Wireless Scheduling with Hybrid ARQ

Jianwei Huang, Randall Berry, Michael L. Honig
Department of Electrical and Computer Engineering,
Northwestern University, Evanston IL 60208, USA
jianwei@northwestern.edu, {rberry, mh}@ece.northwestern.edu

***Abstract*— We consider channel-aware scheduling for wireless downlink data transmission with hybrid ARQ. Quality of Service requirements for each user are represented by a cost function, which is an increasing function of queue length. The objective is to find a scheduling rule that minimizes the average cost over time. We consider two scenarios: (1) The cost functions are linear, and packets arrive to the queues according to a Poisson process; (2) The cost functions are increasing, convex and there are no new arrivals (draining problem). In each case, we transform the system model into a different model for stochastic scheduling developed by Klimov. Applying Klimov's results, we show that the optimal schedulers for the transformed models in both scenarios are specified by fixed priority rules. The inverse transformation in each case gives the optimal scheduling policy for the original problem. The priorities can be explicitly computed, and in the first scenario are determined by simple closed-form expressions. For the draining problem, we show that the optimal policy never interrupts the retransmissions of a packet. We numerically compare the performance of the optimal scheduling rule with several heuristic rules, and show that a simple myopic scheduling policy, called the $U'R$ rule, achieves near-optimal performance in specific cases.**

I. Introduction

Scheduling in wireless networks has received considerable attention as a means for efficiently providing high speed data service to mobile users. A basic feature in wireless settings is that the channel quality varies across the user population, due to differences in path-loss, as well as fading effects. A variety of such "channel aware" scheduling approaches have been studied recently (e.g., [1]–[4]), and such approaches are part of most recent wireless standards.

In this paper, we study scheduling in a wireless network, taking into account packet retransmissions. Traditionally, the retransmission is accomplished via a standard ARQ (automatic repeat request) protocol, where, if a packet cannot be decoded it is discarded and retransmitted again. Most of the prior work on wireless scheduling either does not consider retransmissions or considers this standard ARQ approach (e.g., [5]). Here, we are interested in hybrid ARQ schemes [6], where the receiver combines all transmissions of a packet to improve the likelihood of decoding success. A variety of hybrid ARQ techniques have been proposed and studied recently (e.g., see [7]–[9]). Techniques based on hybrid ARQ are an integral part of many third generation wireless standards, such as the GSM EDGE system. For our purposes, the key characteristic of these approaches is that each transmission attempt increases the probability of decoding success with an amount depending on the users' channel conditions.

A goal of any wireless scheduling scheme is to balance the users' Quality of Service (QoS) requirements. Here we represent each user's QoS requirements via a holding cost that is an increasing function of the user's queue length, and attempt to schedule transmissions to minimize the overall cost. Prior work on scheduling, which assumes a linear or quadratic cost function that depends on the packet delay is presented in [10]–[12]. For general cost functions, most authors have focused on developing bounds or heuristic policies [11], [13], [14], or show optimality only in the heavy traffic regime [12]. Here, we consider cost functions that depend on queue length instead of packet delay. From Little's Law, this cost function reflects the average delay of the user's packet with stationary traffic.

We consider two scenarios: (a) linear cost functions with Poisson packet arrivals (LPA scheduling problem), and (b) draining problem (no new arrivals) with general increasing convex costs (DC scheduling problem). By transforming both problems into special cases of the classic Klimov scheduling problem [15], we show that the optimal schedulers for the transformed system in both scenarios are specified by fixed priority rules. We can then map the priority rules back to get the optimal scheduling policy for the original problem. These priorities can be explicitly computed, and for the LPA problem, are given by simple closed-form expressions.

For the DC problem, the priority indices can be iteratively computed. We show that the optimal policy never interrupts retransmissions of a packet. We also observe that the DC problem can be formulated as a Markov Decision Process (MDP), analogous to the formulation in [16], and show that the priority increases with queue length. The performance of the optimal scheduling rule is compared numerically with several heuristic rules. A simple myopic scheduler, called the $U'R$ rule [17], achieves near optimal performance for the cases shown.

II. System Model

We consider downlink traffic from a base station to N mobile users. As shown in Fig. 1, packets for each user arrive at the base station according to independent random processes, and are accumulated in N queues until they are served. The

This work was supported by the Northwestern-Motorola Center for Communications, by NSF under grant CCR 9903055, and by CAREER award CCR-0238382.

base station transmits to one user at a time in slots of fixed durations. In each slot, a scheduler decides which one of the N Head of Line (HoL) packet to transmit.

