, packets are scheduled optimally. The average energy per packet given by the look-ahead algorithm was shown through simulations to be quite close to that of the offline optimal schedule, using only a small look-ahead buffer.

Throughout the paper, the well-known terms "time-division" and "multiple-access" will be used to make the following distinction: In time-division, packets do not overlap in time, whereas in multiple-access scheduling, users' packets interfere

[2]The imposition of a strict deadline T, by which all transmissions had to terminate, was intended to capture several realistic wireless scenarios (see [22] for further details).

[3]In our setting, causality corresponds to the obvious constraint that no packet's transmission can start before its arrival time.

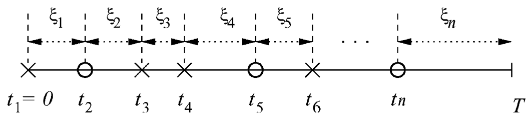

Fig. 2. Packet arrivals in $[0, T)$.

with each other but can, with appropriate multiuser coding and decoding, still be resolved at the receiver.

III. Scheduling for the Multiple-Access Channel

Consider the discrete-time AWGN multiple-access channel with K transmitters and a single receiver. Data packets are generated at each transmitter's buffer at arbitrary times t_i, in the interval $[0, T)$. Fig. 2 shows an example sequence of packet arrival times for two users, where packet arrival times of users 1 and 2 are marked by crosses and circles, respectively. These packets must be transmitted reliably to the receiver in the time interval $(0, T]$. The received signal at time k is

$$Y[k] = \sum_{i=1}^{K} \sqrt{s_i[k]} X_i[k] + Z[k] \tag{3}$$

where $X_i[k]$ is user i's signal, and $Z[1], Z[2], \ldots$ are independent and identically distributed (i.i.d.) zero-mean Gaussian noise with variance σ^2. For now, we assume the s_i's to be constant in time. Later we consider $s_i[k]$'s that vary with k to model frequency-flat fading.

It is well known that for the AWGN multiple-access channel the set of feasible average *received* powers $\{P_i\}_{i=1}^{K}$, for a given set of rates $\{r_i\}_{i=1}^{K}$, is given by (see [20])

$$\sum_{i \in S} P_i \geq \sigma^2 \left(2^{2\left(\sum_{i\in S} r_i\right)} - 1\right)$$

for all $S \subset \{1, 2, \ldots, K\}$. To achieve points on the boundary of the region, one needs to use optimal codes with block lengths approaching infinity. However, for long enough packets, one can come arbitrarily close to the boundary using codes with finite block lengths and achieving reasonable level of reliability. To simplify expressions, here, and throughout the paper, we assume codewords are long enough so that points on the boundary are basically achievable, but much shorter than the time window by which they must be transmitted. We restrict our discussion to two users, set $\sigma^2 = 1$, and define $f(r) \triangleq 2^{2r} - 1$. The results can be readily extended to more than two users.

For $K = 2$, the feasible region of received powers is simply

$$P_1 \geq f(r_1)$$
$$P_2 \geq f(r_2)$$
$$P_1 + P_2 \geq f(r_1 + r_2).$$

This is plotted in Fig. 3. Note that when the received power is P_i, the *transmitted* power is P_i / s_i. Time-division, i.e., one user transmitting at rate $\frac{r_1}{\alpha}$ for a fraction α of the time and the other transmitting at rate $\frac{r_2}{(1-\alpha)}$ for a fraction $1 - \alpha$ of the time, yields the region with the dashed boundary specified by

$$P_1 = \alpha f\left(\frac{r_1}{\alpha}\right) \quad \text{and} \quad P_2 = (1-\alpha) f\left(\frac{r_2}{(1-\alpha)}\right).$$

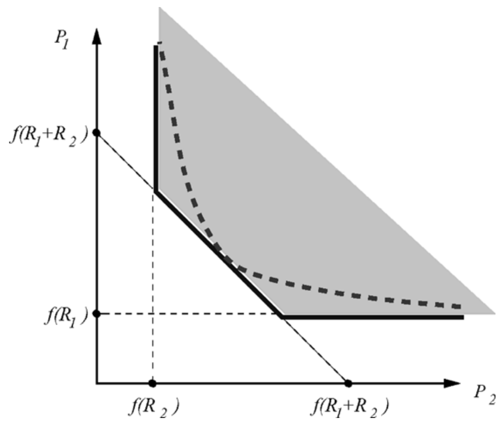

Fig. 3. Feasible region of (P_1, P_2) for a given (R_1, R_2). The multiple-access region is bounded below by the solid boundary and the time-division region is bounded below by the dashed boundary.

Note that the time-division boundary always touches the boundary of the multiple-access region at the point $(\alpha f(r_1 + r_2), (1 - \alpha) f(r_1 + r_2))$, where $\alpha = \frac{r_1}{r_1 + r_2}$.

We refer to the sequence of arrivals of the ith user as *stream* i and merge the two streams into one sequence $\{t_i\}$, as shown in Fig. 2, where stream 1 arrivals are marked by crosses and stream 2 arrivals are marked by circles. We denote the inter-arrivals of this new sequence by *data epochs*, or in short, epochs, and mark them ξ_i, $i = 1, \ldots, m$. Without loss of generality, we assume that a packet (from either one of the two users) is received at time 0, so $t_1 = 0$.

Before we present the multiple-access offline scheduling problem, we make the following two key observations (Lemmas 1,2), the first of which can be obtained from [20, Lemma 3.3], but is included here for completeness.

Lemma 1: In the symmetric case $(s_1 = s_2)$, B_1 and B_2 bits can be transmitted in τ time units with minimum energy by time sharing between the users (i.e., with time-division). In the asymmetric case $(s_1 < s_2)$, time-division is strictly suboptimal, and the unique optimal scheme is the corner point of the multiple-access energy region where P_1 is at its minimum possible value. In the AWGN case, this point corresponds to successive cancellation where users are decoded in decreasing order of s_i's.

Proof: Since the duration of the interval and the number of bits to be transmitted in that interval by each user are fixed, the average rates will be fixed at $r_{1i} = \frac{B_1}{\tau}$, and $r_{2i} = \frac{B_2}{\tau}$. Given this average rate pair, we wish to minimize the total transmitted energy $\tau(P_1/s_1 + P_2/s_2)$. The solution can be readily seen from the multiple-access achievable powers region in Fig. 3. When $s_1 = s_2$, any power pair on the line $P_1 + P_2 = f(r_1 + r_2)$ achieves the minimum. In particular, the point where the time-division boundary touches the multiple-access boundary minimizes the total transmitted power for the time-division scheme. When $s_1 \neq s_2$, the minimum is attained at one of the two corner points. For example, when $s_1 < s_2$, the optimal power pair is $(f(r_1), f(r_1 + r_2) - f(r_1))$. In the AWGN channel case, this is $(P_1, (P_1 + \sigma^2)(2^{2r_2} - 1))$ where $P_1 = \sigma^2(2^{2r_1} - 1)$, which can be achieved by decoding user 2, subtracting its signal from the received signal, and then decoding 1. $\square$

In the rest of this section, for ease of notation we assume that $1/s_1 = a > 1$, $1/s_2 = 1$.

Lemma 2: In an optimal multiple-access offline schedule, the rate of a user need not change during an epoch.

Proof: By definition, new data can only arrive at the start of a data epoch. So, at the beginning of an epoch, the two users together have a certain number of bits to be transmitted, and no new bits are added to this during the epoch. Now, let us focus on a generic data epoch in the optimal schedule. Assume without loss of generality that the epoch starts at $t = 0$ and ends at $t = t_e$. Assume that in this schedule the epoch is divided into k intervals, $[t_1 = 0, t_2), [t_2, t_3), \ldots, [t_k, t_e = t_{k+1})$, where the rates of both users are constant during an interval. Denote the first user's rate in interval $[t_i, t_{i+1})$ by r_{1i}, and the second's by r_{2i}. At optimal power settings (see Lemma 1) the total transmitted energy for the data epoch is given by

$$\mathcal{E} = \sum_{i=1}^{k} (t_{i+1} - t_i)((a-1)f(r_{1i}) + f(r_{1i} + r_{2i})).$$

By convexity of f, it is easy to see that the total energy can be decreased by using the average rates

$$r_1 = \sum_{i=1}^{k} r_{1i}(t_{i+1} - t_i)/t_e$$

and

$$r_2 = \sum_{i=1}^{k} r_{2i}(t_{i+1} - t_i)/t_e$$

throughout the epoch. □

We are now ready to state the minimum energy offline scheduling problem for the multiple-access channel. For simplicity, consider equal-sized packets each with B bits. The formulation and the results we obtain, however, can be readily generalized to packets of unequal size. Define the sequences $\{c_{1i}\}$ and $\{c_{2i}\}$ as the number of bits that have arrived at the beginning of epoch i for users 1 and 2, respectively. In the case of constant sized packets, this means that for $i \in \{1, \ldots, m\}$, $c_{ji} = B$ if there is a stream j arrival at the beginning of data epoch i, and 0 otherwise. By Lemma 2, the optimal multiple-access offline scheduling problem reduces to finding a rate pair sequence $\{(r_{11}, r_{21}), (r_{12}, r_{22}), \ldots, (r_{1m}, r_{2m})\}$ that minimizes the total energy. The problem is then as follows.

