Robust Packet Scheduling in Wireless Cellular Networks

Xiaoqiao Meng*, Zhenghua Fu, Songwu Lu
Computer Science Department
University of California, Los Angeles, CA 90095

Abstract

This paper addresses the following robust scheduling problem: Given that only coarse-grained channel state information (i.e., bounds on channel errors, but not the fine-grained error pattern) is available, how to design a robust scheduler that ensures worst-case optimal performance? To solve this problem, we consider two coarse-grained channel error models and take a zero-sum game theoretic approach, in which the scheduler and the channel error act as non-cooperative adversaries in the scheduling process. Our results show that in the heavy channel error case, the optimal scheduler adopts a threshold form. It does not schedule a flow if the price (the flow is willing to pay) is too small, in order to maximize the system revenue. Among the scheduled flows, the scheduler schedules a flow inversely proportional to the price (the flow is to pay) to minimize the risk of being caught by the channel error adversary. We also show that in the mild channel error model, the robust scheduling policy exhibits a balanced trade-off between a greedy decision and a conservative policy. The scheduler is likely to take a greedy decision if it evaluates the risk of encountering the channel error adversary now to be small. Therefore, robust scheduling does not always imply conservative decision. The scheduler is willing to take "risks" to expect higher gain in some scenarios. Our solution also shows that probabilistic scheduling may lead to higher worst-case performance compared to traditional deterministic policies. Finally, the current efforts show the feasibility to explore a probabilistic approach to cope with dynamic channel error conditions.

I. Introduction

Packet scheduling, which arbitrates packet transmission precedences among multiple contending flows, has long been a popular paradigm to ensure fine-grained service guarantees in the wired network (see [3] for a brief survey). In recent years, researchers have proposed many wireless scheduling algorithms to support service assurance over cellular networks [4] [8][9] [10][11]. These protocols address the issue of error-prone wireless transmissions and typically adopt an adaptive approach. They address the following scheduling problem: assuming fine-grained channel state (i.e., whether the channel is error-free or error-prone at a given time instant) is available, what is the scheduling policy to provide performance bounds (in terms of fairness or network revenue)?

This paper explores a novel approach to wireless scheduling – game-theoretic worst-case optimal scheduling in packet cellular networks. We seek to solve a new scheduling problem: Given that only coarse-grained channel state information (e.g., only bounds on channel errors, but not the exact error pattern that varies over time and location) is available, how to design a scheduler that ensures worst-case performance bounds? The end goal is to maximize the aggregate revenue (collected from all scheduled flows) over all the channel error patterns that fall into the specified error bound. This research is mainly motivated by two factors. The first one is that we would like to relax the requirement for accurate fine-grained (e.g., slot-by-slot) channel estimation. In practice, the wireless channel may exhibit a wide range of error patterns, as we show in Section II. It is nontrivial to have reliable channel state estimation, in particular for indoor mobile users. The second factor is that we intend to explore a novel robust scheduling approach other than the popular adaptive/opportunistic scheme [8], [9], [10], [11], [4]. If the main issue for wireless scheduling is to address the highly dynamic channel error conditions, both adaptive and robust schemes provide valuable solution approaches. While the adaptive scheme has been well studied, the robust approach is not.

To solve the new scheduling problem, we take a zero-sum game theoretic approach. In this approach, the scheduler and the channel error plays a two-player zero-sum game. They act as non-cooperative adversaries in the scheduling process. The scheduler seeks to maximize its aggregate revenue generated by all scheduled flows via scheduling the

*Corresponding author. Tel: +1-310-206-3091; Fax: +1-310-794-5057; Email: xqmeng@cs.ucla.edu

right flow at the right time and minimizing the effect of channel error corruptions. The channel error seeks to minimize the revenue of the scheduler by corrupting the scheduled flow. This is equivalent to a minimax optimization problem that can be solved using game theory techniques. We solve this problems in both single-slot and multi-slot channel error models and give the robust scheduling policy which can ensure the worst-case performance bounds.

Our solution also reveals some interesting insights.

1) In the presence of heavy channel errors (one-slot model), the robust scheduling policy adopts a threshold form. To maximize its revenue, the scheduler will not schedule a flow if the price (that the flow is willing to pay) is too small. However, among the scheduled flows, the scheduler will not select the flow who pays most with the largest probability; instead, the scheduler chooses a flow inversely proportional to the price that the flow is willing to pay. The scheduler employs this conservative policy to minimize the risk being caught by the channel error adversary.
2) In the presence of mild channel errors (multi-slot model), the robust scheduler exhibits a balanced trade-off between the conservative policy and a greedy policy. At each scheduling instant, the scheduler always evaluate the potential penalty incurred by encountering the channel error adversary in the future. When the penalty deems severe, the scheduler is more likely to take a greedy policy at current time. Otherwise, when the penalty is mild, the scheduler is more likely to take the conservative one now. Therefore, robust scheduling does not always imply conservative decision, though its main goal is to provide best worst-case performance. The robust optimal scheduler is willing to take some "risks" in hope for higher gain in some cases.
3) We also explore the probabilistic scheduling policy space. It turns out that, probabilistic scheduling may lead to higher worst-case performance compared with deterministic policies. This worst-case optimal property may benefit some risk-sensitive tasks such as military applications. Moreover, our current effort opens door for future design of probabilistic scheduling approach to cope with channel errors.

The rest of the paper is organized as follows. Section II describes the problem formulation and our general approach. Section III describes our game theoretic approach based on single-slot channel error model. Sections IV extends this game theoretic approach to multi-slot channel error model. Section V evaluates the design via simulations. Section VI discusses the related work and Section VII concludes this paper.

II. Problem Formulation and Game-theoretic Approach

A. Network Model

We consider a wireless cellular network. The scheduler is implemented at the base station and it is responsible for scheduling both uplink (mobile-to-base-station) and downlink (base-station-to-mobile) flows over the shared wireless channel. In our model, we divide time into multiple time slots, which are the units for channel allocation, as in related works [8][1]. The base station can only serve one flow in each slot.

We use u_k to denote the reward that the user is willing to pay for scheduling flow k in one slot. This value can also be viewed as the price of flow k per slot. Due to the difference in user's satisfaction and type of application carried by each flow, the price varies among different flows. In our model, the base station seeks to optimize its total revenue, which is the sum of rewards collected from all the flows being scheduled.

B. Channel Error Model

Recent studies [12][13] have reported a wide range of wireless channel error patterns, from highly bursty to quite sporadic. For example, the measurements in [13] show that each error burst lasts for more than twenty packets, while [12] reports a quite random pattern with the dominant error length as two or three packets. In order to investigate the channel error pattern, we have also done some own measurement. We measured the channel quality in three different locations within a cell in an IEEE 802.11 LAN. Figure 1 shows our measurement results, which indicate that the channel error patterns vary greatly from one location to another, even though they are within a single cell and share the same base station/access point.

Given that the wireless channel may exhibit a wide range of error patterns, we adopt a coarse-grained and elastic channel error model to eliminate the challenging problem of real-time channel state prediction.

