A New Wireless Packet Scheduling Algorithm based on the CDF of User Transmission Rates

Daeyoung Park, Hanbyul Seo, Hojoong Kwon, and Byeong Gi Lee
School of Electrical Engineering, Seoul National University

Abstract—We present a new wireless scheduling algorithm based on the cumulative distribution function (cdf) of user transmission rates and also present a simple modification of it to limit the maximum starving time. This cdf-based scheduling (CS) algorithm selects the user for transmission based on the cdf of user rates in such a way that the user whose rate is high enough but least probable to become higher is selected. It turns out that the cdf-based scheduling algorithm is equivalent to a scheduling algorithm that regards the user rates as independent identically distributed (i.i.d.) and the average throughput of a user is independent of other users’ probability distribution. A distinctive feature of this proposed algorithm is that the exact user throughput can be evaluated if the user’s own distribution is known. The cdf-based scheduling with starving-time limitation (CS-STL) algorithm turns out not to affect the average inter-service time but to limit the maximum starving time at the cost of a negligible throughput loss.

Keywords—Multiuser Diversity, wireless scheduling, starving time, inter-service time.

I. INTRODUCTION

Due to rapid growth of the needs for high-rate data applications, efficient wireless packet scheduling schemes are demanding. The High Data Rate (HDR) system [1] allocates a time-slot to the currently ‘best’ user having the maximum service rate in the current channel condition. Since each user is served when its channel condition falls in the relatively best time slot, the overall throughput of the system becomes a maximum. In order to fully exploit such multiuser diversity [2], it is important to set a good scheduling algorithm that can maximize the system throughput and, in addition, can accomplish a ‘fair’ resource share among users as well.

A simplest scheduling algorithm may be the one that selects the user for which the supportable rate is the highest but in this case it can happen to select some particular users more frequently than other users due to uncontrollable channel conditions, thus causing a critical fairness problem. In order to improve such maximum rate algorithm [3], there have been proposed several other scheduling algorithms for use in HDR systems [4]–[6]. Each of these algorithms intended to optimize the system performance while satisfying its own-defined fairness criterion. The scheduling algorithm by Jalali et al. [4] was proposed to satisfy the so-called proportional fairness, the algorithm by Liu et al. [5] tried to maximize the long-term average total throughput for a given time fraction, and the scheduling algorithm by Borst et al. was made to be Pareto-optimal [6, 7].

In this paper, we are going to introduce a new wireless scheduling algorithm, called cdf-based scheduling (CS) algorithm that provides a fair allocation of transmission time fraction. The CS algorithm schedules the user whose channel condition is at its best state and is least likely to be better. We will investigate the properties of the CS algorithm, including that the average throughput of a user is independent of the other users’ probability distribution. In addition, we will derive the exact average throughput for the case when the user rate is Gaussian distributed. Further, we will consider how to extend the CS algorithm for use in delay-sensitive applications by limiting the number of consecutively unserved slots without sacrificing average throughput much.

II. SYSTEM MODEL AND SCHEDULING ALGORITHMS

We consider the HDR downlink channel [1], in which each mobile station (MS) measures the downlink channel, computes the maximum possible rate that the downlink can support, and feeds this rate information back to the base station (BS). Based on the rate information, the BS determines which user (i.e., MS) to serve among K users at the next time slot and transmits a packet to that user.

Let $R_k(n)$ denote the maximum available transmission rate of user $k$, $k = 1, 2, \cdots, K$, at time slot $n$; $k^*(n)$ the user selected for transmission at slot $n$; $T_k(n)$ the average throughput of user $k$ until slot $n$; and $T_k$ the long-term average throughput of user $k$. We assume each $R_k(n)$ is an independent and stationary random sequence. The existing scheduling algorithms may be represented as follows:

- **Maximum rate (MR) algorithm** [3]

  $$k^*(n) = \arg \max_k R_k(n)$$  \hspace{1cm} (1)

- **Proportional fairness (PF) algorithm** [4]

  $$k^*(n) = \arg \max_k \frac{R_k(n)}{T_k(n)}$$  \hspace{1cm} (2)