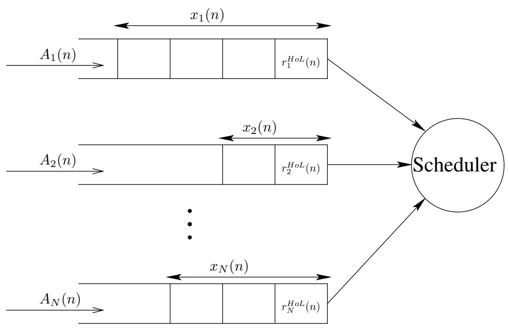

Fig. 1. System Model

If a transmitted packet fails to decode, it stays at the HoL and is immediately available for retransmission. This approximates the case when the feedback delay is small compared to the packet transmission time [9]. Given that a packet for user i has not been successfully decoded in r_i transmission attempts, let $g_i(r_i)$ denote the probability of decoding *failure* for the next transmission. This depends on the specific hybrid ARQ scheme, and on user i's channel conditions. We assume that $g_i(\cdot)$ is time-invariant, which is reasonable in a slow fading environment. In any reasonable hybrid ARQ approach, $g_i(\cdot)$ is nonincreasing in the number of transmission attempts r_i, i.e., $g_i(r_i) \geq g_i(r_i')$ for all $r_i \leq r_i'$. We also assume that there is a maximum number of transmission attempts $r_i^{\max}$ for user i, and that $g_i(r_i^{\max}) = 0$. Note that the special case $g_i(r_i) = g_i(0)$ for all r_i models standard ARQ.

Let $A_i(n)$ denote the arrivals for user i during the nth slot, which is independent of the arrivals of other users. Let $\boldsymbol{S}(n) = \left(r_1^{HoL}(n), ..., r_N^{HoL}(n); x_1(n), ..., x_N(n)\right)$ be the state vector at the nth decision epoch (i.e., the start of the nth time-slot), where $r_i^{HoL}(n) \in \{0, 1, ..., r_i^{\max}\}$ is the number of transmission attempts for the ith HoL packet, and $x_i(n) \in \{0, 1, ...\}$ is the queue length for user i.

Given $\boldsymbol{S}(n)$, the scheduler must determine which HoL packet should be transmitted in the nth slot. A scheduling policy π is defined to be a mapping from each state vector to an index in $\{0, 1, ..., N\}$. If $\pi(\boldsymbol{S}(n)) = i$, the scheduler transmits the HoL packet of queue i; if $\pi(\boldsymbol{S}(n)) = 0$, the scheduler idles and no packet is transmitted. Given a policy π, the state $\boldsymbol{S}(n)$ evolves according to

$$r_i^{HoL}(n+1) = \begin{cases} 0, & \pi(\boldsymbol{S}(n)) = i \text{ and success} \\ r_i^{HoL}(n) + 1, & \pi(\boldsymbol{S}(n)) = i \text{ and failure} \\ r_i^{HoL}(n), & \pi(\boldsymbol{S}(n)) \neq i, \end{cases} \tag{1}$$

and

$$x_i(n+1) = \begin{cases} x_i(n) + A_i(n) - 1, & \pi(\boldsymbol{S}(n)) = i \text{ and success} \\ x_i(n) + A_i(n), & \text{otherwise.} \end{cases} \tag{2}$$

Here "success" and "failure" refer to the decoding outcome for the given transmission. We restrict ourselves to the set of *feasible policies* Π, which contains all nonidling, nonpreemptive, nonanticipative, and stationary policies.[1]

Assume that a cost function $U_i(x_i(n), r_i^{HoL}(n))$ is associated with each user i at the start of the nth slot. We assume that $U_i(x, y)$ is increasing and convex in x, and that $U_i(0,0) = 0$, i.e., there is no holding cost for an empty queue. Different cost functions reflect different QoS requirements or priorities for the users.

We consider two scenarios. In the LPA problem, to be discussed in Section IV, packets arrive to the queues according to independent Poisson processes with rates λ_i, $i = 1, \cdots, N$. The cost function for user i is *linear*, and is given by

$$U_i(x_i(n), r_i^{HoL}(n)) = \begin{cases} c_{i,0}(x_i(n) - 1) + c_{i, r_i^{HoL}(n)} & , x_i(n) > 0 \\ 0 & , x_i(n) = 0 \end{cases}, \tag{3}$$

where c_{i,r_i} is the holding cost rate (cost per unit time per packet) for a packet of user i with r_i transmission attempts. If $r_i < r_i'$, then $0 \leq c_{i,r_i} \leq c_{i,r_i'}$ for all i, i.e., the holding cost increases with the number of retransmissions. We wish to find $\pi \in \Pi$ that minimizes the long-term average expected cost

$$J_{LPA} = \lim_{\tau \to \infty} \frac{1}{\tau} E_\pi \left[\sum_{n=1}^{\tau} \sum_{i=1}^{N} U_i(x_i(n), r_i^{HoL}(n)) \right]. \tag{4}$$

In Sect. V, we consider a draining problem with no new arrivals (i.e., $A_i(n) = 0$ for all i and n). The cost function is allowed to be an arbitrary increasing convex function of the queue length, and independent of the number of transmission attempts, i.e., $U_i\left(x_i(n), r_i^{HoL}(n)\right) = U_i(x_i(n))$. We refer to this as the Draining Convex (DC) problem. Given an initial batch of packets $(x_1(1), ..., x_N(1))$, the goal is to find $\pi \in \Pi$, which minimizes the total expected draining cost, i.e.,

$$J_{DC} = E_\pi \left[\sum_{n=1}^{\infty} \sum_{i=1}^{N} U_i(x_i(n)) \right]. \tag{5}$$

This can also be interpreted as a model for a system with batch arrivals, where the inter-arrival time is long enough to finish each batch before the arrival of a new batch.