[1]The underlying MAC protocols can be either TDMA or CSMA/CA. In CSMA/CA, one slot corresponds to the time spent to complete a RTS-CTS-DATA-ACK handshake.

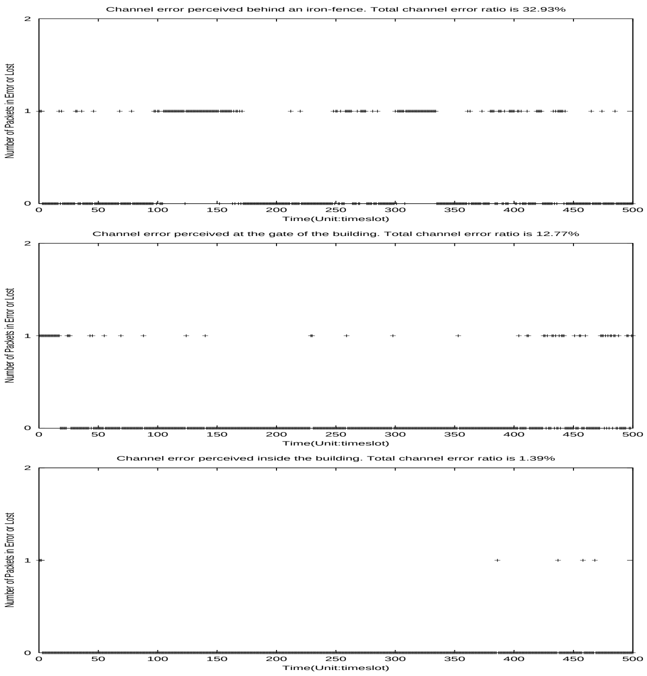

Fig. 1. Different channel error patterns measured in three different locations within a cell

Multi-slot channel error model (n, δ)*:* Given n slots, the scheduler will not perceive channel error in more than δ slots.

The above model does not seek to characterize or predict the exact error pattern. Instead, it gives a statistical estimation on the capability that the channel error may disrupt the transmission. On the other hand, the base station always has to prepare for the worst case, since it has no information on what the real distribution of channel error will be. In this work, we are interested in the scheduling policy of a base station which seeks to optimize its worst case revenue, and we call this the *robust scheduling* policy.

C. Zero-sum Game

We model the interactions between the scheduler[2] and the channel error as a two-player zero-sum game (We refer to [1] for an introduction to the basics of game theory). In this game, the two players, scheduler and channel error, are non-cooperative adversaries against each other. The scheduler always seeks to maximize its total revenue by scheduling m flows, while the channel error (CE) always seeks to minimize the scheduler's total revenue. The two players act simultaneously, and neither of them knows the other player's action in advance. To this end, the CE adversary is empowered to exhibit any error pattern that conforms to our (n, δ) model.

This game theoretic approach to wireless scheduling provides a framework to design optimal worst case scheduling policy without the need of channel state prediction, and enables us to characterize the worst case bound on the total revenue of the scheduler. This is important due to the variety of wireless channel error patterns and the difficulty of wireless channel estimation. As shown later, the game theory framework also stimulates a new family of probabilistic scheduling policies, which can outperform the traditional deterministic policies [8] in terms of their worst-case performance.

We admit that this approach may leads to overly conservative scheduling policy in the normal situation, because the CE player is assumed to be *intelligent* and the real channel error may not match this *intelligence*. However, this worst case optimality is useful when the channel state is error-prone and difficult to predict, and the system wants to make sure its performance will never be worse than a lower bound.

[2]We use 'scheduler' and 'base station' interchangeably in this paper.

	Corrupt flow 1	Corrupt flow 2
Schedule flow 1	0	u_1
Schedule flow 2	u_2	0

TABLE I
GAME MATRIX IN TWO-FLOW SCENARIO

III. SINGLE-SLOT CHANNEL ERROR MODEL AND ROBUST SCHEDULING

Before introducing the (n, δ) error model, we first study a simple single-slot model in this section; this is also the model adopted in a recent work [4]. In this model, we consider every individual time slot in which the scheduler must choose one to transmission from m backlogged flows. We assume that channel error will corrupt one and only one flow among these m flows in each slot. Each flow has an advertised price representing the reward that the user is willing to pay for scheduling this flow per unit slot. The same assumptions have also be used in [5]. We denote the price of flow i as u_i. Without loss of generality, we always assume $u_1 \geq u_2 \geq ... \geq u_m$ in this paper.

We study this single-slot model due to two reasons. First, it models the scenarios in which channel error is heavy. Secondly, due to its simplicity, it can help us to gain some insights on the structure and characteristics of the robust scheduling policy with optimal worst-case performance.

A. 2-flow Case

We start with the simplified two-flow scenario and illustrate the nature of the zero-sum game between the scheduler and the channel error (CE). There are only two flows with prices as u_1 and u_2, respectively ($u_1 \geq u_2$). CE may choose any one of them to corrupt to minimize the scheduler's aggregate revenue, and the scheduler seeks to defeat CE by choosing the other flow in order to maximize its revenue. This can be modeled by a matrix game, as shown in Table I. Each matrix entry denotes the value when the scheduler and CE take the corresponding decisions.

Let us consider the determinstic strategy for the scheduler and CE: the scheduler chooses to schedule either flow 1 or flow 2, while the CE chooses to corrupt either flow 1 or flow 2. Because the scheduler and CE make their decisions simultaneously, if they take such a determinstic strategy, there is no equilibrium point (or so-called saddle point in game-theoretic terminology [1]) in the sense whatever action each player takes, he will regret after the game is over. For example, if the scheduler decides to schedule flow 1 and the CE simultaneously decides to corrput flow 1, the scheduler's gain is zero. Then the scheduler may regret: "If I knew this, I should choose flow 2 to enjoy an outcome of u_2." On the other hand, if the scheduler chooses to schedule flow 1 and the CE chooses to corrupt flow 2, the scheduler gains u_1. CE may regret: "If I knew this, I would have corrupted flow 1 to make the scheduler suffer from an outcome of 0." Therefore, the final policy does not possess an equilibrium property.

We now enlarge the feasible scheduling policy space to a probabilistic setting, which will lead to the equilibrium property in the zero-sum game. In order to decrease the risk that its decision is guessed by the adversary, each player independently chooses both flows with some probability, In game theoretic terminology, this is called the *probabilistic strategy*. The probabilistic strategy for the scheduler is defined as $\{$schedule flow 1 with probability q, schedule flow 2 with $1-q\}$, while the probabilistic strategy for the CE is $\{$corrupt flow 1 with probability p, corrupt flow 2 with $1-p\}$, where $p, q \in [0, 1]$. The expectation of the revenue is given as $f(q, p) = q(1-p)u_1 + (1-q)pu_2$.

The scheduler is the maximizer and CE is the minimizer in the above game, which can be formulated as a minimax problem $\max_q \min_p f(q, p)$. It is easy to solve this problem and get the solution as follows: the optimal strategy for the scheduler is $q^* = \frac{u_2}{u_1+u_2}$; the optimal strategy for the CE is $p^* = \frac{u_1}{u_1+u_2}$, and the scheduler's optimal worst-case revenue is $f^* = \frac{u_1 u_2}{u_1+u_2}$. Note that this revenue is in the statistical sense and represents the expectation of the revenue that the scheduler can achieve in one slot.