Problem 3: Multiple-Access Channel Offline Scheduling:

$$\text{Minimize}: \sum_{i=1}^{m} \xi_i((a-1)f(r_1) + f(r_1 + r_2))$$

$$\text{subject to}: \sum_{i=1}^{k} r_{ji}\xi_i \le \sum_{i=1}^{k} c_{ji}$$
$$k = 1, \ldots, m-1, j = 1, 2$$
$$\sum_{i=1}^{m} r_{ji}\xi_i = \sum_{i=1}^{m} c_{ji}, j = 1, 2.$$

Thus, similar to Problems 1 and 2, multiple-access channel scheduling is a convex optimization problem with linear constraints. We now show that in the symmetric case $(s_1 = s_2)$ time-division is optimal and thus the problem can be optimally solved using MoveRight. For convenience, define

$$g(r_1, r_2) \triangleq (a-1)f(r_1) + f(r_1 + r_2)$$

and note that g is convex in r_1 and r_2.

Theorem 1: In the symmetric case, there exists a schedule that achieves minimum energy by time-division between the packets of users. In the asymmetric case, time-division is strictly suboptimal.

Proof: Consider the symmetric case first. We show that any schedule can be converted into a time-division schedule with equal or lower energy. First, note that from Lemma 2, it suffices to consider the schedule as a sequence of rate pairs (r_{1i}, r_{2i}), one pair for each epoch. Also note that we can limit attention to the case where the received powers (P_{1i}, P_{2i}) are the optimal corner point of the feasible region for rates (r_{1i}, r_{2i}) (if not, the energy can be reduced without changing the schedule). Consider epoch i. From Lemma 1, in the symmetric case, there is a point on the time-division curve that achieves the average rates (r_{1i}, r_{2i}) with minimum total energy. We can move to this time-division point by letting the first user transmit alone in a fraction α_i of the total interval, i.e., for a duration $\alpha_i\xi_i$, using a rate $\frac{r_{1i}}{\alpha_i}$, and the second user transmit in the remaining with rate $\frac{r_{2i}}{(1-\alpha_i)}$, where $\alpha_i = \frac{r_{1i}}{r_{1i}+r_{2i}}$. Proceeding like this with other epochs, the schedule we were provided with has been converted to a time-division schedule of equal or lower energy. Now, consider a packet from user 1 that is being transmitted across the epochs $(i, i+1, \ldots, i+l)$, as l chunks, with instantaneous rates

$$\left[\frac{r_{1i}}{\alpha_i}, \frac{r_{1(i+1)}}{\alpha_{i+1}}, \ldots, \frac{r_{1(i+l)}}{\alpha_{i+l}}\right].$$

This packet has B bits, so $\sum_{j=0}^{l} r_{1(i+j)}\tau_{1(i+j)} = B$. The same amount of data can be transmitted by averaging user 1's rate over the l pieces, setting

$$\tilde{r} = \sum_{j=1}^{l} \frac{r_{1(i+j)}}{\alpha_j}\beta_j, \qquad \text{where } \beta_j = \frac{\alpha_j \tau_{i+j}}{\sum_k \alpha_k \tau_{i+k}}.$$

By convexity of the power function f, energy is reduced. Now that the rate has been averaged out, one can always collect these l pieces together to transmit the packet of user 1 as a whole. All of the above can be repeated for user 2. The result is time-division between the packets of user 1 and user 2, where rates (and powers) are set independently.

Now, consider the asymmetric case and suppose a time-division schedule is given. Take any interval that is divided between two users. By Lemma 1 one can convert[4] this epoch's rates to the optimal corner point on the multiple-access boundary, and strictly decrease energy. □

Problem 3 is a convex optimization problem and considering the conditions on $f(\cdot)$, it is easy to see that it has a unique solution. However, except for the symmetric case, the problem has no closed-form solution. Standard convex optimization methods could be employed to compute solutions. Such a general approach, though, is unlikely to provide as much insight as an approach that notices the special structure of the problem. The next section describes such an algorithm that

[4]If causality does not permit this, pick another time-division interval.

will be called "FlowRight." The FlowRight algorithm performs simple iterations starting from a feasible initial schedule. Each iteration strictly improves the schedule (decreases the total energy), which ultimately converges to the unique optimal.

A. FlowRight: An Algorithm for Optimal Offline Scheduling

FlowRight is an iterative algorithm. In the beginning, the transmission time of each packet is set to precisely the data epoch at the beginning of which the packet arrived. That is, in the starting schedule, packets begin transmission when they arrive, and end transmission when the next data epoch starts. Let the rates obtained in this way be $\{r^0_{1i}\}$ and $\{r^0_{2i}\}$, such that $r^0_{ji} = \frac{c_{ji}}{\xi_i}, j = 1, 2, i = 1, 2, \dots, m$.

The FlowRight algorithm performs local optimizations on pairs of epochs in the following way. Consider the first two data epochs. The total number of bits transmitted by users 1 and 2 in these two data epochs are $c_1 = r^0_{11}\xi_1 + r^0_{12}\xi_2$ and $c_2 = r^0_{21}\xi_1 + r^0_{22}\xi_2$, respectively. Keeping the number of bits fixed at c_1 and c_2, we update (r^0_{11}, r^0_{21}) to (r^1_{11}, r^1_{21}), where (r^1_{11}, r^1_{21}) is the allocation of rates to the first data epoch that minimizes the overall energy of the pair of data epochs. Obviously, when (r^1_{11}, r^1_{21}) are decreased (i.e., at least one component is decreased and neither is increased), (r^1_{12}, r^1_{22}) will increase, since the bits that leave the first epoch go to the second.[5] Note that $r^1_{11} \le r^0_{11}$ and $r^1_{21} \le r^0_{21}$, since, from the initial condition, information can only flow to the right (otherwise, causality would be violated.) We therefore have to reset (r^0_{12}, r^0_{22}) to new values which are larger (or equal to) their initial values.

Moving to the second pair of data epochs, this time optimally decrease (r^0_{12}, r^0_{22}) to (r^1_{12}, r^1_{22}), and reset the values of (r^0_{13}, r^0_{23}). Proceed in this way to obtain (r^1_{1i}, r^1_{2i}) for $i = 1, \dots, n$. This completes the first *pass* of the algorithm. It is easy to see that in the first pass, information can only flow to the right. Interestingly, we will later prove that information always flows right in the algorithm, and consequently, after each iteration the rates are closer to the optimal solution than they were before that iteration.

After the first pass is complete we start from the beginning and update the rates two data epochs at a time similarly to the above. Terminate after pass K, where

$$K = \min\{k : r^k_{ji} = r^{k-1}_{ji},\ i = 1, \dots m, \text{ and } j = 1, 2\}.$$

A pseudocode for the algorithm is given as follows.

```
passes = 0;
for i = 1 : n
{
   $r^0_{1i} = \frac{c_{1i}}{\xi_i}$;
   $r^0_{2i} = \frac{c_{2i}}{\xi_i}$;
}
done = 0;
while (done == 0)
{
  passes = passes + +;
  for i = 1 : n − 1
  {
  $[r^k_{1i}\ \ r^k_{1(i+1)}\ \ r^k_{2i}\ \ r^k_{2(i+1)}]$
  = update($[r^{k-1}_{1i}\ r^{k-1}_{1(i+1)}\ r^{k-1}_{2i}\ \ r^{k-1}_{2(i+1)}]$);
  }
  if ($r^k_1 == r^{k-1}_1$ and $r^k_2 == r^{k-1}_2$)
  {
   done = 1;
  }
}
```

[5]We are allowing fractional numbers of bits to move between epochs.

