Reducing Feedback for Opportunistic Scheduling in Wireless Systems

Shailesh Patil and Gustavo de Veciana

Wireless Networking and Communications Group

Dept. of Electrical and Computer Engineering

The University of Texas at Austin

{patil,gustavo}@ece.utexas.edu

Abstract

We study reducing the feedback overheads for users' channel state information required for opportunistic scheduling at a base station. We first consider only best effort traffic, here we propose a contention based scheme known as 'static splitting' to reduce the amount of feedback needed. The idea is to divide users into static groups, with users that belong to a group and have their current channel quality above a threshold contending to send their current feedback. We combine static splitting with maximum quantile scheduling – scheduling a user whose current rate is high relative to its distribution, to obtain thresholds that are independent of users' channel capacity distributions. Our simulation results show that the proposed scheme can do better than other schemes. Next we consider supporting a mixture of best effort and real-time traffic. Here we combine combine contention based approach with polling subsets of users to propose a joint polling and opportunistic scheduling (JPOS) scheme that reduces the amount of feedback, while meeting real-time users' quality of service guarantees. Under fast fading, we prove a lower bound on the service seen by a real-time user under JPOS and propose a heuristic that exploits opportunism across all users.

Index Terms – Feedback, opportunistic scheduling, protocols, quality of service.

Part of this work was presented at Allerton 2005 as an *invited* paper.

I. INTRODUCTION

Motivation. Scheduling of users' downlink data transmissions at a base station has attracted a substantial amount of attention, see e.g., [6][7]. A key feature of wireless systems relative to the traditional wireline systems is that the channel capacity, or service rate, may exhibit temporal variations. This allows one to consider scheduling policies that choose to send to, or receive from, a user (or a subset of users) which at a given point in time has (have) the 'best', e.g., highest, capacity. Such opportunistic scheduling can lead to substantial increases in the aggregate capacity of a wireless system, and has thus been adopted in various wireless standards, such as CDMA-HDR [1].

Whenever a base station makes an opportunistic decision on the user(s) to serve, it needs to know the 'current' channel capacity (or some function of it) for all of the users. Therefore before *each* decision, *all* users need to transmit their current channel conditions to the base station. This can be a high overhead in terms of the bandwidth and the energy expended (especially at the mobile) for feedback, as compared to the gains in throughput that one might hope to glean from opportunistic scheduling.

For example, consider a 100 user system where *all* users are experiencing independent and identically distributed (i.i.d.) Rayleigh channel fading signal to noise ratio (SNR) and all feedback their channel state prior to each data transmission slot. Here one can achieve 90% of the gain in average SNR of the user served by soliciting feedback from a random subset of only 60 users and serving the user with the highest current SNR among them. This underscores the need and the potential to reduce feedback required to realize the major gains of opportunistic scheduling.

One can reduce the resources used for feedback in the following two simple ways.

Contention. In the contention based approach users compete for a pool of resources allocated, e.g., CDMA code, a time slot, etc., to feed back their current channel capacity state. For example, a user may opportunistically send its feedback if its current channel capacity is above a certain threshold. The thresholds are designed so that feedback is successful, i.e., only one or a limited number of users contend at the same time. This allows one to achieve a large part of the opportunistic gains possible over long time periods. However, it is possible that feedback gets wasted because too many users contend to transmit their states. Therefore occasionally no opportunism is exploited.

Polling. Alternatively the base station may solicit feedback from a subset of users, i.e., allocates feedback resources for the subset of users, and choose to opportunistically serve among them. Since we are exploiting opportunism over only a subset of users the long term gains of this approach are generally lower than the contention based approach. However, since there is no wasting of feedback resources, some degree of opportunism is almost always exploited.

Related Work. Let us discuss some of the previous work done in this area. A simple threshold based scheme was proposed in [4] to reduce the feedback overhead. Their each user had a dedicated resource for sending its feedback. The idea was to only allow a user whose current channel capacity exceeds a threshold to feedback his current state. Their scheme reduced the amount of feedback required significantly, while achieving most of the gains of opportunism. However, even though the energy spent in transmitting and receiving the feedback is reduced, the need for other resource requirements (like bandwidth, etc.) for sending feedback is not reduced.

Some contention based schemes have also been proposed in the literature. One of the frequently cited ideas is ‘opportunistic splitting’ proposed in [11]. The scheme was proposed in an uplink context, but it is applicable to downlink scheduling (which is the focus of this paper). The idea is to divide time into equal sized time units, each unit consists of mini slots which are pooled resources used to learn the current channel capacity of users via feedback, while the rest of the unit is used for data transmission.

In opportunistic splitting, initially a pair of thresholds depending on the number of users is set. At the start of the first mini slot, every user whose current channel capacity is between the pair of thresholds contends, i.e., transmits to the base station. The base station then broadcasts to all the users whether on the mini slot no user contended, exactly one user contended, or a collision occurred, i.e., more than one user contended and the base station was unable to decode any information. Depending on the broadcast message received, each user modifies its threshold according to a binary search like algorithm and users’ whose channel capacity is between the new thresholds contend in the next mini slot. This process continues until exactly one user contends, this user is guaranteed to be the user with currently the highest channel capacity. The authors show that on average 2.5 mini slots will be required for the algorithm to find the user with the highest current channel capacity.