We can show that the above probabilistic strategy indeed has an equilibrium property: both players are satisfied with the result after the game is over. A brief explanation for this is as follows. Let $\{q\}$ and $\{p\}$ denote the strategy of the scheduler and CE, respectively. Given q, the worst-case revenue of the scheduler is defined as $f(q) = \min_{p\in[0,1]}(f(p, q)) = \min_p(qu_1 + p(u_2 - q(u_1 + u_2)))$. If the scheduler chooses q to be larger than the optimal strategy, i.e., $q > q^*$, and gives more chance to the flow with higher price, there exists a counter-strategy for the CE as $p = 1$, which yields $f(q) = (1 - q)u_2 < f^*$. This counter-strategy states that CE always chooses to corrupt flow 1, and it makes the worst case revenue of the scheduler less than the optimal value. Similarly, if the scheduler chooses q to be smaller than q^*, its worst case revenue will also decrease. On the other hand, when the scheduler takes the above optimal strategy, the CE can never change p to reduce $f(q^*)$. Therefore, a nice property of the optimal solution, $f(q^*) = f(p, q*) = f^*$ holds for any $p \in [0, 1]$. Any player unilaterally deviates from its optimal strategy will suffer from a gain decrease from its own perspective. Hence, the strategies p^* and q^* reach an equilibrium point. And at this equilibrium point, the scheduler's optimal strategy q^* can ensure the system revenue will be no less than a threshold $f(q^*)$, no matter what the channel error distribution is. In this sense, we call it robust scheduling policy with optimal worst case performance.

B. General Case

In this section we generalize to m flows ($m > 2$). We also consider the different mobile users' exposure to channel errors. We model this exposure level as λ_i, which means if CE decides to corrupt flow i, whether this flow indeed experiences channel error solely depends on the outcome of an independent random coin-flipping event with probability $1 - \lambda_i$ ($0 < \lambda_i < 1$). When λ_i increases, flow i becomes more resistant to channel error (if $\lambda_i = 1$, flow i will be error-free). We assume these λ_i are known *a priori* to the scheduler and CE. Analogously to previous 2-flow case, we model the scheduling problem as a zero-sum game, and the corresponding game matrix is

$$A = \begin{vmatrix} \lambda_1 u_1 & u_1 & u_1 & \cdots & u_1 \\ u_2 & \lambda_2 u_2 & u_2 & \cdots & u_2 \\ u_3 & u_3 & \lambda_3 u_3 & \cdots & u_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ u_m & u_m & u_m & \cdots & \lambda_m u_m \end{vmatrix}_{m \times m} \tag{1}$$

We are again interested in probabilistic strategies. Now the scheduler's total revenue is given as

$$f = \sum_{i=1}^{m} [q_i(1 - p_i)u_i + \lambda_i p_i q_i u_i]$$

where q_i and p_i are the probability with which the scheduler schedules and the CE corrupts flow i, respectively. Let f^* be the optimal worst-case revenue. Since this is a two-player zero-sum game, justified by Minimax theory [1], there should be

$$\begin{aligned} f^* &= \max_{\{q_i\}}(\min_{\{p_i\}} \sum_{i=1}^{m} [q_i(1 - p_i)u_i + \lambda_i p_i q_i u_i]) \\ &= \min_{\{p_i\}}(\max_{\{q_i\}} \sum_{i=1}^{m} [q_i(1 - p_i)u_i + \lambda_i p_i q_i u_i]) \end{aligned} \tag{2}$$

We have solved this minimax problem in Appendix. For clarity of illustration, in Theorem 1, we give the solution when all the λ_i in matrix (1) are equal to zero. This special case does not change solution structure.

Theorem 1: Consider m flows ordered by their prices $u_1 \geq u_2 \geq \ldots \geq u_m$. Let k be the largest integer $2 \leq k \leq m$ such that $\frac{k-2}{u_k} \leq \sum_{i=1}^{k-1} \frac{1}{u_i}$. Then, the scheduler's optimal strategy q_j^* has the form of

$$q_j^* = \begin{cases} \frac{1/u_j}{\sum_{i=1}^{k} 1/u_i} & \text{for } j = 1, \ldots, k \\ 0 & \text{for } j = k + 1, \ldots, m. \end{cases} \tag{3}$$

while for the scheduler, the worst case happens when CE uses p_j to corrupt flow j where

$$p_j^* = \begin{cases} 1 - \frac{(k-1)/u_j}{\sum_{i=1}^{k} 1/u_i} & \text{for } j = 1, \ldots, k \\ 0 & \text{for } j = k + 1, \ldots, m. \end{cases} \tag{4}$$

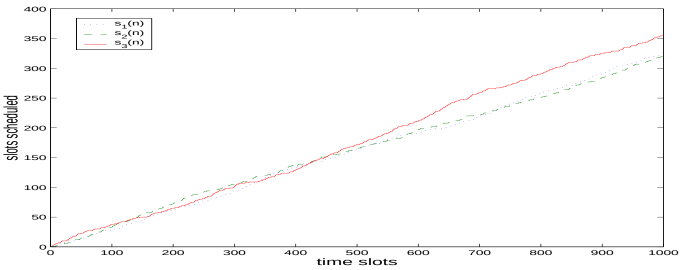

Fig. 2. Probabilistic strategy scheduling in a 3-user system

the optimal worst-case revenue is given as $f^* = \frac{k-1}{\sum_{i=1}^{k} 1/u_i}$.

Theorem 1 gives the expected optimal worst-case revenue for the scheduler. If this matrix game is independently played for a sufficiently large number of times, and the worst-case channel error distribution always happens in each slot , the arithmetic mean of the outcomes earned by the scheduler is equal to the expected optimal worst-case revenue in Theorem 1. Such a single-slot channel error model may be overly conservative, because CE is assumed to be able to attack in each slot while the scheduler always has to be prepared for the worst situation. Thus, such a single-slot error model is suitable for modeling situations where the channel error ratio is high.

Theorem 1 also reveals several interesting points regarding the scheduler's optimal decision. (1) There always exists a *cut-off* price u_k. To achieve the optimal worst-case performance, the scheduler will not schedule flows with prices lower than this u_k. Such a threshold-based scheduling policy reflects the collective agreement involving all the flows in order to maximize the total revenue while taking the least risk. (2) The scheduler also tends to diversify its scheduling decision among multiple flows with prices higher than the threshold u_k. This diversification is employed to be illusive to the CE adversary and will hopefully decrease the risk of being corrupted by CE. (3) Among the flows that the scheduler chooses for scheduling, a flow with higher price has a comparatively lower probability to be scheduled. This counter-intuitive property results from the scheduler's inherent property to achieve both high performance and low risk.

We have provided in Appendix the solution to the general case when $\lambda_i \neq 0$. From the solution, we can see that each λ_i acts as the weight of flow i's price. It does not change the *structure* of the scheduler's robust scheduling policy, and there still exists a threshold u_k playing the same role as in Theorem 1.