III. Klimov Model

The Klimov model [15] has a single non-preemptive server, which is allocated to the jobs in a network of K $M/G/1$ queues. Jobs arrive according to a Poisson process with rate λ, and are assigned to queue m with probability p_m, where $\sum_{m=1}^{K} p_m = 1$. The service time for a job at queue m $(m = 1, .., K)$ has distribution function $B_m(x)$, and finite mean b_m. After service completion at queue m, a job enters queue j $(j = 1, ..., K)$ with probability p_{mj}, or leaves the system with the probability $1 - \sum_{j=1}^{K} p_{mj}$. The transition matrix $P = [p_{mj}, 1 \leq m, j \leq K]$ is such that every job eventually leaves the system, i.e., $\lim_{n\to\infty} P^n = 0$. For stability, the

[1] A policy is nonpreemptive if the transmission of a packet is not interrupted by an arrival, and is nonanticipative if it does not account for future decoding results or arrivals.

arrival rate should not exceed the processing capacity of the system, i.e., $\lambda \boldsymbol{p}(I-P)^{-1}\boldsymbol{b} < 1$, where $\boldsymbol{p} = (p_1, ..., p_K)$ and $\boldsymbol{b} = (b_1, ..., b_K)'$.

The objective is to find a feasible scheduling policy π that minimizes a linear combination of the time-averaged number of jobs at each queue,

$$J_{\text{KM}} = \lim_{\tau \to \infty} \frac{1}{\tau} E_\pi \left[\int_0^\tau \sum_{m=1}^{K} c_m x_m(t) dt \right], \tag{6}$$

where $x_m(t)$ is the number of jobs in queue m at time t and $c_m \geq 0$ is a (linear) holding cost rate for queue m.

The optimal scheduling policy is a fixed-priority rule [15]. This means that a time-invariant priority index can be calculated for each queue, which is independent of the arrival process and queue lengths. At each decision epoch, the server serves a job from the nonempty queue with the highest priority.

The optimal priority indices can be calculated via an iterative algorithm [15], which starts from the set of queues $\Omega = \{1, 2, ..., K\}$ and selects the lowest priority queue at each iteration. Given a subset of queues $M \subset \Omega$, the priority for queue $m \in M$ is determined by $C_m^{(M)} / T_m^{(M)}$, where $C_m^{(M)}$ is the *equivalent holding cost rate*, and $T_m^{(M)}$ is the *average total service time* (not including waiting time) for a job in queue m (i.e., until it exits from M). Since the service times are independent, for each $m \in M$,

$$T_m^{(M)} = \sum_{j \in M} p_{mj} T_j^{(M)} + b_m. \tag{7}$$

The optimal priority indices are computed by the following *Klimov algorithm*:

1) *Initialization:* $M_K = \Omega$, $C_m^{(M_K)} = c_m$ for all $m \in M_K$, $k = K$.
2) Find a queue α_k with lowest priority, i.e.,
$$\alpha_k = \arg\min_{m \in M_k} \left\{ \frac{C_m^{(M_k)}}{T_m^{(M_k)}} \right\}, \tag{8}$$
with ties broken arbitrarily.
3) $M_{k-1} = M_k - \{\alpha_k\}$
4) If $M_{k-1} = \phi$ (null set), then stop. Otherwise, for each $m \in M_{k-1}$, compute
$$C_m^{(M_{k-1})} = T_m^{(M_k)} \left[\frac{C_m^{(M_k)}}{T_m^{(M_k)}} - \frac{C_{\alpha_k}^{(M_k)}}{T_{\alpha_k}^{(M_k)}} \right]. \tag{9}$$
Decrement k and go to step 2.

In this way, the queues are ordered in descending priorities, $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_K$, where $(\alpha_1, \alpha_2, ..., \alpha_K)$ is a permutation of queue indices $(1, 2, ..., K)$. The optimal policy π always assigns the server to the nonempty queue α_k with the smallest index k. Moreover, this scheduler minimizes the total holding cost for each busy period of the system, starting from any initial state [18].

When discussing the DC problem, we consider a variation of the Klimov model, which we call the *Draining Klimov model*. In this case the goal is to find a policy, which minimizes the total expected holding cost for a batch of packets initially in the system with no new arrivals. The priority rule specified by the Klimov algorithm is also optimal for the draining model, since the scheduler minimizes the total holding cost of each busy period. In other words, the draining problem can be viewed as a special busy period with no further arrivals.