In each pass, the function update is run $m-1$ times, i.e., once for each consecutive pair of data epochs. It locally minimizes the energy of the two epochs in the following way. Consider the kth pass when update is running on epochs $i, i+1$. The problem is that of choosing (r_{1i}, r_{2i}) that minimize the total energy of epochs i and $i+1$, while the total number of bits on each stream is fixed at

$$\xi_i r^{k-1}_{1i} + \xi_{i+1} r^{k-1}_{1(i+1)} = b^{(k-1)}_{1i}$$

and

$$\xi_i r^{k-1}_{2i} + \xi_{i+1} r^{k-1}_{2(i+1)} = b^{(k-1)}_{2i}$$

respectively.

The following definition will be useful in the proofs later:

$$h^{(b^{(k-1)}_{1i}, b^{(k-1)}_{2i})}(r_{1i}, r_{2i}) \triangleq \xi_i g(r_{1i}, r_{2i}) + \xi_{i+1} g\left(\frac{b^{(k-1)}_{1i} - r_{1i}\xi_i}{\xi_{i+1}}, \frac{b^{(k-1)}_{2i} - r_{2i}\xi_i}{\xi_{i+1}}\right).$$

Note that $h^{(b^{(k-1)}_{1i}, b^{(k-1)}_{2i})}(r_{1i}, r_{2i})$ is the total energy of epochs i and $i+1$, and that it is convex in the rates r_{1i} and r_{2i}.

In order to prove that this algorithm finds the optimal offline schedule, we make the two observations described later in Lemmas 3 and 4. Consider two data epochs, 1 and 2, with durations ξ_1 and ξ_2. The first stream needs to transmit a total of b_1 bits in the two data epochs and the second stream needs to transmit b_2 bits. Of the b_1 bits, $b_{11} \le b_1$ are available in the first data epoch, and of the b_2 bits, $b_{21} \le b_2$ are available in the first data epoch. Let the rates chosen for the first epoch be r_{11}, r_{21}, hence, the total energy is $h^{(b_1, b_2)}(r_{11}, r_{21})$.

The first observation to make is that after update runs on this epoch pair, the partial derivatives of h are either zero or negative (which corresponds to a causality constraint being met with equality.) This is made precise in Lemma 3.

Lemma 3: The optimal rate pair $(\hat{r}_{11}, \hat{r}_{21})$ is unique, and satisfies $h_x(\hat{r}_{11}, \hat{r}_{21}) \le 0$ and $h_y(\hat{r}_{11}, \hat{r}_{21}) \le 0$, where h_x and h_y are the partial derivatives of h with respect to the first and second components, respectively. If $h_x(\hat{r}_{11}, \hat{r}_{21}) < 0$, then a new packet is starting transmission at epoch 2 on stream 1, and if $h_y(\hat{r}_{11}, \hat{r}_{21}) < 0$, a new packet starts on stream 2.

The proof of Lemma 3 is given in the Appendix . The second observation is about what happens if we take out (eject) some bits from the second epoch or put in (inject) some extra bits into the first epoch and run update again. This is the crucial step in proving the convergence of FlowRight, as such bit ejections/injections happen from one iteration to the next. We record the observation in Lemma 4 (proven in the Appendix).

Lemma 4: Suppose, after update has run on the epochs resulting in the situation in Lemma 3, some additional bits are injected into epoch 1 on one or both streams, that is, set $r'_{11} = \hat{r}_{11} + \Delta_{11}$, and $r'_{21} = \hat{r}_{21} + \Delta_{21}$, $\Delta_{11} \geq 0$ and $\Delta_{21} \geq 0$. Also, some bits are ejected from the second epoch, i.e., set $r'_{12} = \hat{r}_{12} - \Delta_{12}$, and $r'_{22} = \hat{r}_{22} - \Delta_{22}$, $\Delta_{12} \geq 0$, and $\Delta_{22} \geq 0$. For the change to be nontrivial, we require that at least one of $\{\Delta_{11}, \Delta_{12}, \Delta_{21}, \Delta_{22}\}$ is nonzero. Notice that the constraint space of the problem has changed. In this case we have the following.

1) If $h_x(\hat{r}_{11}, \hat{r}_{21}) = 0$ and $h_y(\hat{r}_{11}, \hat{r}_{21}) = 0$, then after the injection/ejection of the new bits, when `update` is run again, there will be a right push, i.e., a nonnegative amount of information will move from epoch 1 to epoch 2 on both streams.
2) If $h_x(\hat{r}_{11}, \hat{r}_{21}) < 0$, the first stream was limited by causality before the injection/ejection, hence it is limited by causality again (because no information has crossed from epoch 1 into epoch 2). If $h_x(\hat{r}_{11}+\Delta_{11}, \hat{r}_{21}-\Delta_{21}) > 0$, there will be a push on stream 1. Otherwise, due to causality, reoptimization will not result in any move on that stream (i.e., no push or pull). Similarly, if $h_y(\hat{r}_{11}, \hat{r}_{21}) < 0$, on the second stream there may only be a push or no movement at all.

Now, let $\{r_{1i}^{\text{opt}}\}$ and $\{r_{2i}^{\text{opt}}\}$ be the pair of *optimal* rate sequences. We shall sometimes use the shorthand r^{opt} to refer to this pair. Such a unique solution exists because of the convexity of the problem and the compactness of the search space. In the following, it is proved that the algorithm **FlowRight** results in $\{r_{1i}^{\text{opt}}\}$ and $\{r_{2i}^{\text{opt}}\}$. In order to show this, we first argue that the algorithm stops at the pair of sequences $\{r_{1i}^{\infty}\}$ and $\{r_{2i}^{\infty}\}$. We then show that this is identical to r^{opt}. The following results are proved similarly to Theorem 1 in [8].

Theorem 2: The following statements hold.

1) As the algorithm FlowRight runs, information always flows right.
2) FlowRight stops, and returns two sequences $\{r_{1i}^{\infty}\}$ and $\{r_{2i}^{\infty}\}$.
3) $\{r_{1i}^{\infty}\} = \{r_{1i}^{\text{opt}}\}$ and $\{r_{2i}^{\infty}\} = \{r_{2i}^{\text{opt}}\}$.

Proof:

1) The claim is that throughout the running of FlowRight, all pushes are to the right. We now prove this by induction. In the first pass, the claim is trivially true, since all left pushes are impossible due to causality. Now, suppose that we are on the kth pass of the algorithm, and so far update has operated on all epoch pairs up to and including the pair $(i-1, i)$, and all pushes so far have been to the right. We will show that the next push will be to the right. On the $(k-1)$th run, `update` performed a local optimization on epochs $(i, i+1)$. Let us call the two pairs of rates resulting from this optimization $(\hat{r}_{1i}, \hat{r}_{2i})$, and $(\hat{r}_{1(i+1)}, \hat{r}_{2(i+1)})$. The number of bits transmitted in the ith epoch on streams 1 and 2 are $b_{1i} \triangleq \hat{r}_{1i}\xi_i$ and $b_{2i} \triangleq \hat{r}_{2i}\xi_i$. Similarly, define $b_{1(i+1)}$ and $b_{2(i+1)}$ to be the bits transmitted the next epoch, and define
$$b_1 \triangleq b_{1i} + b_{1(i+1)} \quad \text{and} \quad b_2 \triangleq b_{2i} + b_{2(i+1)}.$$
From Lemma 4, part 1,
$$h_x^{(b_1,b_2)}(\hat{r}_{1i}, \hat{r}_{2i}) \leq 0$$
and
$$h_y^{(b_1,b_2)}(\hat{r}_{1i}, \hat{r}_{2i}) \leq 0.$$
Now, as the $(k-1)$th pass progresses, update performs a local optimization on $(i+1, i+2)$, and this, by hypothesis, results in a right push (i.e., a push from $i+1$ onto $i+2$), which changes $(\hat{r}_{1(i+1)}, \hat{r}_{2(i+1)})$ to $(\hat{r}_{1(i+1)} - \Delta_{1(i+1)}, \hat{r}_{2(i+1)} - \Delta_{2(i+1)})$, where $\Delta_{j(i+1)} \geq 0$ for $j = 1, 2$. Continuing to the present time, on the kth pass there is a right push (again, by the induction hypothesis), from $i-1$ to i resulting in $(\hat{r}_{1i}+\Delta_{1i}, \hat{r}_{2i}+\Delta_{2i})$, where $\Delta_{ji} \geq 0$ for $j = 1, 2$. By part 2 of Lemma 4, there can only be a right push (if any) from i to $i+1$ on the kth iteration.
2) Consider $r_{11}^k \xi_{11} = b_{11}^k < \infty$, i.e., the total number of bits of stream 1 on epoch 1 after the kth iteration. Since all pushes are to the right, b_{11}^k is monotonically nonincreasing. Also, it is obviously bounded from below by zero. Therefore, $b_{11}^k \downarrow b_{11}^{\infty}$, and therefore $r_{11}^k \downarrow r_{11}^{\infty}$. Similarly, $b_{11}^k + b_{12}^k$ (the total number of bits in epochs 1 and 2 on stream 1) is monotonic nonincreasing and bounded below by zero. Hence, this sum tends to a limit;
$$(b_{11}^k + b_{12}^k) \downarrow (b_{11}^{\infty} + b_{12}^{\infty}).$$
Therefore, $b_{12}^k \downarrow b_{12}^{\infty}$, and $r_{12}^k \downarrow r_{12}^{\infty}$. Similarly, since $(b_{11}^k + b_{12}^k + b_{13}^k)$ converges and $(b_{11}^k + b_{12}^k)$ converges, so does b_{13}^k. Proceeding like this, we will see that $r_{ji}^k \downarrow r_{ji}^{\infty}$, $j = 1, 2$, for all i. Hence, the sequences of rates converge.
3) First, observe that $h_x(r_{1i}^{\infty}, r_{2i}^{\infty}) \leq 0$ and $h_y(r_{1i}^{\infty}, r_{2i}^{\infty}) \leq 0$ for all i. To see why this is true, suppose it is not. That is, let $h_x(r_{1\hat{i}}^{\infty}, r_{2\hat{i}}^{\infty}) > 0$ for some $\hat{i}$. Then, if we run FlowRight on this sequence, there will be a right push. This contradicts the fact that $\{(r_{1i}^{\infty}, r_{2i}^{\infty})\}$ is a fixed point. Hence, we have $h_x(r_{1i}^{\infty}, r_{2i}^{\infty}) \leq 0$ and $h_y(r_{1i}^{\infty}, r_{2i}^{\infty}) \leq 0$ for all i, and using that we will show that $\{(r_{1i}^{\infty}, r_{2i}^{\infty})\}$ satisfies the Karush–Kuhn–Tucker (KKT) conditions [4].