C. Long-term Fairness under Concave Utility Function Model

We have heretofore assumed the price of each flow is invariant during the scheduling process. However, if the user's utility is taking an increasing, concave function (*concave* means its first-order differentiation is a decreasing function. This assumption has been widely used in the pricing literature such as [6]), the proposed robust scheduling algorithm can achieve long-term fairness. In this case, the price for each flow is the marginal utility of the user who is perceiving the flow, thus the price function is decreasing with respect to time slot. Theorem 1 turns out that the scheduler will not schedule flows with prices lower than a threshold. However, given a concave utility function model, this does not mean these temporarily starved flows will never be served in the long run.

Let us consider a specific user's utility function $U_i(s) = \log(s_i)$, where s_i represents the amount of slots in which flow i is served. Then its price function, which is the marginal utility, is given as $u_i(s) = \frac{1}{s_i}$. Without loss of generality, we order m users such that their allocated slots are in increasing order $s_1 \leq s_2 \leq \ldots \leq s_m$ (or equivalently, $u_1 \geq u_2 \geq \ldots \geq u_m$). From Theorem 1, we know the cut-off index k satisfies $s_1 + \ldots + s_{k-1} \geq (k-2)s_k$. It means that $\frac{s_k - s_1}{s_k} + \frac{s_k - s_2}{s_k} + \ldots + \frac{s_k - s_{k-1}}{s_k} \leq 1$. Therefore, the service discrepancy ratio among flows is also upper bounded in the long term. We simulate the proposed robust scheduler with 3 backlogged flows, and plot the amount of slots they have received in Figure 2. We observe that the slot allocation difference tends to be upper bounded in the long run.

D. Location Dependent Channel Error

Our framework can be readily extended to accommodate location-dependent channel errors, which are common in wireless networks due to interference, fades and multipath effects. When location-dependent channel errors happen, a subset of backlogged flows which are destined to physically closed mobile users will experience the same error pattern, i.e., they will be corrupted by CE in the same slot. We handle this situation by considering these correlated flows as one group. Whenever any flow is corrupted by CE, all the other flows in the same group will also be corrupted. For a m-flow case, if we have G groups in total, we can show that it is equivalent to a G-flow scenario. Then we can still apply our previous approach.

We illustrate this by a simple example. Considering a $\{m = 3, G = 2\}$ scenario with flows 1 and 2 in a group and flow 3 in the other group, the matrix game is written as Table II.

	Corrupt flow 1	Corrupt flow 2	Corrupt flow 3
Schedule flow 1	0	0	u_1
Schedule flow 2	0	0	u_2
Schedule flow 3	u_3	u_3	0

TABLE II

GAME MATRIX IN $\{m = 3, G = 2\}$ SCENARIO WITH LOCATION-DEPENDENT ERRORS

From Table II, we can see that scheduling flow 1 is always preferable for the scheduler than scheduling flow 2 due to $u_1 > u_2$, while for CE, whether corrupting flow 1 or flow 2 brings in the same damage to the system revenue. Therefore, flow 2 can be eliminated from the consideration of both the scheduler and CE. This simplifies the problem to a $\{m = 2, G = 1\}$ scenario which is well solved by previous approach.

IV. MULTI-SLOT CHANNEL ERROR MODEL AND ROBUST SCHEDULING

In this section we consider a generalized multi-slot (n, δ) error model to catch a wide range of mild channel error patterns. We first divide the time into time windows. Each time window consists of n time slots. The (n, δ) error model means the scheduler perceives channel errors in no more than δ slots in each time window. When $n \gg \delta$, this model provides an elastic constraint on the channel error occurrence, which can accommodate a wide range of error distributions. Thus it is appropriate for modeling scenarios with mild channel error ratio. The goal of this section is to determine the robust scheduling policy under this generalized error model. Again, we formulate the scheduling problem into a matrix game, in which the scheduler seeks to maximize the total revenue over the remaining slots until the end of the time window, while CE seeks to minimize this total revenue.

We still assume m backlogged flows and their prices are sorted by $u_1 \geq u_2 \geq \ldots \geq u_m$. The flow prices are assumed to be invariant during the total n slots. We also assume that the CE adversary can attack at most one flow in each slot.

We start from the first slot of the n-slot time window. We denote the state of a slot as (r, ϵ), where r is the amount of remaining slots till the end of the current time window, and ϵ is the amount of credits for the CE adversary to corrupt during these remaining r slots. For the first slot of the time window, we have $r = n$ and $\epsilon = \delta$.

Similar to previous single-slot matrix game, the scheduler's strategy is in probabilistic form and denoted as $\{q_1, \ldots, q_m\}$ where q_i represents the probability of scheduling flow i. However, the objective function that the scheduler seeks to maximize is no longer the revenue in the current slot, but the sum of the expected revenue over all the remaining r slots. We denote this expected aggregate revenue as $f(r, \epsilon)$. On the other side, the CE adversary seeks to minimize $f(r, \epsilon)$. Note that CE has one more choice than the scheduler, i.e., it can either choose to corrupt any of the m flows or simply delay the corruption and keep this attacking credit for future use. The latter choice may indeed happen if a later corruption can bring more damage to the system revenue. We denote CE's strategy as $\{p_1, \ldots, p_{m+1}\}$, where $p_i (1 \leq i \leq m)$ is the probability that CE chooses flow i to corrupt, and p_{m+1} is the probability that CE defers its corruption in the current slot.

After the current slot expires and both the scheduler and CE have taken their actions, r is decreased by 1. If CE has decided to corrupt any flow, ϵ is decreased by 1 ; otherwise, if CE has decided to defer the corruption, ϵ remains unchanged. In this way, the game moves forward to another state and it will continue until $r = 0$ (after $r = 0$, a new time window will start). Such a procedure is well modeled by a dynamic zero-sum game. Solving this game will give the robust scheduling policy that ensures a bounded worst-case system revenue.

Considering the state (r, ϵ), we write down its game matrix as follows.

$$\begin{vmatrix} f(r-1,\epsilon-1) & f(r-1,\epsilon-1)+u_1 & f(r-1,\epsilon-1)+u_1 & \cdots & f(r-1,\epsilon-1)+u_1 & f(r-1,\epsilon)+u_1 \\ f(r-1,\epsilon-1)+u_2 & f(r-1,\epsilon-1) & f(r-1,\epsilon-1)+u_2 & \cdots & f(r-1,\epsilon-1)+u_2 & f(r-1,\epsilon)+u_2 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ f(r-1,\epsilon-1)+u_m & f(r-1,\epsilon-1)+u_m & f(r-1,\epsilon-1)+u_m & \cdots & f(r-1,\epsilon-1) & f(r-1,\epsilon)+u_m \end{vmatrix}_{m\times(m+1)} \tag{5}$$

Then we formalize this matrix game as a minimax optimization problem and find the optimal solution $\{q_i\}$, $\{p_j\}$, as we have done in the previous single-slot case. Note that we can extract $f(r-1, \epsilon-1)$ from each entry in the matrix and have

$$\begin{vmatrix} 0 & u_1 & u_1 & \cdots & u_1 & u_1-\theta \\ u_2 & 0 & u_2 & \cdots & u_2 & u_2-\theta \\ u_3 & u_3 & 0 & \cdots & u_3 & u_3-\theta \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ u_m & u_m & u_m & \cdots & 0 & u_m-\theta \end{vmatrix}_{m\times(m+1)} \tag{6}$$

where $\theta = f(r-1, \epsilon-1) - f(r-1, \epsilon)$. θ can also be considered as a fixed charge that CE imposes on the scheduler for deferring the attack in the current slot. Based on Lemma 2.2 in [1], we can verify that the optimal strategies $\{q_i\}$, $\{p_j\}$ do not change when we extract $f(r-1, \epsilon-1)$ from the game matrix.