- Algorithm by Liu et al. [5]

  $$k^*(n) = \arg \max_k (R_k(n) + v_k)$$  \hspace{1cm} (3)

- Algorithm by Borst et al. [6]

  $$k^*(n) = \arg \max_k w_k R_k(n)$$  \hspace{1cm} (4)
All the scheduling algorithms other than the PF algorithm depend not on the history of the user selection but on the current user rate only. We may express such memoryless scheduling algorithms in the general form

\[ k^*(n) = \arg \max_k g_k(R_k(n)) , \tag{5} \]

where \( g_k(x) \) is a non-decreasing function that does not depend on the history of the user selection. So, a memoryless scheduler may be characterized by the set of the functions \( \{ g_k(x) | k = 1, 2, \ldots, K \} \).

\section*{III. CDF-based Scheduling (CS) Algorithm}

It is not feasible, in general, to evaluate the exact throughput of a scheduling algorithm when the user rates are not i.i.d. If we could devise a new scheduler that would yield an exact estimation of the user throughputs even for the case of non-i.i.d. user rates, it would not be difficult to investigate the effects of schedulers on other related protocols (e.g., call admission control). In this context, we consider a new scheduling algorithm that can possibly render a means to estimate the throughputs.

We first assume that a scheduler knows the probability density function (pdf) of each user’s rate. \(^1\) We denote by \( F_{R_k}(r) \) and \( f_{R_k}(r) \) the cdf and the pdf, respectively, of user \( k, k = 1, 2, \ldots, K \).

Let \( \hat{R}_k \) be the root of \( F_{R_k}(r) = 1 - 1/K \) (see Fig. 1). If user \( k \) were to be served when its rate \( R_k \) is above the threshold \( \hat{R}_k \), then it would occupy \( 1/K \) of the total slots and its throughput would be

\[ \hat{T}_k = \int_{\hat{R}_k}^{\infty} r f_{R_k}(r)dr. \tag{6} \]

In reality, however, \( \Pr(R_k \leq \hat{R}_k, k = 1, 2, \ldots, K) > 0 \), so the user throughput cannot attain \( \hat{T}_k \) for any scheduling algorithms. Therefore \( \hat{T}_k \) is just an upper bound of the user throughput.

The MR algorithm does not consider the probability distribution of user rates but blindly selects a user whose rate is the highest. If we take advantage of user rates, however, the selection can be made more intelligent such that a user at its best condition can be selected. In more rigorous term, we define a cdf-based scheduling (CS) algorithm as follows: If \( R_k(n) \) takes on a value \( r_k \) at slot \( n \), \( k = 1, 2, \ldots, K \), we select the user \( k^*(n) \) for which \( \Pr(R_k > r_k) \) is the smallest among all users, that is, the user whose rate \( r_k \) is high enough and least probable to become higher. Noting that \( \Pr(R_k > r_k) = 1 - F_{R_k}(r_k) \), we may express a CS algorithm by

\[ k^*(n) = \arg \max_k F_{R_k}(R_k(n)). \tag{7} \]

If the rates \( R_k \)'s are i.i.d., then the CS algorithm will be equivalent to the MR algorithm.

\textbf{Proposition 1:} The probability that the CS algorithm selects user \( k \) given that \( R_k(n) = r_k \) is \(^2\)

\[ \Pr(k^*(n) = k | R_k(n) = r_k) = F_{R_k}(r_k)^{K-1}. \tag{8} \]

According to this proposition, the probability that user \( k \) is selected for transmission, given that \( R_k(n) = r_k \), would be the same as the case when \( R_k \)'s are i.i.d. and the employed scheduler is the MR algorithm. In addition, this probability depends on the user’s own distribution and the number of users. It is independent of the probability distributions of other users! Recalling that this selection probability varied depending on the distribution of other users in the cases of all other scheduling algorithms, we may identify this property as a distinctive feature of the CS algorithm.

\textbf{Proposition 2:} The CS algorithm achieves an equal-time sharing among users.

\textbf{Proposition 3:} For the CS algorithm the average throughput of a user is independent of the probability distributions of other users.