However, opportunistic splitting requires two way communication and coordination between the base station and users every mini slot, and a variable number of mini slots can be used in a time unit. This may be hard to implement since the time scales involved are quite small, usually less than milliseconds in practical systems. To overcome this problem, a random access based feedback protocol was proposed in [15], where only one way communication is required, and the number of mini slots were fixed. In each mini slot, users whose current channel capacity exceeds a threshold contend with some probability. If on a given mini slot exactly one user contends, then that user's identity is stored at the base station. Subsequently the base station randomly serves one of the identified users. The threshold and probability of contention can be optimized to maximize the overall sum capacity if the channel distributions are known. However, simulation results presented in the paper show that a truncated and thus comparable version of opportunistic splitting usually performed better than the proposed scheme. (Some researchers have also studied reducing feedback in OFDM systems [13][14], which is not the focus of this paper.)

An underlying assumption in the above research was that users in the system see i.i.d. channel capacity distributions. (Note that an extension of opportunistic splitting to the case where users can experience one of two possible channel capacity distributions is presented in [12], however this still does not seem a reasonable model.) In practice users' channel capacity variations are heterogenous, e.g., users close to a base station see significantly different channel capacity than those further off. Extending these schemes to the non i.i.d. case is in general very complex, because the thresholds then will not only be dependent on the user's own channel capacity distribution and the number of users, but also other users' channel capacity distributions. Ideally one would like to have an easy way to set the thresholds for the heterogeneous case.

All of the above mentioned work focussed on reducing feedback in the context of opportunistic scheduling of best effort traffic. Whereas base stations are likely to support a mixture of both best effort and real-time traffic. Real-time traffic has requirements over short time scales, therefore for real-time traffic to benefit from opportunism one needs to exploit opportunism over short time scales. Additionally, as the deadline for meeting users' quality of service (QoS) requirement approaches, the base station needs to serve only those users whose deadline is nearing. Therefore the base station can exploit opportunism among only the subset of users that are roughly in equal danger of not meeting their QoS requirements. Summarizing, the need for efficient utilization of feedback resources is higher here.

Contributions. We first propose a contention based scheme, which we shall call *static splitting*, also geared at reducing feedback overheads in a best effort traffic only scenario. Like [15], our setup consists of a fixed number of mini slots, and does not require two way communication. We will combine static splitting with a distribution based scheduler that we call *maximum quantile scheduling* [7][2][12] to handle heterogeneity in users' channel capacity variations. The idea there is to schedule the user whose current rate is highest relative to his *own* distribution, i.e., in the highest quantile. Under maximum quantile scheduling one can compute a common threshold determining when users are to transmit feedback, which is *independent of their possibly heterogenous channel distributions*. This is unlike other proposed schemes as it allows off-line calculation of 'optimal' thresholds.Our simulation results indicate that static splitting can perform much better (for example 40% improvement) than a truncated form of opportunistic splitting.

We then consider a scenario where traffic is a mixture of best effort and real-time traffic, for which QoS guarantees have to be met over short time scales. We argue that in such a scenario a combination of contention and polling based feedback strategies is needed. Based on this insight, we combine static splitting and dynamic polling with a variation of a token based scheduling scheme proposed in [9] to provide QoS. We call this the *joint polling and opportunistic scheduling* (JPOS) scheme. Under fast fading, we show a lower bound on the service seen by a real-time user under the JPOS scheme. Furthermore, based on this scheme we propose a heuristic that simulations indicate not only meets users' QoS guarantees, but achieves up to 89% of system capacity in terms of the long term throughput that is realized.

Paper Organization. The paper is organized as follows. In Section II we give a brief introduction to maximum quantile scheduling and describe the proposed static splitting scheme in the best effort traffic only scenario. In Section III we consider the case where traffic is a mixture of both best effort and real-time flows and describe the proposed JPOS scheme to reduce feedback in such mixed scenario. Simulation results are presented in Section IV, and Section V concludes the paper.

II. Opportunistic Feedback Based on Static Splitting

A. System Model and Notation

We begin by introducing our system model and some notation. For simplicity, we focus on downlink scheduling from a base station to multiple users. Suppose time is divided equal size 'time units'. Each time unit consists of k equal size mini slots followed by a transmission slot during which at most one user can be served (see Figure 1). The k mini slots are used for collecting feedback. We will define the exact nature of the feedback later. In the sequel we use the terms 'channel capacity' and 'rate' interchangeably and make the following assumption on user's channel capacity characteristics over time units.

Assumption 2.1: We assume the channel capacity for each user is a stationary ergodic process and these processes are independent, but not necessarily identically distributed across users. Further we assume that the marginal distribution for each user is continuous and is known a priori at the user.

Some comments on this assumption. First the channel capacities seen by users might indeed be roughly stationary over a reasonable period of time particularly if users are at fixed locations. The assumption that users' rates are independent is also likely to be true, though a notable exception is the case where mobile users move in a correlated manner, e.g., along a highway. The assumption that a user has a priori knowledge of the marginal distributions for the channel capacity may not be completely reasonable. Yet, simple book keeping of the currently achievable rate can be used to obtain estimates for the marginal distributions. (This assumption is implicitly used in all of the previous work discussed above [15][4][12][11]). We require the users' rates to be continuous only to keep our discussion simple. (In fact we perform our simulations for discrete rate distributions.) For details on handling the discrete case we refer the reader to Chapter 2 of [8]. Note that channel capacities are not restricted to having a specific distribution, or even to a class of distributions, i.e., users can undergo arbitrary fading processes. This makes the analysis presented in the sequel applicable to real world scenarios.