Based on Theorem 1, we give the solution to the game matrix (6) as following

Theorem 2:
1) If $\theta \le (\sum_1^k 1/u_i)^{-1}$ (k is defined in Theorem 1), the scheduler's optimal strategy $\{q_i\}$ and the worst case are given in Theorem 1. Accordingly, the optimal worst-case revenue is $f^* = \frac{k-1}{\sum_{i=1}^k 1/u_i}$.
2) If $\theta \ge u_1$, the scheduler will use greedy algorithm in the sense that it always schedules flow 1 which provides the maximum price. The worst case happens when there is no channel error in the current slot. Accordingly, the optimal worst-case revenue is $u_1 - \theta$.
3) Otherwise if $(\sum_{i=1}^k 1/u_i)^{-1} < \theta < u_1$, we first find d ($2 \le d \le k$) satisfying

$$(\sum_{i=1}^{d} 1/u_i)^{-1} < \theta \le (\sum_{i=1}^{d-1} 1/u_i)^{-1} \tag{7}$$

Then the scheduler's optimal strategy is

$$q_j = \begin{cases} \theta/u_j, & \text{for } j = 1, \dots, d-1; \\ 1 - \theta \sum_{i=1}^{d-1} 1/u_i, & \text{for } j = d; \\ 0, & \text{for } j > d. \end{cases} \tag{8}$$

while the worst case happens when

$$p_j = \begin{cases} 1 - u_d/u_j, & \text{for } j = 1, \dots, d-1; \\ 0, & \text{for } j = d, \dots, m; \\ u_d \sum_{i=1}^{d-1} 1/u_i - (d-2), & \text{for } j = m+1. \end{cases} \tag{9}$$

Accordingly, the optimal worst-case revenue is $f^* = (d-2)\theta + u_d - \theta u_d \sum_{i=1}^{d-1} 1/u_i$, *Proof:* (Sketch) the CE's strategy (9) gives the value for column $j = 1, \ldots, d-1, m+1$ of matrix (6). If $u_j(1-p_j) - \theta p_{m+1} = f^*$ for $j = 1, \ldots, d-1$ and $u_d - \theta p_{m+1} = f^*$ for $j = d$. Using $p_{m+1} = 1 - \sum_1^{d-1} p_j$ and solving these equations for f^* and p_j gives (9). The p_j are nonnegative if $p_{m+1} \geq 0$, which follows since $d \leq k$ and k is defined in Theorem 1. We can show that if the scheduler chooses any of the rows $d+1, \ldots, m$, its average revenue is less than f^* given the CE's strategy (9). For all the rows $j > d$, this average revenue is $u_j - \theta p_{m+1} \leq u_d - \theta p_{m+1} = f^*$. Thus, the CE's strategy keeps the average revenue less than or equal to f^*.

The scheduler's strategy (8) gives the value for rows $j = 1, \ldots, d$ if $\sum_{i=1}^{d} q_i u_i - q_j u_j = f^*$ for $j = 1, \ldots, d-1$ and it gives the value for column $m+1$ if $\sum_{i=1}^{d} q_i u_i - \theta = f^*$. Using $\sum_1^d q_i = 1$ and solving these equations for f^* and q_j gives (8). The q_j are obviously positive for $j = 1, \ldots, d-1$. We must show that if the scheduler uses strategy (8) and CE chooses any of the columns $d, \ldots, m$ to attack, the average revenue will be at least f^*. This average revenue for columns $d+1, \ldots, m$ is $\sum_1^d q_i u_i$, greater than or equal to the average revenue for column d, which is $\sum_1^{d-1} p_i u_i = (d-1)\theta$. The first inequality of (7) implies that this average revenue is greater than f^*, as can be easily proved. Thus the scheduler's strategy keeps the average revenue at least f^*.□

Theorem 2 reveals some interesting features of the robust scheduling policy in the multi-slot channel error model. During each slot, the robust scheduling policy will consider not only the prices of all the backlogged flows, but also the potential penalty $\theta = f(r-1, \epsilon-1) - f(r-1, \epsilon)$, which measures the revenue loss if channel error happens in the next slot. As can be seen from Theorem 2, (1) if this potential penalty is too large compared to the prices of backlogged flows, i.e., $\theta \geq u_1$, the scheduler will make a greedy choice and only schedule the flow with maximum price. Interestingly enough, here the worst case happens when there is no channel error in the current slot. This seems to be counter-intuitive. However, the underlying reason is that it is more profitable for CE to defer its attack to the remaining slots than to corrupt any flow in this slot. (2) If the potential penalty θ is comparatively small, i.e.,$\theta \leq (\sum_1^k 1/u_i)^{-1}$, the scheduler tends to believe the channel error is an inevitable event in this slot, thus it will adopt the conservative scheduling policy, which we have presented in the single-slot case. (3) If the potential penalty is between the previous two extreme cases, the scheduling policy will become a complicated trade-off between the greedy and conservative scheduling policy.

In a summary, in order to achieve optimal performance in the worst case, the scheduler will always evaluate the potential penalty caused by channel error. If this potential penalty is mild, the scheduler is more likely to take conservative policy. Otherwise, the scheduler is more likely to use greedy policy.

A. *Recursive Computation*

To determine the robust scheduling policy in any state (r, ϵ), we need to know $f(r-1, \epsilon-1)$ and $f(r-1, \epsilon)$, which further require to compute $f(r-2, \epsilon-2)$, $f(r-2, \epsilon-1)$ and $f(r-2, \epsilon)$. Such a forwarding recursive procedure will finally stop in those states with pre-determined game value (and strategies for the scheduler and CE). We call these states as *boundary states*. We describe this recursive computation procedure in the tree of Table III. There are two types of boundary states: (1) $(r, 0)$. Such a boundary state means the remaining r slots are error-free. It is trivial to know that the scheduler's optimal strategy is to use greedy algorithm, which always schedules flow 1. So there should be $f(r, 0) = ru_1$. (2) (r, r). This boundary state means channel error will happen in each of the remaining r slots. Thus, in each of the remaining r slots, the scheduler and CE will play a game with single-slot channel error model, as we have addressed in Theorem 1 (Section III).

B. *Algorithms and Time Complexity*

Two approaches can be exploited to compute our robust scheduling policy, and they have different time complexity.