Recall that our problem is a convex problem with linear inequality constraints, so the KKT conditions are sufficient for optimality in this case. The first of those conditions is feasibility, of course, but we already know that $\{(r_{1i}^{\infty}, r_{2i}^{\infty})\}$ is a feasible solution (FlowRight always respects feasibility). Then we need only check if for our solution there is a set of Lagrange multipliers with the properties specified by the KKT conditions. Differentiating the Lagrangian for the problem provides us with $2m$ equations
$$\xi_i g_x(r_{1i}, r_{2i}) + \sum_{l=i}^{m} \lambda_l \xi_i = 0, \quad 1 \leq i \leq m$$
and
$$\xi_i g_y(r_{1i}, r_{2i}) + \sum_{l=i}^{2m} \lambda_l \xi_{i-m} = 0, \quad m+1 \leq i \leq 2m$$
where λ_l, $l = 1, \ldots, 2m$ are the Lagrange multipliers. Now we need to inquire about the values of these La-

grange multipliers. By subtracting the mth equation from the $(m-1)$th, we obtain

$$\begin{aligned}\lambda_{m-1} &= -\left[g_x(r_{1(m-1)}, r_{2(m-1)}) - g_x(r_{1m}, r_{2m})\right] \\ &= -h_x(r_{1(m-1)}, r_{2(m-1)}).\end{aligned}$$

Now, substitute $(r^{\infty}_{1(m-1)}, r^{\infty}_{2(m-1)})$ for $(r_{1(m-1)}, r_{2(m-1)})$, and by what we showed above, we obtain $\lambda_{m-1} \le 0$. Further, using Lemma 4, $\lambda_{m-1} > 0$ if the $(m-1)$th constraint is active (i.e., all the bits that are present by that time have been transmitted by the end of epoch $m-1$ on stream 1), and $\lambda_{m-1} = 0$ otherwise. Proceeding this way, we obtain that $\lambda_l > 0$ if the lth constraint is active (i.e., again, all the bits that are present by that time have been transmitted by the end of epoch l on stream 1), and $\lambda_l = 0$ otherwise, for $1 \le l \le m$, and $\lambda_l > 0$ if the $(m+l)$th constraint is active (i.e., causality is met at the end of epoch l on stream 2), and $\lambda_l = 0$ otherwise, for $1 \le l \le m$. But with these, we have the KKT conditions completely satisfied. This proves that $\{(r^{\infty}_{1i}, r^{\infty}_{2i})\}$ is the globally optimal solution of our problem. □

B. Time-Division Versus Optimal Multiple-Access

As we have pointed out before, one can calculate the best time-division offline schedule by running the MoveRight algorithm on the joint sequence of packet arrivals of all users. It is interesting to find out how time-division compares to the optimal solution. To that end, in this subsection we compare the average energies consumed by MoveRight and FlowRight on the same arrival sequence.

The experiment setup is as follows. The two users' packets arrive according to two independent Poisson processes with identical rates, with a combined rate of λ arrivals per unit time (unit time is a symbol time). For each value of λ, 1000 arrivals are generated, and T is set to $t_{1000} + 1/\lambda$. Then, both FlowRight and MoveRight are run on this sequence, and the average energy per packet and average delay per packet for both users are calculated. Here, $a_1 = 32$ and $a_2 = 1$, these values were chosen because the resulting average energy and delay values are very similar for the two users. In Fig. 4, the average energy and delay values in both time-division and optimal solutions are plotted. The results suggest that the energy per packet can be reduced significantly by using multiple-access codes, especially at high rates. But note that when λ is small, and consequently transmission rates are low, time-division is almost as good.

C. Extension to the Broadcast Channel

Consider an AWGN broadcast channel with one sender and two receivers. The sender has two streams of packet arrivals, with one stream destined to each receiver. As before (see Fig. 2), we merge the packets into a single sequence.

The received signal to the ith receiver at time k is given by

$$Y_i[k] = \sqrt{s_i}X[k] + Z_i[k] \tag{4}$$

where $X[k]$ is the transmitted signal with average power constraint P, $\sqrt{s_i}$ is the channel gain, and the $Z_i[k]$'s are i.i.d.

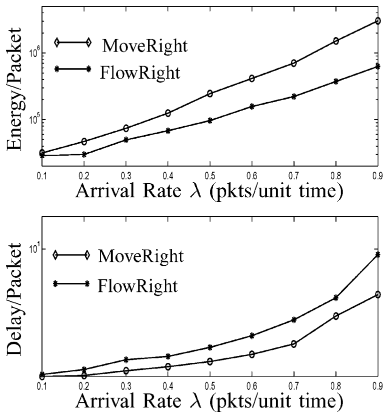

Fig. 4. Comparison of offline time-division and multiple-access schedules as obtained by the MoveRight and FlowRight algorithms for a two-user multiple-access channel. The users' packets arrive according to two independent Poisson processes with identical rates, and the combined arrival process is at rate λ. The energy values correspond to 10^3-bit packets. The signaling rate is 10^6 transmissions/s, and the nominal rate is 6 bits/transmission—that is obtained when a packet takes 1 time unit, i.e., $\frac{10^{-2}}{6}$ seconds, to transmit.

zero-mean Gaussian noise with variance σ^2. The capacity region of the channel (see [7]) assuming $s_1 < s_2$, is the set of rate pairs (r_1, r_2) such that

$$\begin{aligned} r_1 &\le \frac{1}{2}\log_2\left(1 + \frac{\alpha s_1 P}{\sigma^2}\right) \\ r_2 &\le \frac{1}{2}\log_2\left(1 + \frac{(1-\alpha)s_2 P}{\alpha s_2 P + \sigma^2}\right)\end{aligned}$$

for some $0 \le \alpha \le 1$. Now, we express the minimum average power for a given rate pair, where the above inequalities are replaced by equalities, as follows. Rewrite the first equality as $\alpha s_1 P/\sigma^2 = 2^{2r_1} - 1 = f(r_1)$. Hence, $\alpha = (\sigma^2/Ps_1)f(r_1)$. Substituting into the second inequality and rearranging we obtain

$$P = \sigma^2\left(\frac{f(r_1)}{s_1} + \frac{f(r_2)}{s_2} + \frac{f(r_1)f(r_2)}{s_1}\right).$$

Consider epochs defined in the same way as in Section III, as the packet inter-arrival times of the merged sequence. Again, making the observation that in an optimal broadcast schedule rates do not need to change during an epoch, the offline scheduling problem is as follows.