IV. The LPA Scheduling Problem

In this section we reformulate the LPA scheduling problem as a special case of Klimov's problem, which we refer as the *LPAK* scheduling problem. We will show that the optimal scheduling policy for the LPAK problem is also optimal for the LPA problem.

A. LPAK Scheduling Problem

The LPAK problem is a relaxation of LPA with respect to the service discipline. In LPA, there is one queue for each user i, and the HoL packet in a queue has priority over all the other packets in the queue. The LPAK problem is illustrated in Fig. 2 for two users with $r_1^{\max} = 2$ and $r_2^{\max} = 1$. There are $r_i^{\max} + 1$ queues for each user i, and each queue is labelled by (i, r_i) for $r_i = 0, ..., r_i^{\max}$, where r_i is the number of transmission attempts for all packets in the queue. There are a total of $K = \sum_{i=1}^{N} (r_i^{\max} + 1)$ queues. For the example in Fig. 2, $K = 3 + 2 = 5$. At each decision epoch, the server decides which of the K HoL packets to serve. Because of the additional queues in the LPAK problem, the HoL packet corresponding to a particular user (in the original LPA problem) does not necessarily have priority over the user's other packets. This relaxation makes LPAK a standard Klimov problem. Subsequently, we will show that the optimal scheduling rule for LPAK still gives priority to the user's HoL packet.

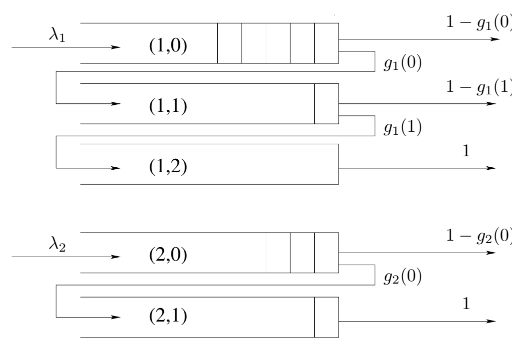

Fig. 2. LPAK System Model

The total arrival process is Poisson with rate $\lambda = \sum_{i=1}^{N} \lambda_i$, and each packet is assigned to queue $(i, 0)$ with probability $p_{i,0} = \lambda_i / \lambda$. The service time for each queue (i, r_i) is deterministic with $b_{i,r_i} = 1$ time slot. The transition probabilities among queues are determined by the probability of decoding failure. That is, after a packet from queue (i, r_i), $r_i < r_i^{\max}$, has been served, it enters queue $(i, r_i + 1)$ with probability $p_{(i,r_i),(i,r_i+1)} = g_i(r_i)$, corresponding to a decoding failure, or leaves the system with probability $1 - g_i(r_i)$, corresponding to a decoding success. After a packet from queue $(i, r_i^{\max})$ has been served, it leaves the system with probability 1. Thus

$$p_{(i,r_i),(j,r_j)} = \begin{cases} g_i(r_i) & , \quad r_i < r_i^{\max}, \quad (j, r_j) = (i, r_i + 1) \\ 0 & , \quad \text{otherwise.} \end{cases} \tag{10}$$

For any set $M \subset \Omega = \{1, \cdots, K\}$ and $(i, r_i) \in M$, the average total service time is

$$T_{i,r_i}^{(M)} = \sum_{(j,r_j)\in M} p_{(i,r_i),(j,r_j)} T_{j,r_j}^{(M)} + 1. \quad (11)$$

The holding cost rate of queue (i, r_i) is c_{i,r_i}, and the number of packets in queue (i, r_i) at the nth decision epoch is $x_{i,r_i}(n)$. The goal is to find a policy $\pi \in \Pi$, which minimizes the time-averaged expected cost

$$J_{LPAK} = \lim_{\tau\to\infty} \frac{1}{\tau} E_\pi \left[\sum_{n=1}^{\tau} \sum_{(i,r_i)\in\Omega} c_{i,r_i} x_{i,r_i}(n) \right]. \quad (12)$$

B. Optimal Policies

For the LPAK scheduling problem, the optimal priority indices can be calculated iteratively using the Klimov algorithm in Sect. III. Consider this algorithm with the following rule used to break any ties that occur in (8): when a tie occurs, set α_k to be the queue (i, r_i) such that for all other queues (j, r_j) in the tie, $j > i$, or $j = i$ and $r_j > r_i$.