1) In the first approach, all the computations are carried on-line. To determine the scheduling strategy in one slot, a recursive procedure should be applied to determine the game value, as shown in Table III. The total number of intermediate states is $O(n\delta)$. To compute the game value for each state, Theorem 2 is invoked and a time complexity $O(m)$ is required. Thus, the total time complexity is $O(mn\delta)$.
2) In the second approach, before the beginning of the on-line scheduling process, all the intermediate states of (n, δ) are solved and stored. During the on-line scheduling process, the scheduler first determines the current

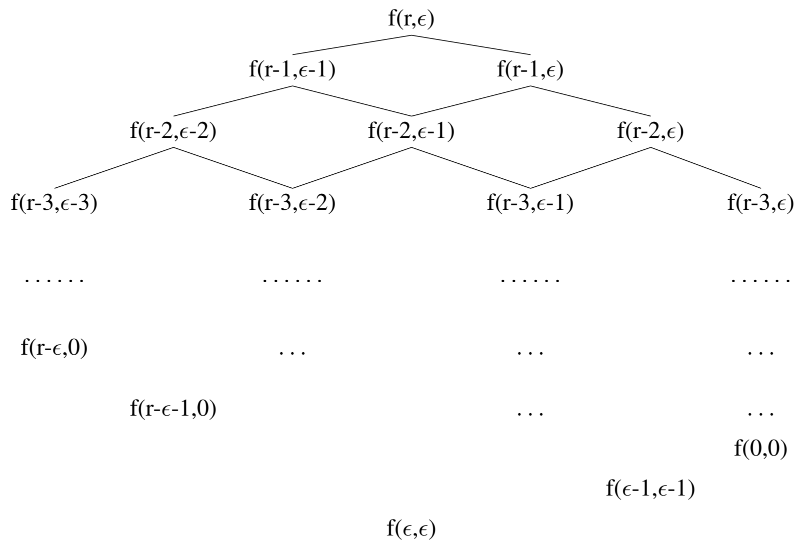

TABLE III

RECURSIVE COMPUTATION OF $f(r, \epsilon)$

state (r, ϵ), then it directly retrieves $f(r-1, \epsilon-1)$, $f(r-1, \epsilon)$ and invokes Theorem 2 to solve the game. Such an off-line approach is more appropriate for real-time requirements but has higher memory consumption.

The pseudo-code for the second approach is given in Algorithm 1.

Algorithm 1 Robust Scheduling with Multi-slot Channel Error Model

Require: $\{m$(number of flows), n(slot number in one session), δ(credits for the channel error to attack$\}$

$\{$Before on-line scheduling, do the following off-line computation$\}$
Compute_boundary_conditions(n,δ)
for $t = 1$ to δ **do**
  for $e = t + 1$ to n **do**
    solve matrix game $\Gamma(t, e)$ based on Theorem 2
    record $\{q_i, i = 1, \ldots, m\}$ for state (t, e)
  end for
end for
$t \leftarrow n$ $\{$beginning of on-line scheduling$\}$
$e \leftarrow \delta$
while $t > 0$ **do**
  retrieve $\{q_i\}$ for current state (t, e)
  Do probabilistic scheduling based on $\{q_i\}$
  if slot_in_error() **then** $\{$perceive a channel error$\}$
    $e \leftarrow e - 1$$\{$decrease e until zero$\}$
  end if
  $t \leftarrow t - 1$
end while

V. SIMULATION EVALUATION

We use simulations to compare the performance of the robust algorithm with other two algorithms, i.e., the greedy scheduler and the adaptive scheduler [8]. For the robust algorithm, we only consider the multi-slot error model case, as in Section IV. The greedy scheduler, as we have considered in Section IV, is defined as always scheduling the flow with the maximum price, regardless of the channel state of the flow. This greedy algorithm can achieve high revenue in error-free case, yet its worst-case performance is sensitive to concrete error distributions. The adaptive scheduler, on the other hand, only schedules the flow with the maximum price among all the flows that perceive a clean channel. In order to dertermine whether a flow perceives a clean channel is based on a one-step prediction, i.e., the channel state

	Lowest	Highest	Average	Variance
Robust Scheduling	51.27	61.39	58.94	1.13
Greedy Scheduling	36.69	65.95	60.43	2.06
Adaptive Scheduling	36.54	67.97	62.59	2.42

TABLE IV

PERFORMANCE COMPARISION AMONG ROBUST, GREEDY AND ADAPTIVE SCHEDULERS

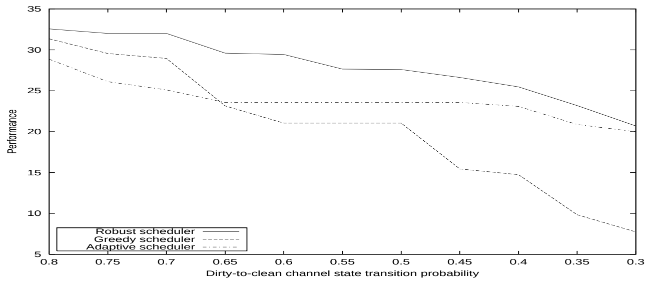

Fig. 3. Worst-case Performance for robust, greedy and adaptive algorithms (*worst-case* is only defined on the 1000 simulations runs)

in previous slot. Such a one-step prediction implicitly assumes the channel error is bursty and it may potentially yield worse performance in less temporal correlated error patterns ([8] also uses this one-step prediction). We believe that such an adaptive algorithm represents a wide range of adaptive scheduling paradigms.

The above three algorithms are evaluated under a wide range of channel error behavior. To this end, we implemented a two-state Markov model as the channel error generator. The two-state Markov error model is configured as follows. For each flow k, p_e^k is the probability of perceiving an error given that the current channel state is *clean*, and p_g^k is the probability of perceiving a clean channel given that the current state is in *dirty*. Therefore, the average probability of perceiving channel error for this flow is $P_E^k = \frac{p_e^k}{p_e^k + p_g^k}$.

We simulate 4 flows ($m = 4$), and the time window is set to be 100 slots ($n = 100$). The aggregate user utility function is chosen as $U(s) = s^\alpha$, where s is the number of slots being scheduled so far. At the beginning of each time window, s is initialized to be 1. Obviously, the price function, which is the marginal utility, is $u(s) = \alpha^{\alpha-1}$. For the four flows, we let α be 0.9, 0.8, 0.7 and 0.6 respectively.

We first use the two-state Markov error model. The two parameters p_e^k, p_g^k are configured as 0.1, 0.8 respectively. We run the simulation for 1000 times (each time there are 100 time slots). The three schedulers' performance are presented in Table IV. It shows that, among the 1000 simulation runs, the lowest performance achieved by the robust scheduler is 51.27, which is 40% higher than that of greedy and adaptive schedulers. This better *worst-case* performance is also reflected from the smaller performance variance of the robust scheduler. However, the average performance of the robust scheduler is 61.39, slightly worse than the greedy and adaptive schedulers.

We then keep p_e^k (probability of channel state transition from clean to dirty) as 0.8 and decrease p_g^k (probability of channel state transition from dirty to clean) from 0.8 to 0.3. Such a decrease indicates a more bursty channel error distribution. As we can observe from Figure 3, the worst-case performance for all these three schedulers decrease as the channel error distribution becomes more bursty. However, the worst-case performance of the robust scheduler decreases much more gracefully. We also notice that there exists a critical p_g^k. Above this critical p_g^k, the adaptive scheduler attains a consistently better worst-case performance than the greedy scheduler. This phenomenon is due to the failure of the one-step prediction in bursty channel error case.