Problem 4: Broadcast Channel Offline Scheduling:

$$\begin{aligned}\text{Minimize : } & \sum_{i=1}^{m}\xi_i\left(\frac{f(r_1)}{s_1} + \frac{f(r_2)}{s_2} + \frac{f(r_1)f(r_2)}{s_1}\right) \\ \text{subject to : } & \sum_{i=1}^{k} r_{ji}\xi_i \le \sum_{i=1}^{k} c_{ji} \\ & k = 1,\dots,m-1,\ j = 1,2 \\ & \sum_{i=1}^{m} r_{ji}\xi_i = \sum_{i=1}^{m} c_{ji},\ j = 1,2.\end{aligned}$$

Note that this is the same as Problem 3 in Section III, except that the objective function is different. But the objective function is still convex, monotonically increasing, and differentiable in both r_1 and r_2. Hence we get the following.

Theorem 3: FlowRight finds the unique solution to Problem 4.

D. Online Scheduling

The optimal offline algorithm provides a lower bound on energy for all possible online algorithms, as it finds the minimum possible energy per packet under complete knowledge of the future. Fortunately, the optimal offline algorithm naturally lends itself to online use by means of a simple look-ahead buffer [8]. We buffer all packets which arrive in the interval $[0, L)$ and optimally schedule them for departure in the interval $[L, 2L)$. Note that FlowRight does not need to perform any iterations—the rate of each user is set to the number of bits it has in the buffer divided by L and transmission powers are chosen optimally according to the feasible power region for the given rates. Continuing with the schedule, the packets that arrive in $[L, 2L)$ are buffered to be transmitted in $[2L, 3L)$, and so on.

In Fig. 5, we compare the online algorithm obtained in this way, with the online algorithm obtained by using the look-ahead buffer with MoveRight. The experiment setup is similar to the one in Section III-C. The only difference is that now we have a look-ahead buffer. In the experiment whose results are shown in the figure, the look-ahead window size was held at $L = 25$ time units.[6] Hence, as our online algorithm "adapts" to the arrival rate, the average delay remains around 25 time units. The resulting average energy per packet is reasonably close to the optimal, which is also plotted in Fig. 5, and which, of course, has much lower delay. Therefore, we see that by incurring some *fixed* delay, it is possible to perform very energy efficiently.

IV. Scheduling Over Slow-Fading Channels

Consider the AWGN channel as specified by (3). We make the block-fading assumption where the power gain $s_i[k]$ changes every T_c channel uses (a"coherence window"). Further assume that fading is slow with respect to codeword lengths. Initially, consider the single-user case, i.e., $k = 1$. We assume that both the transmitter and the receiver have perfect channel state information at the beginning of each coherence window.

As before, consider packets coming at arbitrary instants in $[0, T)$, all of which need to be transmitted within this same time period. The optimal offline schedule is the one that minimizes the total packet transmission energy given perfect knowledge of the packet arrival instants and channel state values for the entire duration $[0, T)$, at time 0.

Define an "epoch" to be a time interval that begins with either a packet arrival or a change in the channel state, and continues until the next arrival or state change. The first epoch starts at $t_1 = 0$, and continues until t_2 or T_c, whichever is smaller, at which point the second epoch starts, and so forth. Let c_i denote the number of bits that have arrived at the beginning of epoch i, so $c_i > 0$ if the ith epoch starts with a packet arrival, and $c_i = 0$ otherwise. Let the duration of epoch i be ξ_i.

Lemma 5: In an optimal schedule, rate is constant during an epoch.

[6]In [8], a window of 20–30 time units was shown to be a good choice for L, in the sense that most of the achievable decrease in energy is obtained.

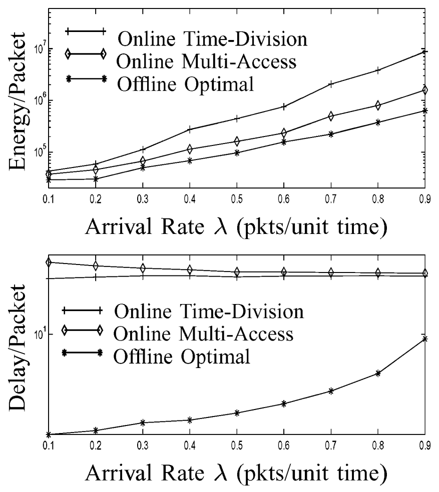

Fig. 5. Comparison of the online algorithms look-ahead time-division and look-ahead multiple-access for a two-user multiple-access channel. The users' packets arrive according to two independent Poisson processes with identical rates. The window size is 25 time units. The energy values correspond to 10^3-bit packets. The signaling rate is 10^6 transmissions/s, and the nominal rate is 6 bits/transmission achieved when a packet takes 1 time unit, i.e., $\frac{10^{-2}}{6}$ seconds, to transmit.

Proof: Noting that the channel state and the number of available bits are constant during an epoch by definition, the proof follows very similarly to the proof of Lemma 2. Suppose the rate is r_1 in the first τ_1 time units of an epoch of length t, and r_2 during the remaining $t - \tau_1$. The transmit energy in this epoch is then $\tau_1 f(r_1)/s + (t - \tau_1) f(r_2)/s$, where s is the fading state during the epoch. The same number of bits can also be transmitted using the uniform rate $(r_1\tau_1 + r_2(t - \tau_1))/t$ for the whole time t. This new rate results in a total energy $tf((r_1\tau_1 + r_2(t - \tau_1))/t)/s$, which, by convexity of f, is strictly lower than previous, unless $r_1 = r_2$. $\square$

From Lemma 5, $\{r_i\}_{i=1}^m$ (m is the number of epochs) is sufficient to characterize the optimal schedule.

Problem 5: Offline Scheduling for the Slow-Fading Channel:

$$\begin{aligned} \text{Minimize : } & \sum_{i=1}^{m} \xi_i f(r_i)/s_i \\ \text{subject to : } & \sum_{i=1}^{k} r_i\xi_i \leq \sum_{i=1}^{k} c_i \\ & k = 1, \ldots, m-1 \\ & \sum_{i=1}^{m} r_i\xi_i = \sum_{i=1}^{m} c_i. \end{aligned}$$

This convex optimization problem can be solved by the FlowRight algorithm. Initially, the rates are set to $r_i^0 = c_i/\xi_i$, $i = 1, 2, \ldots, m$. The first two epochs are then considered. The total number of bits transmitted in these two data epochs is $r_1^0\xi_1 + r_2^0\xi_2$. Keeping the total number of bits fixed, r_1^0 is updated to r_1^1, the value that minimizes the total energy of the first pair of data epochs. Note that $r_1^1 \leq r_1^0$, since from their initial condition information can only be moved to the right (otherwise, causality would be violated.) Therefore, r_2^0 is reset to a new value that is larger than (or equal to) its initial value. Moving to the second pair of epochs, this time r_2^0 is optimally decreased to r_2^1, and the value of r_3^0 is reset. Proceeding in this way we obtain r_i^1 for $i = 1, \dots, n$. This completes the first *pass* of the algorithm. The algorithm then repeats the same procedure and terminates after K passes, where

$$K = \min\{k : |r_i^k - r_i^{k-1}| < \epsilon\}, \qquad \text{where } i = 1, \dots m$$

for small enough $\epsilon > 0$.

Theorem 4: The following statements hold.

1) As the algorithm FlowRight runs on $\{r_i\}$, information always flows to the right.
2) FlowRight stops, and returns a sequence $\{r_i^\infty\}$.
3) $\{r_i^\infty\} = \{r_i^{\text{opt}}\}$.

The only difference between this theorem and Theorem 2 is that the energy function, due to scaling with channel gain, does not have the same form for each epoch pair

$$h(r_i) = \xi_i f(r_i)/s_i + \xi_{i+1} f(r_{i+1})/s_{i+1}.$$

Note, however, that this does not affect any of the steps of the proof of Theorem 2, and therefore the proof of this theorem follows from the proof of Theorem 2.

A. Online Scheduling

We assume that the packet input process into the transmitter buffer is stationary and ergodic. The time average arrival rate

$$\lambda \triangleq \lim_{T\to\infty} \frac{1}{T} \int_0^T \lambda(t) \mathrm{d}t$$

is bounded such that $\lambda < \lambda_{\max}$ with probability 1. We are interested in schedules that are stable, i.e., scheduling algorithms that ensure that the number of packets in the buffer is finite with probability 1.