We will let $x^i(t)$ denote the realization of the downlink channel capacity/rate of user i at time unit t, and let X^i be a random variable whose distribution is that of the channel capacity of user i on a *typical* time unit. Recall that we assume X^i to be continuous random variables that are independent but need not necessarily be identically distributed across users. We denote the distribution function of X^i by $F_{X^i}(\cdot)$. For simplicity, we will assume that $F_{X^i}(\cdot)$ is a strictly increasing function, so that its inverse denoted by $F_{X^i}^{-1}(\cdot)$ is defined. Finally note that by Assumption 2.1 $F_{X^i}(\cdot)$ is known at the user.

For analysis purposes, we will only consider a 'fixed saturated' regime where there is a fixed number of n users in the system and each user is infinitely backlogged. For now we allow only best effort flows.

B. *Maximum Quantile Scheduling*

Let us briefly discuss maximum quantile scheduling now. Maximum quantile scheduling was independently proposed by several researchers under different names in . Specifically [7] proposed a 'CDF based scheme', [2] proposed the so called 'score based scheduler' and [12] proposed the notion of a 'distribution fairness' based scheduler. In our own work we have studied properties of such an opportunistic scheduling scheme: in terms of enabling quality of service guarantees for real-time traffic in [9]; and, evaluating its performance in a measurement based set up in [10].

The main idea is to schedule a user who's rate is highest relative to its *own* distribution, i.e., serve user $i^*(t)$ during time unit t if

$$i^*(t) \in \arg\max_{i=1,\dots,n} F_{X^i}(x^i(t)).$$

It is well known that $F_{X^i}(X^i)$ is uniformly distributed on $[0,1]$. Note that $F_{X^i}(X^i)$ is the quantile of the rate of user i on a typical time unit, then we see that maximum quantile scheduling can be viewed as picking the maximum among i.i.d. uniform random variables corresponding to the quantile for the current rate of each user. Then maximum quantile scheduling is equally likely to serve any user on a typical slot, and as a result all users get served an equal fraction of time, i.e., $\frac{1}{n}^{th}$ of time.

Let $X^{i,(n)}$ denote the maximum of n i.i.d. copies of X^i, i.e., $X^{i,(n)} := \max[X_1^i, \dots, X_n^i],$ where $X_j^i \sim X^i,\ \forall j = 1, \dots, n$. Then, the average throughput seen by user i under maximum quantile is given by

$$G_{mq}^i(n) = \frac{E[X^{i,(n)}]}{n}. \tag{1}$$

Maximum quantile scheduling has very desirable properties, it maximizes the amount of opportunism, i.e., quantile of the user being served, is intrinsically temporally fair, and asymptotically in the number of users, maximizes the sum throughput [10]. As compared to other opportunistic scheduling schemes proposed in literature, e.g., maximum throughput [6], proportionally fair [1], etc., maximum quantile has some distinct advantages in handling cases where users have heterogeneous rate channel capacity distributions and one must resort to estimating parameters [10]. In this paper, we will focus on channel feedback schemes which are compatible with opportunistic scheduling of users currently experiencing channel capacity in the high quantile, i.e., high $F_{X^i}(x^i(t))$.

C. Proposed Static Splitting Feedback Scheme for Best Effort Traffic

Recall that in each time unit the data transmission part is preceded by k mini slots. The objective during the mini slots is to identify a user whose current rate is in a high quantile, and not necessarily identify the user with the highest quantile (we will revisit this point later). For simplicity we will assume in this subsection that the number of users n is such that $\frac{n}{k}$ is an integral value. In the proposed scheme users are split into k equal sized ‘static’ groups, and each group is associated with a mini slot. A user can only contend (i.e., send feedback) on the mini slot with which its group is associated. Based on n and k (which a user learns from the base station), each user calculates (looks up) a common quantile threshold denoted by q, which is used to determine if it will contend by transmitting feedback – we will give details on optimizing q later. Specifically, recall that at time unit t, the rate user i can support is denoted by $x^i(t)$. Prior to user i’s mini slot the user would check if $x^i(t) > F^{-1}_{X^i}(q)$, and if so it would transmit the quantile $F_{X^i}(x^i(t))$ of its current rate to the base station.

If only one user contends during a given mini slot, we assume that the base station is able to both decipher and store the identity and the current value of the quantile of the user. If more than one user contends for a given mini slot, a collision occurs, and the base station stores this fact. Finally, if no user contends for the mini slot, then no action is taken. The process is repeated across all mini slots.

Once the contention for mini slots is finished, if the base station was able to identify at least one user, then it serves the user with the highest quantile among the identified users. Else, if the base station fails to identify any such user, then it serves a randomly selected user. This can occur in two ways, if collisions have been recorded for none of the mini slots, then a user is randomly selected from all the users. However, if a collision occurred on at least one of the mini slots, then a user is randomly selected for service among the groups of users associated with the mini slots where collisions occurred. Doing so, increases the chance of choosing a user with a high quantile.