VI. RELATED WORK

Packet scheduling over wireless cellular networks has been an active research topic in recent years. The most popular approach is to devise various fair scheduling algorithms to provide performance bounds in terms of delay, throughput and fairness. These include IWFQ [8], CIF-Q [9], SBFA [10] and WFS [11]. The goal of these wireless fair scheduling algorithms has been to hide short bursts of location-dependent channel errors from well-behaved flows by dynamically swapping channel allocations between backlogged flows that perceive channel errors and backlogged flows that do not, with the intention of reclaiming the channel access for the former when it perceives a clean channel. Therefore, lagging flows (that lag behind their error-free reference service due to channel errors) will receive compensation from leading flows. All these algorithms rely on accurate channel state estimation for every slot, and their performance may suffer in the presence of wide range of error patterns. Our approach does not rely on channel state prediction, and is robust in the presence of wide range of channel error patterns. Another recent work [4] studies utility-based fair scheduling. However, their design still depends on channel state estimation. All these designs are different from our worst-case optimal approach which does not estimate channel state.

Game theoretic approaches have been applied in networking research in past ten years. In [7], the author studies packet scheduling in wireline networking context and formulates the problem into a nonzero-sum game, in which each of multiple selfish users seeks to maximize its own utility. This is different from our work: it does not consider channel errors, thus the nonzero-sum game is fundamentally different from our zero-sum gamef. Besides, it does not consider any probabilistic scheduling policy, which is another contribution of our work. To the best of our knowledge, we have not seen any previous work to study wireless packet scheduling from a game theoretic perspective.

VII. Conclusions

The main design issue for wireless scheduling is to provide service assurances in the presence of dynamic channel error conditions. Current wireless scheduling algorithms [4]-[11] typically take an adaptive approach in which the scheduler adapts its scheduling decision based on the estimated fine-grained channel state. This paper explores an alternative approach to handle channel errors — game-theoretic, robust packet scheduling in wireless cellular networks. In this approach, the scheduler and the channel error play a zero-sum game and act as adversaries in the scheduling process. By solving the corresponding minimax optimization problem, we characterize the worst-case optimal scheduling policies. The merits of this approach include requirements of only coarse-grained channel state, worst-case optimal performance, guaranteed service in the presence of a wide range of error patterns, and balanced trade-off between conservative and greedy policies. Though this work is mainly of theoretic merit, it opens door to further exploring probabilistic scheduling approach to handle channel errors. Ongoing work seeks to further compare the robust and adaptive approaches.

References

[1] T. Basar and G.J. Olsder, Dynamic noncooperative game theory, *Academic Press*, 1998.

[2] P.R.Thie, An introduction to linear programming and game theory, *John Wiley & SONS Inc.*, 2rd Edition, 1988.

[3] H. Zhang, Service disciplines for guaranteed performance service in packet-switching networks, in *Proceedings of the IEEE*, 83(10), pp. 1374-1396, October 1995.

[4] X. Liu, E.K.P. Chong and N.B.Shroff, Transmission scheduling for efficient wireless utilization, *IEEE INFOCOM'01*, May 2001.

[5] M.Agarwal and A.Puri, Base station scheduling of requests with fixed deadlines, *IEEE INFOCOM'02*, June 2002.

[6] F.P.Kelly, A.Maulloo and D.Tan, Rate control for communication networks: shadow prices, proportional fairness and stability, *Journal of the Operational Research Society 49*, pp.237-252, 1998.

[7] S. Shenker, Making greed work in networks: a game-theoretic analysis of switch service disciplines, *ACM SIGCOMM'94*, 1994.

[8] S. Lu, V. Bharghavan and R. Srikant, Fair scheduling in wireless packet networks, *IEEE Trans. on Networking*, August 1999.

[9] T.S. Ng, I. Stoica and H. Zhang, Packet fair queueing algorithms for wireless networks with location-dependent errors, *IEEE INFOCOM'98*, March 1998.

[10] P. Ramanathan and P. Agrawal, Adapting Packet Fair Queueing Algorithms to Wireless Networks, *ACM MOBICOM'98*, October 1998.

[11] S. Lu, T. Nandagopal, and V. Bharghavan, Fair scheduling in wireless packet networks, *ACM MOBICOM'98*, October 1998.

[12] G. T. Nguyen, R. H. Katz, B. Noble, and M. Satyanarayan, A trace-based approach for modeling wireless channel behavior, in *Proceedings of Winter Simulation Conference*, pp. 597-604, December 1996.

[13] D. Eckhardt and P. Steenkiste, A trace-based evaluation of adaptive error correction for a wireless local area network, to appear in *Mobile Networks and Applications (MONET)*, Special Issue on Adaptive Mobile Networking and Computing.

VIII. Appendix - Proof of Theorem 1 (Sketch)

In this section we solve the problem when both the scheduler and CE are playing a zero-sum game with single-slot channel error model. The game matrix is given in (1).

Before solving the game matrix, we first write down a fundamental theorem for solving linear programming problem [2].

Theorem 3: **(Optimality Criterion)** Given a linear programming problem as following
minimize z when

$$\begin{array}{ccccccc}
y_1 + & & \cdots + & a_{1,m+1}y_{m+1} + & \cdots + & a_{1,n}y_n & = & b_1 \\
& y_2 + & \cdots + & a_{2,m+1}y_{m+1} + & \cdots + & a_{2,n}y_n & = & b_2 \\
& \ddots & & & & & & \\
& & y_m + & a_{m,m+1}y_{m+1} + & \cdots + & a_{m,n}y_n & = & b_m \\
& & & c_{m+1}y_{m+1} + & \cdots + & c_n y_n & = & z
\end{array}$$

where $b_i \geq 0$ $(i = 1, \ldots, m)$ and $x_i \geq 0$ $(i = 1, \ldots, n)$.

The minimal value of the objective function is 0 and is attained at the point $(b_1, b_2, \ldots, b_m, 0, 0, \ldots, 0)$ if $c_j \geq 0(j = m+1, \ldots, n)$.

Now we solve the game corresponding to matrix (1). According to [2], it can be rewritten into the following linear programming problem.