Future arrivals, channel states, or λ are not known. The channel has slow ergodic fading with known statistics where the power gains of different coherence windows are i.i.d. The transmitter knows the present value of the channel gain just before transmitting a packet. The bound on packet arrival rate $\lambda_{\max}$ is also known. We first describe an online scheduling algorithm based on water-filling in time that is known to achieve the capacity of the channel. Next, we describe look-ahead water-filling, an algorithm that simultaneously adapts to both the channel and the data arrival rate.

1) Water-Filling in Time: It is well known (see [11]) that the capacity of the AWGN channel with ergodic fading and with the channel gain known at both the transmitter and receiver is given by

$$C = \frac{1}{2} \int_{s_o}^{\infty} \log\left(\frac{s}{s_o}\right) p(s) \mathrm{d}s \text{ bits/transmission}$$

where $p(s)$ is the probability density function of the channel gain s, and s_o is the solution to the equation

$$\int_{s_o}^{\infty} \left(\frac{1}{s_o} - \frac{1}{s}\right) p(s) \mathrm{d}s = \frac{P}{\sigma^2}$$

where P is the average transmit power, and σ^2 is the noise variance. Furthermore, capacity is achieved by a "water-filling" power allocation to states. This means that transmit power is set to $\sigma^2(\frac{1}{s_o} - \frac{1}{s})$ if s is larger than the cutoff value s_o, and if not, there is no transmission until s changes. Hence the instantaneous transmit power is

$$P(s) = \begin{cases} \sigma^2(\frac{1}{s_o} - \frac{1}{s}), & \text{if } s \geq s_o \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

To use water-filling adaptation in our setting, we set the average rate equal to $\lambda_{\max}$ (packets/time unit) $\times B$ (bits/packet) $\times T_s$ (time units/symbol) to ensure stability. We then calculate s_o (hence, determine the average power) such that the capacity is equal to this target average rate. Before each packet transmission, the instantaneous power and rate are set according to the channel gain s, which is assumed constant during packet transmission.

2) Look-Ahead Water-Filling Algorithm: The water-filling scheduling algorithm presented above optimally adapts to the channel state, and is energy optimal if the average rate of packet arrivals is close to $\lambda_{\max}$. But this algorithm can be wasteful when the instantaneous packet arrival rate is much lower than $\lambda_{\max}$. Now we describe an online algorithm, which we refer to as look-ahead water-filling algorithm, that adapts jointly to the channel and backlog.

The algorithm is as follows: suppose just before time t, a packet transmission ended. Let the backlog at time t be $q(t)$. If $q(t) > 0$, then we begin transmitting the packet at the head of the queue at time t (otherwise, wait until there is a packet in the queue). We set the target transmission rate to

$$\hat{\mu} = \min\{q(t)/L, \lambda_{\max}\} \text{ packets/time unit}$$

for some constant $L > 0$. Given $\hat{\mu}$, we determine the instantaneous transmission rate according to water-filling. That is, the optimal cutoff value s_o is computed as in Section IV-B1, which corresponds to an average power for which the capacity is $BT_s\hat{\mu}$ (Using the concavity of the logarithm, the value of s_o is calculated iteratively.) The current power and rate are then determined from (5). We transmit the packet at the head of the queue with this rate. The following pseudocode summarizes the algorithm.

```
calculate rate estimate
                  $\hat{r} = T_s \min\{q(t)/L, \lambda_{\max}\}/B$;
find $s_o$ for which $\frac{1}{2}\int_{s_o}^{\infty} \log(\frac{s}{s_o}) p(s) = \hat{r}$
if ($s > s_o$)
{
 snr $= s/s_o - 1$;
}
else
{
 snr $= 0$
}
$r = \frac{1}{2}\log_2(1 + \text{snr})$;
if $r > 0$
{
  $q(t+B/r) = q(t) - 1 +$ number of arrivals in $(t, t+B/r)$;
 $t = t + B/r$;
}
else
{
 $t=$ time of next packet arrival;
}
repeat
```

Note that, in the look-ahead water-filling algorithm, the target packet transmission rate $\hat{\mu}$ never exceeds $\lambda_{\max}$, yet the queue is stable. This is shown in the following lemma.

Lemma 6: The look-ahead water-filling algorithm is stable, i.e., given any t, with probability one there exists $t_1, t < t_1 < \infty$, such that $q(t_1) = 0$.

Proof of Lemma 6: Suppose the claim is false. Consider the Markov process[7] where the state at time t is $q(t)$, and state transitions occur in the underlying Markov chain whenever there is a packet arrival or departure. Let $q(0) = 0$. Defining $A(t)$ and $D(t)$ as the counting processes of the number of arrivals and departures, respectively, from 0 to t

$$q(t) = q(0) + A(t) - D(t). \tag{6}$$

Since $q(0)$ is finite and $A(t)$ increases at most linearly with t, the expectation $\mathrm{E}(q(t))$ is defined for every t. Then

$$\mathrm{E}(q(t)) = q(0) + \mathrm{E}(A(t)) - \mathrm{E}(D(t)). \tag{7}$$

Our hypothesis that the queue is not stable implies that the underlying Markov chain is transient, hence, eventually any finite set of states has probability zero. Consequently, the event $\{q(t) < \lambda_{\max}L\}$ will eventually have zero probability. But, referring to the algorithm, in the event $\{q(t) > \lambda_{\max}L\}$, the expected transmission rate (where the expectation is over the fading process) is λ, and herefore in the limit (referring to the algorithm) departures will happen at rate $\lambda_{\max}$ packets/time unit. Dividing by t and taking the limit in (7), we obtain

$$\begin{aligned}\lim_{t\to\infty} \mathrm{E}\left(\frac{q(t)}{t}\right) &= \lim_{t\to\infty} \mathrm{E}(\frac{A(t,0)}{t}) - \lim_{t\to\infty} \mathrm{E}\left(\frac{D(t,0)}{t}\right) \\ &= \lambda - \lambda_{\max}. \end{aligned}\tag{8}$$

This implies $\lim_{t\to\infty} \mathrm{E}q(t) < 0$, which is a contradiction. □

To compare the look-ahead water-filling algorithm to water-filling, we perform the following experiment: Let 1-kbit packets arrive at the buffer at a rate $\lambda < 1$ arrivals/ time unit. A time unit is 1/6 ms, which corresponds to the transmission duration of a packet if it is transmitted at $r = 6$ bits/symbol (symbol rate is constant at 10^6 symbols/s). The packet arrival process is a Markov-modulated Poisson process for which $\lambda(t) = \beta\lambda$ with probability $0.9/\beta$, and $\lambda(t) = \frac{1}{10-9/\beta}\lambda$ otherwise. The parameter $\beta \geq 1$ is chosen such that the process is ergodic with expected rate λ. Note that when $\beta > 1$, the arrival process is bursty, and for $\beta = 1$ it reduces to a Poisson process at rate λ.

Fig. 1 shows an example run of bursty packet arrivals at $\lambda \simeq 0.5$, scheduled by the three algorithms WF (water-filling), LW(look-ahead water-filling), and OPT (optimal offline). Notice that water-filling transmits with much higher rate than the other two algorithms, thus quickly finishes its backlog and idles a significant amount of the time. Look-ahead water-filling, on the other hand, spreads its rate more uniformly over time, almost as uniformly as OPT which has the lowest rate transmission.

[7]Sometimes referred to as a semi-Markov process [10].

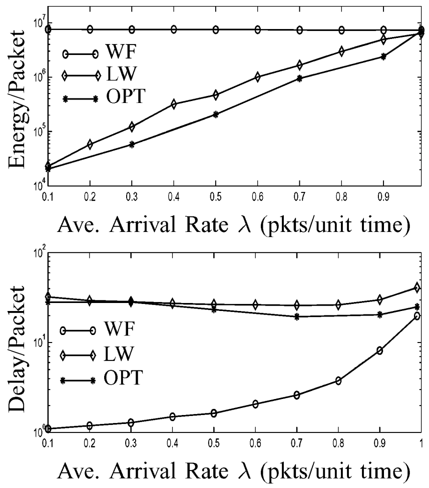

Fig. 6. Energy per packet as arrival rate changes for $\beta = 1$ in Rayleigh fading; $L = 25$.