The last challenge for this simple protocol is determining a good choice for the contention thresholds q. Let A^i denote the event that user i is selected under the above scheme and $\mathbf{1}_{A^i}$ be the indicator function for A^i. Our goal is to serve users so as to maximize the expected *sum quantile* $E[\sum_{i=1}^n F_{X^i}(X^i)\mathbf{1}_{A^i}]$ of the scheduled users. Since each mini slot has exactly the same number of users associated with it, $F_{X^i}(X^i)$ are i.i.d. across users, and since all users share a common threshold q, it follows by symmetry that each user is equally likely to be selected, i.e., $\Pr(A^i) = \frac{1}{n}$, and so

$$E[\sum_{i=1}^{n} F_{X^i}(X^i)\mathbf{1}_{A^i}] = \sum_{i=1}^{n} E[F_{X^i}(X^i)|A^i]\Pr(A^i) = \frac{1}{n}\sum_{i=1}^{n} E[F_{X^i}(X^i)|A^i].$$

Again by symmetry, it suffices to optimize q to maximize $E[F_{X^i}(X^i)|A^i]$ for any user.

Let A_b^i, $b = 1, \ldots, k$ denote the event that successful contention occurs over b mini slots, and user i is selected for service (because it had the highest quantile among the b identified users). We shall let A_0^i denote the event that user i is selected at random in the case where the feedback scheme was not able to identify any user. Note that A_b^i, $b = 1, \ldots, k$ and A_0^i form a partition of A^i and so we have that

$$E[F_{X^i}(X^i)|A^i] = \sum_{b=1}^{k} E[F_{X^i}(X^i)|A_b^i]\Pr(A_b^i|A^i) + E[F_{X^i}(X^i)|A_0^i]\Pr(A_0^i|A^i). \tag{2}$$

Let p_n denote the probability that a user is able to successfully contend in a mini slot in a time unit with n competing users, then $p_n = \frac{n}{k}(1-q)q^{\frac{n}{k}-1}$. Now consider $\sum_{i=1}^n \Pr(A_b^i)$, it is the total probability of selecting a user that has the highest quantile among the exactly b identified users. This is equal to the probability that the base station identifies exactly b users, i.e., successful contention on b mini slots, then,

$$\sum_{i=1}^{n} \Pr(A_b^i) = \binom{k}{b}(p_n)^b(1-p_n)^{k-b}.$$

Now by symmetry, for any user i, $\Pr(A_b^i) = \frac{1}{n}\binom{k}{b}(p_n)^b(1-p_n)^{k-b}$. Furthermore since $\Pr(A^i) = \frac{1}{n}$,

$$\Pr(A_b^i|A^i) = \binom{k}{b}p_n^b(1-p_n)^{k-b}.$$

One can also easily show that

$$E[F_{X^i}(X^i)|A_b^i] = \frac{b}{b+1}(1-q)+q.$$

Finally, we have that

$$\Pr(A_0^i|A^i) = 1 - \sum_{b=1}^{k} \Pr(A_b^i|A^i),$$

and by ignoring the conditioning we can approximate $E[F_{X^i}(X^i)|A_0^i]$ (i.e., the average quantile of the selected user when it is selected at random) as

$$E[F_{X^i}(X^i)|A_0^i] \approx \frac{1}{2}.$$

Putting these results together we can rewrite (2) as

$$E[F_{X^i}(X^i)|A^i] \approx \sum_{b=1}^{k} \binom{k}{b} p_n^b (1-p_n)^{k-b} (\frac{b}{b+1}(1-q)+q) + \frac{1}{2}(1-p_n)^k. \tag{3}$$

A good threshold q should maximize the above approximation. This can be done numerically by searching over $q \in [0,1]$. Table I lists the optimum threshold q for an increasing number of users for $k = 2, \ldots, 9$. The threshold increases with n for a given k, and with k for a given $\frac{n}{k}$, as might be expected.

Note that (3) is independent of users' channel capacity distributions and can also be used to find an approximate value of q even when $\frac{n}{k}$ is not an integral value. As mentioned earlier, unlike other schemes, this eliminates the need for online real-time calculations, it is sufficient to do off-line calculation of thresholds and store them in a table.

Some final comments, note that opportunistic splitting was designed to find the user with the highest quantile/rate. In a practical system where the number of mini slots may be limited, if the scheme is unable to find the user with the highest quantile in those many mini slots, a user has to be chosen at random. This is not desirable. Whereas if static splitting is unsuccessful in finding the user with the highest quantile, it is still likely to serve a user with high quantile. The possibility of serving a high quantile user is captured in (3), in fact the expression also captures the performance of the scheme even when a user is selected at random. Therefore by maximizing (3), one can obtain better performance as compared to *truncated* opportunistic splitting especially for small values of k. We will verify this using simulations in Section IV.

III. Reducing Feedback for Scheduling Schemes providing Quality of Service

As discussed in Section I, when attempting to ensure quality of service to flows one has to exploit opportunism over small time scales and exploit opportunism among only those users that are roughly in the same danger of not meeting their QoS requirements. Polling based feedback mechanisms meet these criteria. In particular one can exploit opportunism over short time scales, and by dynamically deciding from whom to solicit feedback, one can exploit opportunism only among a desired subset of users.

However as mentioned earlier, a polling based approach is not as efficient at exploiting opportunism as a contention based approach. Therefore it makes sense to use contention based static splitting for scheduling best effort users and polling for scheduling of real-time users. We propose a modified form of the scheduling scheme proposed in [9] that is compatible with such a feedback strategy.

First let us modify our system setup to include real-time traffic flows. In our new setup each user is either associated with a real-time or a best effort stream, but not both [1]. The total number of users is still denoted by n, while the number of real-time users will be denoted by n_r. For simplicity consider the case where both real-time and best effort users are infinitely backlogged.