Minimize z
subject to constraints

$$\begin{array}{lcl}
\lambda_1 u_1 p_1 + u_1 p_2 + \cdots + u_1 p_i + \cdots + u_1 p_m & \leq & z \\
u_2 p_1 + \lambda_2 u_2 p_2 + \cdots + u_2 p_i + \cdots + u_2 p_m & \leq & z \\
\cdots\cdots & & \\
u_m p_1 + u_m p_2 + \cdots + u_m p_i + \cdots + \lambda_m u_m p_m & \leq & z \\
p_1 + p_2 + \cdots + p_i + \cdots + p_m & = & 1
\end{array}$$

For every i-th $(i = 1, \ldots, m)$ inequality in the above problem, we bring in a positive slack variable d_i and use $y_i = \frac{p_i}{z}$ to replace p_i. We also use the fact that *Minimize* $-y_1 - y_2 - \ldots - y_m$ is equivalent to *Maximize* $y_1 + y_2 + \ldots + y_m$. In this way the problem is simplified as

Minimize $-y_1 - y_2 - \ldots - y_m$
subject to constraints

$$\begin{array}{lcl}
\lambda_1 y_1 + y_2 + \cdots + y_i + \cdots + y_m + d_1 & = & \frac{1}{u_1} \\
y_1 + \lambda_2 y_2 + \cdots + y_i + \cdots + y_m + d_2 & = & \frac{1}{u_2} \\
\cdots & & \\
y_1 + y_2 + \cdots + y_i + \cdots + \lambda_m y_m + d_m & = & \frac{1}{u_m}
\end{array}$$

For easy of illustration, we express the above inequalities in the following tableau.

d_1	d_2	d_3	$\ldots$	d_m	y_1	y_2	y_3	$\ldots$	y_m	
1	0	0	$\cdots$	0	λ_1	1	1	$\cdots$	1	$\frac{1}{u_1}$
0	1	0	$\cdots$	0	1	λ_2	1	$\cdots$	1	$\frac{1}{u_2}$
0	0	1	$\cdots$	0	1	1	λ_2	$\cdots$	1	$\frac{1}{u_3}$
		$\ldots$						$\ldots$		$\ldots$
0	0	0	$\cdots$	1	1	1	1	$\cdots$	λ_m	$\frac{1}{u_m}$
0	0	0	$\cdots$	0	-1	-1	-1	$\cdots$	-1	0

Our goal is to convert the negative coefficients in the last row of the above tableau to be positive through variable elimination. Such a technique is often called *pivot operation* in Simplex method. We first use the second equation in the tableau as reference to eliminate y_1 from all the other equations. The tableau becomes

d_1	d_2	$\dots$	d_m	y_1	y_2	y_3	$\dots$	y_m	
1	$-\lambda_1$	$\cdots$	0	0	$1-\lambda_1\lambda_2$	$1-\lambda_1$	$\cdots$	$1-\lambda_1$	$\frac{1}{u_1}-\frac{\lambda_1}{u_1}$
0	1	$\cdots$	0	1	λ_2	1	$\cdots$	1	$\frac{1}{u_2}$
0	-1	$\cdots$	0	0	$1-\lambda_2$	λ_3-1	$\cdots$	0	$\frac{1}{u_3}-\frac{1}{u_2}$
	$\dots$						$\dots$		$\dots$
0	-1	$\cdots$	1	0	$1-\lambda_2$	0	$\cdots$	λ_m-1	$\frac{1}{u_m}-\frac{1}{u_2}$
0	1	$\cdots$	0	0	λ_2-1	0	$\cdots$	0	$\frac{1}{u_2}$

Since $\lambda_2 < 1$, there still exists a negative coefficient $\lambda_2 - 1$ in the last row. However, if $\frac{1}{u_1} - \frac{\lambda_1}{u_2} > 0$, we can use the first row in the tableau as reference to eliminate y_2 from all the other rows, and it can be easily verified that this operation will convert all the coefficients in the last row to be positive. Thus we achieve the ideal form in Theorem 3 and we can directly apply Theorem 3 to determine the solution. Otherwise, if $\frac{1}{u_1} - \frac{\lambda_1}{u_2} < 0$, we continue previous pivot operation, and this time choose the third row as reference to eliminiate y_2 from all the other rows. Due to space limitation, we skip the detailed description of the procedure and only give the final solution as following:

1) if $\frac{1}{\lambda_1 u_1} < \frac{1}{u_2}$, the optimal strategy for the scheduler is

$$q_j = \begin{cases} 1, & \text{for } j = 1; \\ 0, & \text{for } j = 2, \dots, k. \end{cases} \tag{10}$$

Accordingly, the scheduler's optimal reward is $f^* = \lambda_1 u_1$.

2) if

$$\begin{cases} \frac{1}{\lambda_1 u_1} & \geq & \frac{1}{u_2} \\ \frac{1}{(1-\lambda_1)u_1} + \frac{1}{(1-\lambda_2)u_2} & < & \frac{1-\lambda_1\lambda_2}{(1-\lambda_1)(1-\lambda_2)}\frac{1}{u_3} \end{cases}$$

the optimal strategy for the scheduler is

$$q_j = \begin{cases} \frac{(1-\lambda_2)u_2}{(1-\lambda_2)u_2+(1-\lambda_1)u_1}, & \text{for } j = 1; \\ \frac{(1-\lambda_1)u_1}{(1-\lambda_2)u_2+(1-\lambda_1)u_1}, & \text{for } j = 2; \\ 0, & \text{for } j = 3, \dots, m. \end{cases} \tag{11}$$

Accordingly, the scheduler's optimal reward is

$$f^* = \frac{u_1 u_2 (1-\lambda_1\lambda_2)}{(1-\lambda_2)u_2 + (1-\lambda_1)u_1}$$

3) if there is a *cut-off* k $(2 \leq k \leq m-2)$ s.t.

$$\begin{cases} \sum_{i=1}^{k} \frac{1}{(1-\lambda_i)u_i} & \geq & \left[\sum_{i=3}^{k} \frac{1}{1-\lambda_i} + \frac{1-\lambda_1\lambda_2}{(1-\lambda_1)(1-\lambda_2)}\right] \frac{1}{u_{k+1}} \\ \sum_{i=1}^{k+1} \frac{1}{(1-\lambda_i)u_i} & < & \left[\sum_{i=3}^{k+1} \frac{1}{1-\lambda_i} + \frac{1-\lambda_1\lambda_2}{(1-\lambda_1)(1-\lambda_2)}\right] \frac{1}{u_{k+2}} \end{cases}$$

the optimal strategy for the scheduler is

$$q_j = \begin{cases} \frac{\prod_{\substack{i=1 \\ i\neq j}}^{k} (1-\lambda_i)u_i}{\sum_{i=1}^{k} \prod_{\substack{l=1 \\ l\neq i}}^{k} (1-\lambda_l)u_l}, & \text{for } j = 1, \dots, k; \\ 0, & \text{for } j = k+1, \dots, m. \end{cases} \tag{12}$$

Accordingly, the scheduler's optimal reward is

$$f^* = \frac{\prod_{i=1}^{k} u_i \left[-\prod_{i=1}^{k}(1-\lambda_i) + \sum_{i=1}^{k} \prod_{\substack{l=1 \\ l \neq i}}^{k} (1-\lambda_m) \right]}{\sum_{i=1}^{k} \prod_{\substack{l=1 \\ l \neq i}}^{k} (1-\lambda_l) u_l} \tag{13}$$

4) if $\sum_{i=1}^{m-1} \frac{1}{(1-\lambda_i)u_i} \geq \left[\sum_{i=3}^{m-1} \frac{1}{1-\lambda_i} + \frac{1-\lambda_1\lambda_2}{(1-\lambda_1)(1-\lambda_2)} \right] \frac{1}{u_m}$, the optimal strategy for the scheduler is (12) with $k=m$, and the scheduler's optimal reward is (13) with $k=m$.