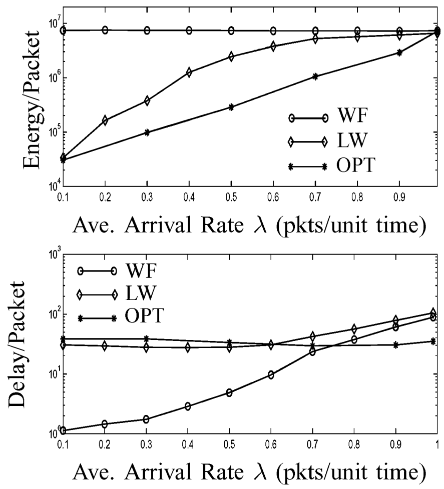

Fig. 7. Average energy per packet as arrival rate changes for $\beta = 2$ in Rayleigh fading; $L = 25$.

Figs. 6 and 7 explore the energy and delay performance of these algorithms. Note that the water-filling schedule has constant energy for all arrival rates, since the rate it assigns to packets is independent of λ. This energy is much higher than the average energy values achieved by look-ahead water-filling when λ is small; both for bursty and nonbursty arrival processes. Of course, the energy efficiency is achieved at the expense of an increase in delay. The delay of look-ahead water-filling is essentially lower-bounded by L, as it allows this time to monitor the arrival process. However, as can be observed from Fig. 6, the variation of its delay is much smaller than that of water-filling. In the figure, the delay of water-filling varies by about 7000% as λ is varied from 0 to $\lambda_{\max} = 1$, while the delay of look-ahead water-filling varies only about 60%. The fact that the *delay jitter* is so much smaller makes the backlog-adaptive algorithm attractive for data applications, especially streaming media.

B. Extension to Multiple-Access and Broadcast Channels With Fading

In this subsection, we extend the slow-fading single-user offline and online scheduling results to multiple-access and broadcast channels. To formulate the offline scheduling problem we first merge all users' packet arrival sequences and the times at which channel states change to obtain m epochs and note as before that in an optimal schedule rates do not need to change during an epoch.

Problem 6: Offline Scheduling for the Slow Fading Multiple-Access Channel:

$$\begin{aligned}\text{Min.}:\ &\sum_{i=1}^{m}\xi_i\left(\left(\frac{1}{s_{k_1 i}}-\frac{1}{s_{k_2 i}}\right)f(r_{k_1 i})+\frac{1}{s_{k_2 i}}f(r_{k_1 i}+r_{k_2 i})\right)\\ \text{s.t.}:\ &\sum_{i=1}^{k}r_{ji}\xi_i\le\sum_{i=1}^{k}c_{ji}\qquad k=1,\dots,m-1,\ \ j=1,2\\ &\sum_{i=1}^{m}r_{ji}\xi_i=\sum_{i=1}^{m}c_{ji},\qquad j=1,2\end{aligned}$$

where

$$k_1 i=\arg\min_{k\in\{1,2\}}(s_{1i},s_{2i})\quad\text{and}\quad k_2 i=\{1,2\}-\{k_1 i\}$$

and where $c_{ji}=B$ if a packet for user j arrives at the beginning of epoch i, and 0 otherwise.

This is a convex optimization problem with linear constraints and can be solved by FlowRight.

Recall that in the single-user case, the optimal adaptation to the channel state is given by the water-filling solution. In the multiple-user case, analogous results exist. When the fading processes of users are i.i.d., and the goal is to maximize the sum rate with respect to a total power constraint, the important result of Knopp and Humblet [15] says that the optimal power control scheme allows only the user with the best channel to transmit at any given time. The rate of that user is then determined by water-filling across the channel states. Tse and Hanly [20] exhibit the optimal power control when users are not necessarily symmetric, and the goal is to maximize a weighted sum of the rates. They propose a "greedy algorithm" which also solves the dual problem, i.e., achieves a given vector of average rates with minimum power. The greedy algorithm is an optimal online algorithm, as long as the transmitter knows the fading state.

A "look-ahead greedy" online schedule that uses a look-ahead buffer to adapt to both backlog and channel state (similar to look-ahead water-filling) can be obtained as follows. Each user's required rate is estimated from the current backlogs. The power allocation is then determined using the greedy algorithm in [20].

Finally, note that the broadcast scheduling problem in the slow-fading channel can be stated and solved similarly.

C. Fast-Fading Channels

Now, suppose that fading is fast with respect to our codeword lengths, so that a codeword will experience many channel realizations. In this setting, it is natural to assume that the transmitter does not have CSI; by the time feedback from the receiver about channel state reaches the receiver, the channel has already changed. It is well known (see, e.g.,[5]) that the ergodic capacity of this (single-user) Gaussian fading channel is given by

$$E_{\boldsymbol{s}}\left(\frac{1}{2}\log(1+sP)\right)\triangleq C(P)\tag{9}$$

where $E_{\boldsymbol{s}}$ denotes expectation over the channel state $\boldsymbol{s}$, and P is the average transmit power.

Notice that $C(P)$ is a monotonically increasing and concave function in P. Hence, it is invertible, with inverse $C^{-1}(r)$ monotonically increasing and convex in r. Reliable communication at rate r is possible if received power is in the set: $\{P : C(P)\ge r\}=\{P : P\ge C^{-1}(r)\}$.

Therefore, the power needed to communicate at rate r is $C^{-1}(r)$. This problem can be written in the style of Problem 5, by defining epochs as packet inter-arrival times, of durations ξ_i, $1\le i\le m$.

Problem 7: Offline Scheduling for the Fast-Fading Channel:

$$\begin{aligned}\text{Minimize}:\ &\sum_{i=1}^{m}\xi_i C^{-1}(r_i)\\ \text{subject to}:\ &\sum_{i=1}^{k}r_i\xi_i\le kB,\quad k=1,\dots,m-1\\ &\sum_{i=1}^{m}r_i\xi_i=mB.\end{aligned}$$

Note that the definition of Problem 7 does not involve channel states. This is because the transmitter does not track the channel state, which is assumed to vary over a codeword resulting in channel capacity to be constant (see (9)). This is unlike the slow-fading case, where we assumed the transmitter can obtain channel state information and code accordingly, and thus the capacity changes in time. Clearly, this problem can be solved using FlowRight. Note, however, that it is much simpler than Problem 5, since we do not need to consider channel state change instants when defining epochs. In fact, this problem is identical to Problem 1 and thus its closed-form solution is given by (2).

At this point, it is quite clear that online scheduling in the fast-fading case can also be done by the look-ahead algorithm: set the rate at time t to $r=BT_s\ q(t)/L$ bits/channel use. This is more straightforward than the slow-fading case as no adaptation to the channel state needs to be performed.

V. Conclusion

Progress in wireless networking has greatly increased the need for transmitting information with minimum energy under reasonable delay constraints. In the study [19], the minimum-energy packet transmission offline scheduling problem under deadline constraint was formulated and solved for a single transmitter–receiver pair. The multiple-user case, when transmission is restricted to time-division was studied in [8]. In this paper, we extended the work in [19] and [8] to transmission scenarios where the channel is time-varying due to interference and fading. We showed that offline scheduling for classes of multiple-access and broadcast channels with and without fading can be reduced to convex optimization problems with linear constraints and devised an algorithm, FlowRight, that finds the optimal offline schedule. Using FlowRight, we were able to find the minimum energy per packet achievable by any algorithm in the uplink scenario. Through simulations, it was observed that the significance of multiple-access coding over using time-division increases as the system gets more loaded.

We devised a heuristic online algorithm, look-ahead water-filling, which adapts to both the channel variation and backlog. It was demonstrated through simulations that significant energy saving can be achieved by such joint adaptation.

In this paper, we addressed the point-to-point, multiple-access and broadcast settings. An interesting direction for future work would be to investigate energy-efficient scheduling for multihop networks. This is of particular interest due to the increasing practical importance of *ad hoc* sensor and mobile networks, where energy conservation is a key design criterion. Finding optimal energy-efficient scheduling algorithms for multihop settings, however, is nontrivial. The optimal transmission rates, which we used in deriving the optimal offline schedules, are not known even for the simplest such setting with a single sender–receiver pair and a single-relay node.

Another direction for future work is formulating and solving the question of *optimal online* scheduling. Recently, [1] has considered the single transmitter–receiver case where the transmitter has a finite buffer, and solved the problem of dynamically assigning rates/powers to packets in order to minimize the long-term average transmission energy subject to an upper bound on the buffer overflow probability. It would be interesting to pursue the generalization of such a dynamic control formulation to multiuser settings.