The notion of QoS considered in this paper involves ensuring a user i sees a desired rate r^i over a frame of length τ with an outage probability of δ^i. More formally, we divide time into equal sized frames consisting of τ time units and our goal is to ensure that for each of these frames

$$\Pr(R^i(\tau) \geq r^i) \geq 1 - \delta^i,$$

where $R^i(\tau)$ is a random variable denoting the cumulative rate seen by user i during a frame.

To guarantee the required QoS, we will use a stochastic envelope based approach [3][5]. The idea is to stochastically lower bound the actual service $R^i(\tau)$ by a quantity $\underline{R^i}(\tau)$, i.e., $\forall r$, $\Pr(R^i(\tau) \geq r) \geq \Pr(\underline{R^i}(\tau) \geq r)$. This is usually denoted as follows

$$R^i(\tau) \geq^{st} \underline{R^i}(\tau).$$

[1]We note that this is done only for simplicity, our scheme can be easily modified to allow a user to get associated with multiple real-time and/or best effort streams.

Therefore if $\underline{R}^i(\tau)$ meets the QoS guarantee then so will $R^i(\tau)$. Ideally we also want $\underline{R}^i(\tau)$ to be analytically tractable so as to enable resource allocation strategies.

We make an additional assumption here on users' rate in this section. We assume that users' channel capacity is fast fading, i.e., for any user i the realizations of X^i's are independent across slots.

A. Joint Polling & Opportunistic Scheduling Scheme

As discussed above, our goal is to give users rate guarantees over a frame of τ time units. In the proposed scheme the idea is to opportunistically serve each real-time user exactly l times every frame. This can be thought of as allocating each real-time user l tokens at the start of each frame. Whenever a real-time user is served, its token count goes down by 1, and when it has been served l times it is no longer allowed to contend for service. Note that it is required that $n_r l \leq \tau$ so that it is feasible to allocate l tokens to each real-time user. A key task is to calculate the value of l so as to meet the users' QoS requirements, this will discussed later.

We describe the scheme with an example. Consider a frame size of $\tau = 10$ time units, where each time unit contains $k = 3$ mini slots. The system has 5 best effort users and 3 real-time users, each real-time user has $l = 2$ tokens. We illustrate a realization of the scheme for the example in Fig. 2.

In order to serve each real-time user l times, we divide each frame into two parts. The first part of the frame consists of $\tau - n_r l$ time units and only best effort users are served here. Every best effort user contends for service in every time unit of the first part via the static splitting with maximum quantile scheduling mechanism described in Section II. In our example the first part of the frame consists of 4 time units, where best effort users are served (Fig. 2).

During the second part of the frame consisting of $n_r l$ time units, only real-time users are served. Unlike the first part of the frame, here the feedback mechanism is based on polling. In a time unit the base station decides to solicit the quantile of the current rate from $k - 1$ real-time users that currently have *the highest remaining number of tokens*. Ties among real-time users with equal remaining tokens are broken randomly. If there are less than $k - 1$ real-time users that currently have a positive token count, then all users are solicited for feedback. The first mini slot of each time unit is used to broadcast which mini slot has been assigned to which real-time user for polling its feedback. The remaining $k-1$ mini slots are used to get feedback on users' quantile information. The base station serves the real-time user with the highest quantile among those from which feedback was solicited, and the token count for that real-time user goes down by 1. This process continues until the end of the frame. It should be clear that by the end of the frame all real-time users would have been served l times.

Returning to our example, let us describe scheduling in the second part of the frame (Fig. 2). The second part starts with the 5^{th} time unit. During the 5^{th} time unit, all real-time users have $l = 2$ tokens each. Using random tie breaking, the base station chooses to solicit feedback from real-time users 1 and 3. Since real-time user 3 currently has the higher quantile, it gets served and its token count goes down. In the 6^{th} time unit because real-time users 1 and 2 have a higher remaining token count of 2, feedback is solicited from them, and real-time user 1 (on account of its higher current quantile) gets served. In the 7^{th} time unit using random tie breaking among real-time users 3 and 1, feedback is solicited from real-time users 2 and 1, and real-time user 1 gets served. Now since real-time user 1 has been served $l = 2$ times, it will no longer be served. In the next time unit real-time users 2 and 3 get polled, and real-time user 2 is served. In the 9^{th} time unit real-time user 3 gets served and is no longer considered for service. Finally in the 10^{th} time unit real-time user 2 is polled and served. Note that this is only one of the realization of the scheme the scheme could have proceeded in multiple ways.

Note that by choosing to poll users that have the highest remaining number of tokens we are not only polling users that have the highest danger of not meeting their QoS requirements, but also are keeping as many real-time users in the system as possible. This allows one to exploit a larger amount of opportunism as compared to the case where some users leave early.

The above scheme ensures QoS by serving each user exactly l times. However in doing so, it exploits opportunism available among the best effort users in the first part of the frame, and exploits opportunism among real-time users in the second part of the frame. In other words the scheme is able to exploit (only) intra class opportunism. Later we will propose a heuristic that will also exploit inter class opportunism.

B. Analysis and Resource Allocation

The number of ways in which the above described scheme can serve a real-time user grows exponentially with the number of tokens allocated, make it hard to analyze the service seen by a real-time user. Therefore we try to lower bound the service seen by a real-time user, which in turn will allow us to conservatively estimate the value of l.

The service seen by a real-time user under the JPOS scheme satisfies three properties which allows us to develop a lower bound. These were first described in [9], we reproduce them here.