According to above propositions, in the case of the CS algorithm, the average throughput of a user remains fixed if the rate distribution of the user is fixed, whether or not the rates of some other users are relatively better distributed to yield a higher throughput. This contrasts to the case of the algorithm by Liu et al. \([5]\), in which the average throughput of a user becomes lower if the distribution of other users becomes relatively better, and vice versa.

\(^1\) This assumption is not unrealistic as in case the pdf is not known we can collect the feedback rate information to make a histogram and regard it as the pdf. In addition, in case the rate is known to be Gaussian distributed but its mean and variance are unknown, we can estimate them based on the feedback rate information.

\(^2\) Refer to \([10]\) for the proofs of all the propositions in this paper.
IV. CS Algorithm for Gaussian Distributed Rates

Now we investigate the user throughput of the CS algorithm for the case when the rate is Gaussian distributed. When the number of transmit or receive antennas (or both) is large, the distribution of capacity approaches Gaussian distribution according to the central limit theorem [9]. So, we may assume that the rate of user \( k \) is a Gaussian random variable.

Let \( R_k(n) \) be a Gaussian random variable with mean \( m_k \) and variance \( \sigma_k^2 \), \( k = 1, 2, \ldots, K \). Then we have \( F_{R_k}(r) = 1 - Q \left( \frac{r - m_k}{\sigma_k} \right) \), where \( Q(x) = \frac{1}{\sqrt{2\pi}} \int_{x}^{\infty} e^{-\frac{t^2}{2}} dt \). We may represent the CS algorithm in an equivalent simple form

\[
k^*(n) = \arg \max_k \frac{R_k(n) - m_k}{\sigma_k}.
\]

We can evaluate the average user throughput of the CS algorithm as follows:

**Proposition 4:** For the CS algorithm with Gaussian distributed rates, the average throughput of user \( k \) is

\[
T_k = \frac{m_k}{K} + \frac{\sigma_k}{K} G_K
\]

for

\[
G_K = K \int_{0}^{1} u^{K-1} Q^{-1}(1-u) du.
\]

Note that, for the case of the CS algorithm, the average throughput of a user for which the rate is Gaussian distributed is given by (10) whether or not the other users' distribution is Gaussian because the average user throughput does not depend on the distribution of other users. If we adopt a round robin (RR) scheduling algorithm, the user throughput becomes \( m_k/K \), so we may regard \( \sigma_k G_K / m_k \) as a scheduling gain and \( \sigma_k G_K / m_k \) as a normalized scheduling gain.

We consider the case where all \( K \) user rates are Gaussian distributed with mean \( m \) and variance \( \sigma^2 \). Then, the total throughput \( S_K \) of the CS algorithm becomes

\[
S_K = m + \sigma G_K.
\]

Note that this is the exact form of the average total throughput that Hochwald, Marzetta, and Tarokh derived in [9] in the form

\[
S_K^{HMT} = m + \sigma \sqrt{2 \ln K}
\]

using a lemma on the maximum of a sequence of independent Gaussian random variables. The difference lies in the terms \( G_K \) and \( \sqrt{2 \ln K} \). It turns out that \( \sqrt{2 \ln K} \) is larger than \( G_K \), which indicates that (13) overestimates the total throughput.

**Numerical Example:**

We compare the scheduling performances by simulation.

We consider the case of five users (i.e., \( K = 5 \)) and assume that the user rates are Gaussian distributed with mean \( m = 10 \) and standard deviations \( \sigma_k = 0.2k + 1, k = 1, 2, \ldots, 5 \). Fig. 2 depicts the time fraction that each user occupies for data transmission. We observe that the MR algorithm is not fair in terms of the transmission time, with the larger-variance user occupying more time. In contrast, each user gets almost equal sharing of transmission time in the cases of the Liu's algorithm and the CS algorithm.