Theorem 1: For the LPAK scheduling problem, the optimal scheduling policy is a fixed priority rule in which the priorities, $\alpha_1, \alpha_2, \cdots, \alpha_K$, satisfy

$$\frac{c_{\alpha_1}}{T_{\alpha_1}^{(\Omega)}} \geq \frac{c_{\alpha_2}}{T_{\alpha_2}^{(\Omega)}} \geq \cdots \geq \frac{c_{\alpha_K}}{T_{\alpha_K}^{(\Omega)}}. \quad (13)$$

To derive the optimal LPA scheduler, let $\boldsymbol{R} = \left(r_1^{HoL}, r_2^{HoL}, ..., r_N^{HoL}\right)$ denote the vector of retransmission attempts for HoL packets across the N queues. Let $T_{i,r_i^{HoL}}$ be the expected total service time for user i's HoL packet (not including any waiting time) until it exits the system, which is given by

$$T_{i,r_i^{HoL}} = 1 + \sum_{j=r_i^{HoL}}^{r_i^{\max}-1} \prod_{l=r_i^{HoL}}^{j} g_i(l). \quad (14)$$

Corollary 1: For the LPA scheduling problem, the optimal scheduling rule is to transmit the HoL packet with the highest priority index $c_{i,r_i^{HoL}}/T_{i,r_i^{HoL}}$ among all nonempty queues. Furthermore, the optimal policy is a monotonic threshold policy on the number of transmission attempts, i.e., if it is optimal to transmit user i when $\boldsymbol{R} = \left(r_1^{HoL}, .., r_i^{HoL}, .., r_N^{HoL}\right)$, then it is optimal to transmit user i when r_i^{HoL} is replaced by $r_i'^{HoL} > r_i^{HoL}$.

The optimal LPA scheduling rule depends on the set of holding cost rates c_{i,r_i}, the number of transmission attempts r_i^{HoL}, and the probability of decoding success $g_i(\cdot)$ (i.e., the channel condition) across all users. A higher cost rate, more transmission attempts or a better channel results in a higher priority. Notice that scheduling decisions do not explicitly depend on the arrival processes or queue lengths, although the latter affect the holding costs.

Computing the priority indices via the Klimov algorithm generally requires K iterations with computational complexity $O\left(K^2\right)$. For the LPA problem, due to the special structure of the transition matrix, we can determine the priority indices using (14) directly, with associated complexity $O(K)$. This may be suitable for on-line scheduling with time-varying channel conditions.

We illustrate the optimal scheduling policy with some numerical examples. Consider a system with $N = 2$ users, and probability of decoding failure

$$g_i(r_i) = \begin{cases} \eta_i \cdot 0.5^{r_i} & ; 0 \leq r_i < r_i^{\max} \\ 0 & ; r_i = r_i^{\max} \end{cases}, \quad (15)$$

for $i = 1, 2$. That is, the initial probability of decoding failure is η_i, and is reduced by a half with each retransmission until $r_i = r_i^{\max}$. This type of exponentially decreasing $g_i(r_i)$ is motivated by numerical results in [8].

Fig. 3 shows the optimal scheduling policy as a function of the number of transmission attempts for each user. Parameters are $(\eta_1, r_1^{\max}) = (0.02, 5)$, $(\eta_2, r_2^{\max}) = (0.1, 5)$, $c_{1,r_1} = 1$ (for all r_1) and $c_{2,r_2} = 1.01$ (for all r_2). In this case, user 1 has the better channel, but has a slightly lower holding cost than user 2. As stated in Corollary 1, the optimal scheduling policy is a monotonic threshold policy on r_i^{HoL} $(i = 1, 2)$; the threshold is shown by the solid line in Fig. 3. Comparing this with the dash dotted line $r_1^{HoL} = r_2^{HoL} = r$, when r is small $(r \leq 3)$, user 1 has priority because of the better channel (smaller $T_{i,r}$). However, when r is large $(r > 3)$, user 2 has priority. The reason is that $g_i(r)$ is very small, which makes $T_{i,r}$ very close to 1 for both users. Thus the difference between the cost rates $c_{i,r}$ is the main factor in determining the priority order.

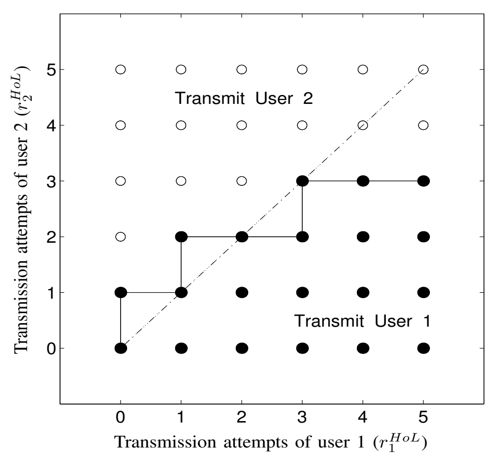

Fig. 3. The optimal scheduling policy as a function of transmission attempts for two users (LPA problem). A dot (circle) means it is optimal to transmit the HoL packet for user 1 (user 2).