Property 3.1: (*Equal Resource Allocation.*) All real-time users are allocated an equal number l of tokens.

Property 3.2: (*Symmetric Selection.*) In a typical time unit, each real-time user (with a positive token count) is equally likely to be selected for service.

Property 3.3: (*Monotonicity.*) Let $\tilde{X}^{i,(m)}$ be the random variable denoting the rate seen by user i given it is selected for service while competing with $m-1$ other users, then $\forall\ l$, $\tilde{X}^{i,(m+1)} \geq^{st} \tilde{X}^{i,(m)}$.

It is clear that since all real-time users are allocated l tokens each, Property 3.1 is satisfied. Property 3.2 is trivially satisfied in the first part of the frame. In the second part of the frame, all users start with an equal number of tokens. Furthermore in each time unit, among real-time users having an equal token count there is random tie breaking in deciding whom to poll, and the quantile of all real-time users are uniformly distributed. From these three symmetrical conditions one can show that Property 3.2 holds for the second part of the frame. For Property 3.3, consider a time unit in the second part of the frame with m real-time users are competing. If user i gets selected, then if $m \geq k-1$ it sees a service of $X^{i,(k-1)}$, else it sees a service of $X^{i,(m)}$ (see Subsection II-B). Clearly Property 3.3 is satisfied here.

We introduce some notation now. Let Z_j^{i*} be the random variable denoting the rate received by real-time user i conditioned on it getting selected for service the j^{th} time under the above described JPOS scheme. Then the total service seen by real-time user i under the JPOS scheme is given by $\sum_{j=1}^{l} Z_j^{i*}$. Now define a mixture random variable Y^i

$$Y^i = \begin{cases} X^{i,(k-1)} \text{ w.p. } 1 - \frac{k-2}{n_r} \\ X^{i,(k-2)} \text{ w.p. } \frac{1}{n_r} \\ \quad \ldots \quad \text{w.p.} \quad \ldots \\ X^{i,(1)} \text{ w.p. } \frac{1}{n_r}. \end{cases} \tag{4}$$

One can think of Y^i as user i getting selected for service in the second part of the frame when competing with greater than or equal to $k-1$ real-time users with probability $1-\frac{k-2}{n_r}$, or getting selected for service when competing with $k-2$ users with probability $\frac{1}{n_r}$ and so on. Note that Y^i only depends on X^i and n_r, but does not depend on the channel rate distribution of other users.

We now show that $\sum_{j=1}^{l} Z_j^{i*} \geq^{st} \sum_{j=1}^{l} Y_j^i$, where Y_j^i's are i.i.d. and $\forall j = 1, \ldots, l$, $Y_j^i \sim Y^i$. In other words, the service seen by user i under the JPOS scheme can be lower bounded by a sum of *i.i.d.* random variables that depends only on the number of real-time users and X^i, and yet factors in the opportunism that can be exploited. In fact, we show a stronger bound, i.e., for any set $S \subseteq \{1, \ldots, l\}$, $\sum_{j \in S} Z_j^{i*} \geq^{st} \sum_{j \in S} Y_j^i$. The following theorem formally states our claim. The proof of the theorem follows from that of Theorem 3.3 in [9], and is omitted here.

Theorem 3.1: Consider the JPOS scheme where all n_r real-time are allocated an equal number l of tokens. Then under Assumption 2.1 and fast fading on users' channel capacities, for any real-time user i

$$\sum_{j \in S} Z_j^{i*} \geq^{st} \sum_{j \in S} Y_j^i,$$

for $S \subseteq \{1, \ldots, l\}$. Here Y_j^i's are i.i.d. and $Y_j^i \sim Y^i$, $\forall j = 1, \ldots, l$.

Now if l' is large enough, then the distribution of $\sum_{j=1}^{l'} Y_j^i$ can be roughly approximated, e.g., using the Central Limit Theorem. Then since each real-time user knows its distribution and can learn the value of n_r from the base station, it can calculate its required number of tokens l^i as

$$l^i = \min_{l'} \{ l' \mid \Pr(\frac{\rho}{\tau} \sum_{j=1}^{l'} Y_j^i \geq r^i) \geq 1 - \delta^i \}, \tag{5}$$

here ρ is the fraction of time unit that is used for data transmission. The value of l^i can be communicated to the base station and the base station can allocate each real-time user l tokens, where

$$l = \max_{i=1,\ldots,n_r} l^i, \tag{6}$$

and allocate each real-time user l tokens. (Of course one requires that $n_r l \leq \tau$.)

Note that the overhead involved in transmitting the value of l^i from each real-time to the base station is needed only when the number of real-time users change or a real-time user's channel distribution changes so much that its token requirement changes. Also note that even if a real-time user requires fewer than l tokens, it is still allocated l tokens. This may seem conservative, however one can group users together to reduce the total number of tokens required (see [9]).

We performed a simple numerical experiment to evaluate the usefulness of the proposed bound. We considered a system containing 12 real-time users, with each users requiring a rate guarantee of 40 kbps over 1 sec with an outage probability of 1%. All users are experiencing i.i.d. Rayleigh fading with a mean SNR of 2, and CDMA-HDR based mapping was used to map from SNR to rates. Each time unit was of size 5 msec (so the frame size was 200 time units) and consisted of $k = 6$ mini slots. The fraction of each time unit used for data transmission was $\rho = 0.9$. Our bound suggested that 10 tokens were needed for each real-time user. To put this in contrast we calculated the number of tokens needed if no opportunism is exploited, i.e.,

$$l^i = \min_{l'} \{ l' \mid \Pr(\frac{\rho}{\tau} \sum_{j=1}^{l'} X_j^i \geq r^i) \geq 1 - \delta^i \},$$

and found the number of tokens needed to be 18, clearly illustrating the advantage of the bound.