We further compare the performances of the Liu's algorithm and the CS algorithm in the aspect of the effects of other users' distribution on the average throughput. We consider the same environment as above except that \( \sigma_5 \) varies from 1 to 2. Fig. 3 depicts the resulting average user throughput for the Liu's algorithm and the CS algorithm.
algorithm. We observe that the Liu’s algorithm yields the average user throughputs that vary depending on the variation of $\sigma_5$. The average throughput of some user decreases as $\sigma_5$ increases. Moreover, the ratio of the scheduling gain and the standard deviation is not kept constant. In contrast, the CS algorithm maintains the same average user throughputs of users 1–4 for the both cases $\sigma_5 = 1$ and $\sigma_5 = 2$. In addition, we observe that the simulated user throughput is close to the analysis ($\sigma$ = 2.0) and in reality, however, independence among the users. In providing data services average throughput while achieving fairness and investigated its properties by setting it a goal to maximize the Liu’s algorithm in terms of scheduling gains.

V. STARVING-TIME LIMITED CS ALGORITHM

So far, we have introduced the CS algorithm and investigated its properties by setting it a goal to maximize the long-term average throughput while achieving fairness and independence among the users. In providing data services in reality, however, short-term performance is equivalently important as a user who gets a long-time starvation before data transmission would not be equally satisfied. In this context we consider a starving-time limited version of the CS algorithms in the following:

A. Starving Time

We denote by $N_k(n)$ the cumulative number of selections that user $k$ has got until the time slot $n$, i.e.,

$$N_k(n) \equiv \sum_{i=1}^{n} 1_{k^*(i)=k},$$

and denote by $Q_k(m)$, $m = 1, 2, \cdots$, the time (in time slots) when user $k$ got its $m$th selection for transmission, i.e.,

$$Q_k(m) \equiv \min\{n | N_k(n) = m\}.$$  

We define the starving time of user $k$ at time slot $n$ to be the time (in time slots) lapsed since the last selection for transmission, i.e.,

$$t_k(n) \equiv n - Q_k(N_k(n)),$$  

and the $i$th inter-service time of user $k$ to be the time difference (in time slots) between the $(i-1)$th and $i$th selections for transmission, i.e.,

$$t_k^i \equiv Q_k(i) - Q_k(i-1).$$

We modify the CS algorithm as follows to incorporate that the maximum inter-servie time of all users be limited to $t_{\text{max}}$:  

1. If there exists a user $k$ such that $t_k(n) = t_{\text{max}} - 1$, then the scheduler selects user $k$ and sets $0 \rightarrow t_k(n+1)$ and $t_i(n+1) + 1 \rightarrow t_i(n+1)$ for $i \neq k$.
2. Otherwise, the scheduler selects a user $j$ according to the original CS scheduling algorithm and sets $0 \rightarrow t_j(n+1)$ and $t_i(n) + 1 \rightarrow t_i(n+1)$ for $i \neq j$.

In order for this starving-time limited CS (CS-STL) algorithm to work properly, $t_{\text{max}}$ (in time slots) should be greater than or equal to the number of users, $K$. When $t_{\text{max}} = K$, in particular, the scheduler reduces to an RR scheduler, whose average inter-service time is $K$.

The original CS algorithm without starving-time limitation corresponds to the special case of the CS-STL algorithm with $t_{\text{max}} = \infty$. In this case, if the selection probability of user $k$ is $p_k$, $k = 1, 2, \cdots, K$, which is assumed to be independent along the time slots, then the inter-service time is binomially distributed and the average inter-service time becomes

$$E[t_k] = \sum_{t=1}^{\infty} t(1-p_k)^{t-1} p_k = \frac{1}{p_k}.$$  

Thus, for the case of the CS algorithm, the average inter-service time is $K$ since $p_k = 1/K$ for all $k$.

The CS-STL algorithm does not discriminate one user from the others, which implies $\Pr(t_k = 0) = 1/K$ for $k = 1, 2, \cdots, K$. Therefore, the time fraction occupied by user $k$ converges to $1/K$ a.s. [8]. So, in case the value of $t_{\text{max}}$ lies in between (i.e., $K < t_{\text{max}} < \infty$), the time-averaged inter-service time converges to $K$ a.s. as the following Proposition dictates:

Proposition 5: If the time fraction scheduled for user $k$ converges to $p_k$ a.s., i.e.,

$$N_k(n)/n \rightarrow p_k \quad \text{a.s.}$$

as $n \rightarrow \infty$, then the time-averaged inter-service time converges to $1/p_k$ a.s.