Fig. 4 shows the optimal priority orders vs. the holding cost rate of user 2. In this case, both users have the same channel conditions $(\eta_1, r_1^{\max}) = (\eta_2, r_2^{\max}) = (0.05, 2)$. There are six types of packets in the system, (i, r_i), $i = 1, 2$, $r_i = 0, 1, 2$, and their priorities are ordered from 1 (highest) to 6 (lowest). The holding cost rates for user 1 are $c_{1,0} = 0.98$, $c_{1,1} = 1$ and $c_{1,2} = 1.02$. The holding cost rates for user 2 are $c_{2,0} = c_{2,1} = c_{2,2} \triangleq c_2$, which varies from 0.91 to 1.11. Fig. 4 shows that the packet priorities increase with r_i. This reflects the fact that the HoL packet has priority over the other user's packets. At $c_2 = 0.91$, user 1 has priority over user 2. Hence a new packet arrival for user 1 has priority over a retransmission from user 2. Of course, as c_2 increases, the priorities for user 2 increase from lowest $(4, 5, 6)$ to highest $(1, 2, 3)$.

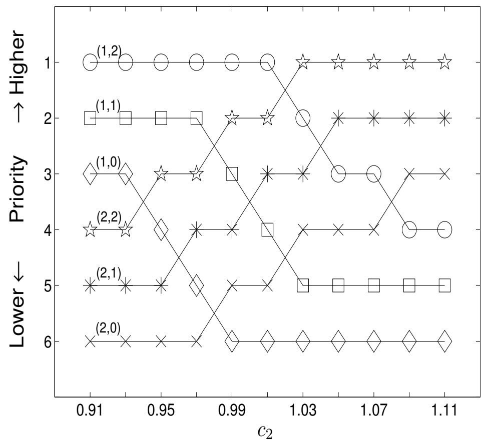

Fig. 4. Optimal priorities vs. holding cost rate c_2 (LPAK problem).

V. The DC Scheduling Problem

For the DC problem, the cost function can be nonlinear, which precludes a direct association with the Klimov model. We circumvent this difficulty by again transforming the problem into a related Klimov problem with more queues. We refer to the latter problem as the DCK (Draining Convex Klimov) scheduling problem.

A. DCK Scheduling Problem

Let A_i be the number of user i's packets initially in the system in the DC model. Each queue in the DC model is replaced by $K_i = (r_i^{\max} + 1) A_i$ queues in the DCK model. Assume, for the DC model, that at time n user i's queue length is $x_i(n) > 0$ with holding cost $U_i(x_i(n))$, and the number of transmission attempts of the HoL packet is $r_i^{HoL}(n)$. In the DCK model, this corresponds to there being *one* packet in the queue $\left(i, r_i^{HoL}(n), x_i(n)\right)$ with *linear* holding cost rate $c_{i, r_i^{HoL}(n), x_i(n)} = U_i(x_i(n))$, and no packets in any of the other $K_i - 1$ queues corresponding to user i.

Let Ω denote the set of all $K = \sum_{i=1}^{N} K_i$ queues in the DCK model. The service time for each queue $(i, r_i, x_i) \in \Omega$ is still deterministic with $b_{i,r_i,x_i} = 1$. Suppose that user i's HoL packet is transmitted during slot n (DC model). Then in the DCK model, the corresponding packet in queue $\left(i, r_i^{HoL}(n), x_i(n)\right)$ either (a) enters queue $\left(i, r_i^{HoL}(n) + 1, x_i(n)\right)$ with probability $g_i\left(r_i^{HoL}(n)\right)$ (decoding fails), (b) enters queue $(i, 0, x_i(n) - 1)$ with probability $1 - g_i\left(r_i^{HoL}(n)\right)$ (decoding succeeds and $x_i(n) > 1$), or (c) leaves the system ($r_i = r_{max}$). The transition probabilities in the DCK model are therefore given by:

$$p_{(i,r_i,x_i),(j,r_j,x_j)} = \begin{cases} g_i(r_i) & , r_i < r_i^{\max}, \quad (j,r_j,x_j) = (i, r_i+1, x_i) \\ 1 - g_i(r_i) & , x_i > 1, \quad (j,r_j,x_j) = (i, 0, x_i - 1) \\ 0 & , \text{otherwise.} \end{cases} \tag{16}$$

The DCK scheduling problem is to find a scheduling policy $\pi \in \Pi$ that minimizes the total expected holding cost for draining all the packets,

$$J_{DCK} = E_\pi \left[\sum_{n=1}^{\infty} \sum_{(i,r_i,x_i)\in\Omega} \mathbf{1}_{i,r_i,x_i}(n) U_i(x_i) \right]. \tag{17}$$

where

$$\mathbf{1}_{i,r_i,x_i}(n) = \begin{cases} 1, & (i,r_i,x_i) \text{ is nonempty in slot } n \\ 0, & (i,r_i,x_i) \text{ is empty in slot } n \end{cases}. \tag{18}$$

The DCK problem is therefore a special case of Klimov's scheduling problem. Hence, we can apply the Klimov algorithm to calculate the optimal priorities, which in turn solves the DC problem. Unlike the LPA problem, for the DC problem the priorities cannot be computed in closed-form. However, we can characterize some basic properties of the optimal policy.