C. Heuristic based Joint Polling & Opportunistic Scheduling Scheme

As discussed earlier the proposed JPOS scheme only exploits the intra class opportunism among the best effort and real-time users, now we propose a heuristic that exploits both inter and intra class opportunism.

Recall that our goal of polling real-time users was to ensure that users with roughly equal danger of not meeting their QoS requirements are able to send their feedback and opportunism was exploited among them. However, in our proposed scheme the danger of not meeting the QoS requirement only occurs if the total remaining number of token is equal to (or less) than remaining time units in the frame. Otherwise, there is leeway to exploit opportunism across all users, and this can be exploited by allowing real-time users to compete with best effort users during the first part of the frame. This is the basis for our heuristic.

Like the JPOS scheme, in the proposed modification each real-time user is allocated l tokens at the start of each frame. Whenever a real-time user is served, its token count goes down by 1 and when it has been served l times, it is no longer considered for service. However unlike the original scheme, we allow both the best effort and real-time users to compete using maximum quantile based static splitting during the first part of the frame. Furthermore the size of the first part is dynamic and it lasts until the *total number of remaining tokens in the system is equal to the number of remaining time units in the frame*. Then the second part of the frame starts, here like the JPOS scheme only real-time users are served using polling. The method of deciding the real-time users that are to be polled for their current quantile is the same as the original scheme, i.e., polling real-time users with the $k-1$ highest remaining tokens counts and using random tie breaking for users with equal remaining token count. As before, the real-time user with the highest quantile among those polled is served.

To illustrate the scheme more clearly, we use the example described in Subsection III-A to describe the original scheme. A realization of the scheme is illustrated in Figure 3. Since real-time users are allowed to contend in the first part of the frame, real-time users 3 and 2 get served in the 2^{nd} and 3^{rd} time units respectively. As a result, the second part of the frame starts at the 7^{th} time unit. Therefore, inter class opportunism is exploited in this period of time. The second phase proceeds as in the original JPOS scheme, and is shown in the figure.

Even though the proposed modification scheme does exploit both the inter and intra class opportunism it is difficult to bound the QoS seen by a real-time user. This is because it is not clear whether the Monotonicity property holds under the scheme. However, we conjecture that calculating the value of l according to (5) and (6) will allow us to meet the required QoS guarantees. This conjecture is supported by the simulation results presented in the next section.

IV. Simulation Results

Let us describe the general simulation setup. In order to compare the performance of our scheme to other schemes discussed above, we restricted all users to undergo i.i.d. Rayleigh fast fading with a mean SNR of 2. We used the CDMA-HDR [1] SNR to rate mapping.

A. Static Splitting Performance

We first simulated static splitting and a truncated form of opportunistic splitting for the best effort traffic only scenario. The number of users associated with a mini slot were increased from 1 to 7, while $k = 2, 4, 6, 8$ mini slots were used.

Note that a mini slot in opportunistic splitting consists of two transmissions, whereas a mini slot in our scheme consists of only one transmission. Therefore to be fair, we count each transmission for opportunistic splitting as a mini slot. At the end of the mini slots if the algorithm is unable to find the user with the highest quantile, then it selects a user at random, i.e., the algorithm is truncated.

We compared the throughput achieved by the schemes to that achieved by a virtual scheme that is able to select the user with the highest rate every time unit, i.e., the best that the schemes can hope to achieve. We plot our results as the relative percentage loss in throughput compared to that achieved by the virtual scheme in Figures 4, 5, 6 and 7. Note that as expected the relative penalty for both the schemes goes down with increasing value of k, while it increases with the number of users.

The results also illustrate the advantage of using static splitting, our scheme does better than opportunistic splitting for $k = 2, 4, 6$. The difference in performance can be significant, for example at $k = 4$ and $n = 12$ in Figure 5, the relative loss for goes down from 15.93% for opportunistic splitting to 9.63% for static splitting, i.e., a 40% reduction. Note that for $k = 6$ truncated opportunistic splitting has greater than the average of $2.5 * 2$ needed for the scheme to converge, yet static splitting does better.

At $k = 8$ in Figure 7, opportunistic splitting does better than static splitting. This is because as k increases, opportunistic splitting is increasingly able to find the user with the highest rate. However, we note that the engineering complexity needed for opportunistic splitting may not justify the gain it shows over static splitting.

B. Performance of Joint Polling & Opportunistic Scheduling Scheme

As a second experiment we simulated the proposed JPOS scheme and its heuristic modification. In our setup there were 12 best effort users and 12 real-time users. Each time unit consisted of $k = 6$ mini slots and was 5 msec long. The data transmission part of each time unit was 4.5 msec long (i.e., $\rho = 0.9$) with the rest being used to gather feedback. Each real-time user was given a guarantee of 40 kbps with an outage probability of 1% over frame size varying from 100 time units to 600 time units in steps of 100 time units (i.e., 500 msec to 3 sec in steps of 500 msec).