As the time fraction scheduled for each user converges to $1/K$ a.s., the time-averaged inter-service time converges to $K$ a.s., which is the same value as for the extremal cases $t_{\text{max}} = K$ and $t_{\text{max}} = \infty$. Therefore it turns out that the CS-STL algorithm limits the maximum starving time without changing the average inter-service time.

B. Throughput Analysis

We examine the effects of limiting the starving time on the average throughput of the CS-STL algorithm. In the case of the original CS algorithm with $K$ users, if we denote by $T_k^{(K)}$ the average throughput of user $k$, then

$$T_k^{(K)} = \int r F_{R_k}(r)^{K-1} f_{R_k}(r) dr.$$  

This starving-time limitation mechanism may be applied to other existing scheduling algorithms as well.
We consider the case of the CS-STL algorithm with a maximum inter-service time $t_{\text{max}}$. If we collect the starving time of every user at the current time slot to define a state $(t_1, t_2, \ldots, t_K)$, then it forms a discrete Markov chain. The event that the starving time of user $k$ is $t_{\text{max}} - 1$ can be regarded as a “deadline” of user $k$ because the scheduler has no choice but to select user $k$. Then, as the CS-STL algorithm does not discriminate one user from the others, the “deadline probability,” $\Pr(t_k = t_{\text{max}} - 1)$, is the same for all users and is denoted by $P_{\text{max}}$. If there exists a user $k$ such that $t_k = t_{\text{max}} - 1$, then the scheduler should select user $k$ as if it were the single user in the system, so the average user throughput becomes $\bar{T}_k^{(1)}$. Otherwise, the scheduler selects user $k$ with probability $1/K$ according to (7) and the average throughput of user $k$ is $\bar{T}_k^{(K)}$. Therefore, the user throughput of the CS-STL algorithm becomes

$$\bar{T}_k = P_{\text{max}}\bar{T}_k^{(1)} + (1 - KP_{\text{max}})\bar{T}_k^{(K)} = \bar{T}_k^{(K)} - \left[K\bar{T}_k^{(K)} - \bar{T}_k^{(1)}\right]P_{\text{max}},$$

where $\bar{T}_k^{(1)}$ is the average inter-service time, which (in time slots) be-

As $t_{\text{max}}$ grows to infinity, $P_{\text{max}}$ approaches 0. So, $\bar{T}_k$ converges to $\bar{T}_k^{(K)}$, which is the average user throughput of the original CS algorithm. Therefore, we may regard $\left[K\bar{T}_k^{(K)} - \bar{T}_k^{(1)}\right]P_{\text{max}}$ as the throughput loss due to the limited starving time, which can be made arbitrarily small by appropriately choosing the value of $t_{\text{max}}$.

Fig. 4 plots the total throughput, $\sum_k \bar{T}_k$, of the CS-STL algorithm for the Gaussian distributed user rates with $m = 10$ and $\sigma = 2$. We observe (and can confirm by evaluation) that the throughput loss drops below 1% if $t_{\text{max}}$ is set to be larger than 5, 8, and 12, respectively for the cases $K = 2, 3, and 4$.

VI. Concluding Remarks

So far we have introduced a new wireless packet scheduling algorithm that considers the throughput and fairness at the same time. The CS algorithm distinguishes itself from other existing algorithms in that the probability of selecting a user depends only on the rate distribution of that particular user without regard to the probability distributions of others. This decoupling effect enables an exact estimation of throughput, which is a very desirable property that helps to envision the relation between the scheduler and other related protocols.

In wireless communications with varying channel conditions, it can happen that a user does not get served for a long time while its channel condition remains relatively worse. The CS-STL algorithm alleviates this problem in a mild way by limiting the maximum inter-service time to a pre-specified value $t_{\text{max}}$. As proved in Proposition 5, it turned out that the CS-STL algorithm does not change the average inter-service time, which (in time slots) be-

**Fig. 4. Total throughput of the CS-STL algorithm when $m = 10$ and $\sigma = 2$. (The dashed lines are for the corresponding CS counterparts.)**

**REFERENCES**