B. Properties of the Optimal Scheduler

As in Sect. IV-B, consider the Klimov algorithm with the following rule to break any tie that occurs in (8): set $\alpha_k = (i, r_i, x_i)$ so that for all other queues (j, r_j, x_j) in the tie, $j > i$.

Theorem 2: The optimal DCK scheduler assigns queue (i, r_i', x_i) higher priority than queue (i, r_i, x_i) for all i, x_i, and $r_i' > r_i$.

This leads to the following result:

Corollary 2: Once the optimal DC scheduler starts to transmit a packet to user i, it continues to transmit the packet until it is successfully decoded.

Note that Corollary 2 is not true for the LPA problem, as shown in Fig. 4. The key difference here is that there are no arrivals which can change the priority orders among the users during a retransmission. Another difference is that the DC optimal scheduler depends on the queue lengths in a complicated way, which depends on the specific choice of cost function.

C. Markov Decision Formulation

The DC problem can be formulated as an MDP. For two users, the state space is $S = \{(r_1, r_2, x_1, x_2) | 0 \le r_i \le r_i^{\max}, 0 \le x_i \le A_i, i \in \{1,2\}\}$, and the action space is $V = \{v_0, v_1, v_2\}$, where v_0 represents idling (no packet in the system), and v_i, $i = 1, 2$, represents transmitting user i's HoL packet. This formulation can be used to prove the following theorem.

Theorem 3: The optimal DC scheduling policy is a monotonic threshold policy with respect to the queue lengths, i.e., if it is optimal to transmit to user i in state (r_1, r_2, x_1, x_2), then it is optimal to transmit to user i in state (r_1, r_2, x_1', x_2') for $x_i' > x_i$ and $x_j' = x_j$ $(j \ne i)$.

VI. Numerical Results

In this section, we compare the optimal LPA and DC scheduling policies with three simple policies, which select the HoL packet of user i^* as follows:

$U'R$ **rule**: $i^*_{U'R} = \underset{1 \le i \le N}{\arg\max}\ U_i'(x_i(n))[1 - g_i(r_i(n))]$, where $U'(\cdot)$ is the derivative of the cost function. This rule takes into account both the user's marginal cost and expected transmission rate, which depends on $g_i(\cdot)$ [17]. It can be shown that this rule is optimal with standard ARQ without packet combining.

Max U' rule: $i^*_{MaxU'} = \arg\max_{1\leq i\leq N} U'_i(x_i(n))$. This rule takes into account only the user's marginal cost, and ignores channel conditions and number of transmissions attempts. This could model a situation where the scheduler has no channel information available.

Max R rule: $i^*_{MaxR} = \arg\max_{1\leq i\leq N} (1 - g_i(r_i(n)))$. This rule maximizes the expected transmission rate without regard to relative costs.

Fig. 5 compares the optimal DC scheduling policy with the preceding heuristic policies. Here we plot the cost per packet vs. channel parameter η_2. The cost functions are $U_i(x_i) = x_i^{\kappa_i}$ where $\kappa_1 = 1.05$, and $\kappa_2 \in \{1.08, 1.15\}$. The channel parameters are $(\eta_1, r_1^{\max}) = (0.01, 2)$ and $r_2^{\max} = 4$, i.e., user 1 has a better channel, but incurs less cost than user 2. The initial queue lengths are $A_1 = A_2 = 40$. Results are shown for both values of κ_2.

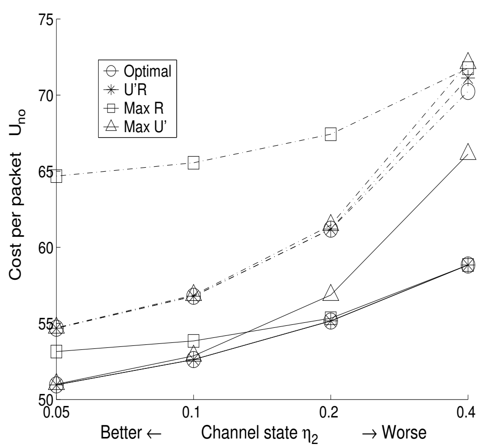

Fig. 5. Comparison of the optimal and heuristic scheduling policies for the DC problem (solid line: $\kappa_2 = 1.08$, dash dotted line: $\kappa_2 = 1.15$)

Fig. 5 shows that the $U'R$ rule performs quite close to the optimal policy. The relative performance of the other policies depend on the users' cost functions. When the cost functions are relatively close (e.g., $\kappa_1 = 1.05$ and $\kappa_2 = 1.08$), scheduling decisions are determined primarily by the probability of decoding success. In this region the Max R rule is nearly optimal (within 5%) and the Max U' rule performs significantly worse (up to 10% higher cost). On the other hand, when $\kappa_1 = 1.05$ and $\kappa_2 = 1.15$, scheduling decisions are determined primarily by the difference between the cost functions. In that case, the Max U' rule is nearly optimal, and the Max R rule performs significantly worse (up to 18% higher cost).