APPENDIX
PROOFS OF LEMMAS 3 AND 4

A. *Proof of Lemma 3*

For simplicity, we shall drop the superscript and refer to $h^{(b_1,b_2)}(r_{11}, r_{21})$ as $h(r_{11}, r_{21})$. This function is strictly convex in both variables, and $(\hat{r}_{11}, \hat{r}_{21})$ is the result of minimizing it over a bounded region. Hence, the solution is unique. The solution, $(\hat{r}_{11}, \hat{r}_{21})$, is either at the boundaries of the region defined by $0 \le r_{11} \le b_{11}/\xi_1$, $0 \le r_{21} \le b_{21}/\xi_1$, or inside. If it is inside, it must satisfy $h_x(\hat{r}_{11}, \hat{r}_{21}) = 0$ and $h_y(\hat{r}_{11}, \hat{r}_{21}) = 0$.

Due to convexity, partial derivatives of h are monotonic increasing. At the point $\hat{r}_{11} = \hat{r}_{21} = 0$, h_x and h_y are both negative (this can be seen by substituting the values). If $h_x = 0$ is not achieved in the region, then due to monotonicity, $h_x(b_{11}/\xi_1, r_{21}) < 0$ for all permissible values of r_{21}. In this case, increasing the rate r_{11} further than the boundary would decrease total energy, but this cannot be done due to *causality* constraints, so $\hat{r}_{11} = b_{11}/\xi_1$. Similarly, if $h_y = 0$ is not achieved inside the region, then $\hat{r}_{21} = b_{21}/\xi_1$. □

B. *Proof of Lemma 4*

First consider case a), i.e.,

$$h_x(\hat{r}_{11}, \hat{r}_{21}) = 0 \quad \text{and} \quad h_y(\hat{r}_{11}, \hat{r}_{21}) = 0.$$

Define $b'_j = \xi_1 r'_{j1} + \xi_2 r'_{j2}$ for $j = 1, 2$. Due to strict convexity (hence, the monotonicity of the derivative) it can be easily shown that

$$h_x^{(b'_1,b'_2)}(\hat{r}_{11} + \Delta_{11}, \hat{r}_{21} + \Delta_{21}) \ge h_x^{(b_1,b_2)}(\hat{r}_{11}, \hat{r}_{21}) = 0$$

and that the inequality is *strict* unless $b'_1 = b_1$, $b'_2 = b_2$, $\Delta_{11} = 0$, *and* $\Delta_{21} = 0$. But $h_x^{(b'_1,b'_2)}(0,0) < 0$. Hence, energy is uniquely minimized by a point $[\tilde{r}_{11}, \tilde{r}_{21}]$ that satisfies $0 < \tilde{r}_{11} < \hat{r}_{11} + \Delta_{11}$ and $0 < \tilde{r}_{21} < \hat{r}_{21} + \Delta_{21}$. This solution results from *pushing* a nonzero amount of information *right*, from data epoch 1 to data epoch 2. In case b)

$$h_x(\hat{r}_{11}, \hat{r}_{21}) < 0 \quad \text{or} \quad h_y(\hat{r}_{11}, \hat{r}_{21}) < 0.$$

So when the injection and subtraction is done, these derivatives can remain negative or become positive. In the case that they become positive, there will be a right push. If either of these, say h_x, remains negative, then a right push (i.e., decreasing r_{11}) on that stream can only increase the total energy. That stream was shown in part 1 to be limited by causality, and it still is, because injection brought only bits from the left. So we cannot pull any bits from epoch 2 to epoch 1 on this stream (i.e., increase r_{11}). So, there will be no move on this stream. □

ACKNOWLEDGMENT

The authors would like to thank Balaji Prabhakar, Sina Zahedi, and Mayank Sharma for their useful comments, and the anonymous reviewers for their careful reading.

REFERENCES

[1] B. Ata, "Dynamic power control in a wireless static channel subject to a quality of service constraint," *Oper. Res.*, to be published.

[2] R. Berry, "Power and delay trade-offs in fading channels," Ph.D. dissertation, MIT, Cambridge, MA, 2000.

[3] R. Berry and R. Gallager, "Buffer control for communication over fading channels," in *Proc. Int. Symp. Information Theory*, Sorrento, Italy, June 2000, p. 409.

[4] D. P. Bertsekas, *Nonlinear Programming*. Belmont, MA: Athena Scientific, 1995.

[5] E. Biglieri, J. Proakis, and S. Shamai (Shitz), "Fading channels: Information-theoretic and communications aspects," *IEEE Trans. Inform. Theory*, vol. 44, pp. 2619–2692, Oct. 1998.

[6] B. Collins and R. Cruz, "Transmission policies for time varying channels with average delay constraints," in *Proc. 1999 Allerton Conf. Communication, Control and Computing*, Monticello, IL, 1999.

[7] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991, Wiley Series in Telecommunications.

[8] A. El Gamal, C. Nair, B. Prabhakar, E. Uysal-Biyikoglu, and S. Zahedi, "Energy-efficient scheduling of packet transmissions over wireless networks," in *Proc. IEEE INFOCOM*, vol. 3, New York, June 2002, pp. 1773–1782.

[9] A. Fu, E. Modiano, and J. Tsitsiklis, "Optimal energy allocation and admission control for communications satellites," in *Proc. IEEE INFOCOM*, vol. 2, New York, June 2002, pp. 648–650.

[10] R. G. Gallager, *Discrete Stochastic Processes*. Boston, MA: Kluwer Academic, 1995.

[11] A. Goldsmith, "The capacity of downlink fading channels with variable rate and power," *IEEE Trans. Veh. Technol.*, vol. 46, pp. 569–580, Aug. 1997.

[12] A. Goldsmith and S.-G. Chua, "Variable-rate variable-power MQAM for fading channels," *IEEE Trans. Commun.*, vol. 45, pp. 1218–1230, Oct. 1997.

[13] S. Hanly and D. N. C. Tse, "Multi-access fading channels: Part II: Delay-limited capacities," *IEEE Trans. Inform. Theory*, vol. 44, pp. 2816–2831, Nov. 1998.

[14] K. J. Hole, H. Holm, and G. E. Øien, "Adaptive multidimensional coded modulation over flat fading channels," *IEEE J. Select. Areas Commun.*, vol. 18, pp. 1153–1158, July 2000.

[15] R. Knopp and P. A. Humblet, "Information capacity and power control in single-cell multiuser communications," in *Proc. Int. Conf. Communications*, vol. 1, Seattle, WA, June 1995, pp. 331–335.

[16] M. Médard, S. P. Meyn, J. Huang, and A. J. Goldsmith, "Capacity of time-slotted ALOHA packetized multiple-access systems," in *Proc. IEEE Int. Symp. Information Theory*, Sorrento, Italy, June 2001, p. 407.

[17] M. Neely, E. Modiano, and C. Rohrs, "Power and server allocation in a multi-beam satellite with time varying channels," in *Proc. IEEE INFOCOM*, vol. 3, New York, June 2002, pp. 1451–1460.

[18] P. Nuggehalli, V. Srinivashan, and R. R. Rao, "Delay constrained energy efficient transmission strategies for wireless devices," in *Proc. IEEE INFOCOM*, vol. 3, New York, June 2002, pp. 1765–1772.

[19] B. Prabhakar, E. Uysal-Biyikoglu, and A. El Gamal, "Energy-efficient transmission over a wireless link via lazy packet scheduling," in *Proc. IEEE INFOCOM*, vol. 1, Anchorage, AK, Apr. 2002, pp. 386–394.

[20] D. N. C. Tse and S. Hanly, "Multi-access fading channels: Part I: Polymatroid structure, optimal resource allocation and throughput capacities," *IEEE Trans. Inform. Theory*, vol. 44, pp. 2796–2815, Nov. 1998.

[21] E. Uysal-Biyikoglu and A. El Gamal, "Energy-efficient packet transmission over a multiaccess channel," in *Proc. Int. Symp. Information Theory*, Lausanne, Switzerland, June/July 2002, p. 153.

[22] E. Uysal-Biyikoglu, B. Prabhakar, and A. El Gamal, "Energy-efficient packet transmission over a wireless link," *IEEE/ACM Trans. Networking*, to be published.

[23] W. S. Yoon and T. E. Klein, "Delay-optimal power control for wireless data users with average power constraints," in *Proc. 2002 Int. Symp. Information Theory*, Lausanne, Switzerland, June/July 2002, p. 53.