VII. Conclusions

We have considered channel-aware scheduling for wireless downlink data transmission with hybrid ARQ. An optimal scheduler minimizes the total average cost, where the cost function assigned to each user depends on queue length and the number of transmission times for the HoL packet. We characterized the optimal scheduling policies in two situations by transforming these problems so that they fit into the Klimov framework. Namely, with linear cost functions and Poisson arrival processes, the optimal scheduling policy for the transformed problem is a fixed-priority policy. The priority indices can be computed in closed-form, and increase with the number of unsuccessful transmissions. A different transformation is used for the draining problem with general increasing convex cost functions. The optimal scheduling rule for the transformed problem is again a fixed-priority policy, but the priorities must be computed via Klimov's iterative algorithm. In that case, the priorities increase with queue length, and each packet is transmitted continuously until it leaves the system. We also compared the optimal scheduler with several heuristic scheduling rules. Simulation results show that for the cases shown, a simple myopic scheduling policy, the $U'R$ rule, is near-optimal.

References

[1] D. Tse, "Forward link multiuser diversity through proportional fair scheduling," Presentation at Bell Labs, August 1999.

[2] X. Liu, E. K. P. Chong, and N. B. Shroff, "Opportunistic transmission scheduling with resource-sharing contraints in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 10, pp. 2053–2064, Oct. 2001.

[3] M. Andrews, K. Kumaran, K. Ramanan, S. Stolyar, R. Vijayakumar, and P. Whiting, "CDMA data QoS scheduling on the forward link with variable channel conditions," Bell Labs Technical Memorandum.

[4] R. Agrawal, A. Bedekar, R. La, and V. Subramanian, "A class and channel-condition based weighted proprotionally fair scheduler," in *Proc. of ITC 2001*, Salvador, Brazil, Sept. 2001.

[5] S. Shakkottai and R. Srikant, "Scheduling real-time traffic with deadlines over a wireless channel," *ACM/Baltzer Wireless Networks Journal*, vol. 8, no. 1, pp. 13–26, Jan. 2002.

[6] S. Lin and D. Costello, *Error Control Coding - Fundamentals And Applications*. Prentice-Hall, 1983.

[7] A. Banerjee, D. Costello, and T. Fuja, "Diversity combining techniques for bandwidth-efficient turbo ARQ systems," in *IEEE International Symp. on Inform. Th.*, June 2001.

[8] V. Tripathi, E. Visotsky, R. Peterson, and M. Honig, "Reliability-based type II hybrid ARQ schemes," in *ICC'03*, Anchorage, Alaska, USA, May 2003.

[9] G. Caire and D. Tuninetti, "The throughput of hybrid-ARQ protocols for the gaussian collision channel," *IEEE Trans. on Inform. Th.*, vol. 47, no. 5, pp. 20–25, July - Aug 2001.

[10] L. Kleinrock and R. P. Finkelstein, "Time dependent priority queues," *Op. Res.*, vol. 15, pp. 104–116, 1967.

[11] A. Netterman and I. Adiri, "A dynamic priority queue with general concave priority functions," *Op. Res.*, vol. 27, pp. 1088–1100, 1979.

[12] J. A. Van Mieghem, "Dynamic scheduling with convex delay costs: The generalized $c\mu$ rule," *Ann. Appl. Prob.*, vol. 5, no. 3, pp. 809–833, 1995.

[13] U. Bagchi and R. Sullivan, "Dynamic, non-preemptive priority queues with general, linearly increasing priority function," *Op. Res.*, vol. 33, pp. 1278–1298, 1985.

[14] P. A. Franaszek and R. D. Nelson, "Properties of delay-cost scheduling in time-sharing systems," *IBM Journal of Research and Development*, vol. 39, no. 3, pp. 295–314, 1995.

[15] G. P. Klimov, "Time-sharing service systems I," *Th. of Prob. and its Appl.*, vol. 19, no. 3, pp. 558–576, 1974.

[16] E. Visotsky, V. Tripathi, and M. Honig, "Optimum ARQ design: A dynamic programming approach," in *IEEE International Symp. on Inform. Th.*, Yakohama, Japan, 2003.

[17] P. Liu, R. Berry, and M. Honig, "Delay-sensitive packet scheduling in wireless networks," in *IEEE WCNC'03*, New Orleans, LA, March 2003.

[18] P. Nain, P. Tsoucas, and J. Walrand, "Interchange arguments in stochastic scheduling," *J. Appl. Prob.*, vol. 27, pp. 815–826, 1989.