We kept track of the throughput achieved by the schemes and whether the real-time users were able to meet their guarantee. As expected the JPOS scheme was able to provide the required guarantee to all the real-time users. Additionally, the heuristic modification was also able to provide the required guarantee to all the real-time users in all the cases, supporting our conjecture that the heuristic modification will be able to meet real-time users' QoS requirements. We again compared the throughput achieved by the schemes to a virtual scheme that always serves the user with the highest current rate. The results as a percentage of throughput achieved by the virtual scheme are plotted in Fig 8. Note that the throughput for both schemes increases as the QoS guarantee is given over longer time frames. We also observe that the heuristic modification has a higher overall throughput, clearly illustrating the advantage of exploiting inter class opportunism. Furthermore, both schemes are able to achieve a fairly high fraction of the overall throughput possible, with the JPOS scheme achieving 79% and the heuristic modification achieving 89%.

V. Conclusion

In this paper we presented a simple scheme to reduce feedback overheads under opportunistic scheduling in wireless networks. The scheme is novel in the sense that one can theoretically compute the required contention thresholds independent of users' distributions, making it applicable to real world scenarios. The good performance of the proposed scheme is verified using simulations.

We also developed the insight that to reduce feedback in an opportunistic system where a mixture of real-time and best effort traffic is being served a combination of contention and polling based feedback approach is appropriate. We proposed two schemes based on this insight. Simulation results indicate that both schemes are able to exploit a large part of the available opportunism.

References

[1] P. Bender, P. Black, M. Grob, R. Padovani, N. Sindhushayana, and A. Viterbi. CDMA-HDR: A bandwidth-efficient high-speed wireless data service for nomadic users. *IEEE Communication Magazine,*, pages 70–77, July 2000.

[2] T. Bonald. A score-based opportunistic scheduler for fading radio channels. In *Proc. of European Wireless*, 2004.

[3] R. R. Boorstyn, A. Burchard, J. Liebeherr, and C. Oottamakorn. Statistical service assurances for traffic scheduling algorithms. *IEEE Journal on Selected Areas in Communications*, 18:2651 – 2664, Dec. 2000.

[4] D. Gesbert and M. Slim-Alouini. How much feedback is multi-user diversity really worth? In *Proc. Int. Conf. on Commun.*, pages 234–238, June 2004.

[5] E. Knightly and N. B. Shroff. Admission control for statistical QoS: Theory and practice. *IEEE Network*, 13:20 – 29, Mar. 1999.

[6] R. Knopp and P. Humblet. Information capacity and power control in single cell multi-user communications. In *Proc. IEEE International Computer Conference*, volume 1, pages 331 – 335, June 1995.

[7] D. Park, H. Seo, H. Kwon, and B. G. Lee. A new wireless packet scheduling algorithm based on the CDF of user transmission rates. In *Proc. IEEE Globecom*, pages 528–532, November 2003.

[8] S. Patil. Opportunistic scheduling and resource allocation among heterogeneous users in wireless networks, Ph.D. thesis, Univeristy of Texas at Austin. *available at http://www.ece.utexas.edu/˜ patil/Thesis.pdf*, 2006.

[9] S. Patil and G. de Veciana. Managing resources and quality of service in wireless systems exploiting opportunism. In *Submitted for journal publication, available at http://www.ece.utexas.edu/˜ patil/noniidQoS.pdf*.

[10] S. Patil and G. de Veciana. Measurement-based opportunistic scheduling for heterogenous wireless systems. In *Submitted for journal publication, available at http://www.ece.utexas.edu/˜ patil/measurement.pdf*.

[11] X. Qin and R. Berry. Opportunistic splitting algorithms for wireless networks. In *INFOCOM 2004. Twenty-Third Annual Joint Conference of the IEEE Computer and Communications Societies*, March 2004.

[12] X. Qin and R. Berry. Opportunistic splitting algorithms for wireless networks with heterogeneous users. In *Proc. Conference on Information Sciences and Systems (CISS)*, March 2004.

[13] S. Sanayei, A. Nosratinia, and N. Aldhahir. Opportunistic dynamic subchannel allocation in multiuser OFDM networks with limited feedback. In *IEEE Information Theory Workshop*, October 2004.

[14] P. Svedman, S. K. Wilson, L. Cimini, and B. Ottersten. A simplified feedback and scheduling scheme for OFDM. In *IEEE Vehicular Technology Conference*, May 2004.

[15] T. Tang and R. W. Heath. Opportunistic feedback for downlink multiuser diversity. *IEEE Communication Letters*, 9:948–950, 2005.

TABLE I

OPTIMUM VALUES OF QUANTILE THRESHOLD q

$\frac{n}{k}$	1	2	3	4	5	6	7
$k = 2$	0	0.6796	0.7591	0.8064	0.8380	0.8606	0.8777
$k = 3$	0	0.6931	0.7689	0.8134	0.8432	0.8647	0.8810
$k = 4$	0	0.7069	0.7791	0.8211	0.8491	0.8693	0.8847
$k = 5$	0	0.7205	0.7895	0.8291	0.8554	0.8744	0.8888
$k = 6$	0	0.7337	0.7998	0.8372	0.8620	0.8798	0.8934
$k = 7$	0	0.7462	0.8097	0.8452	0.8686	0.8854	0.8981
$k = 8$	0	0.7580	0.8191	0.8530	0.8752	0.8910	0.9030
$k = 9$	0	0.7690	0.8279	0.8604	0.8815	0.8965	0.9078

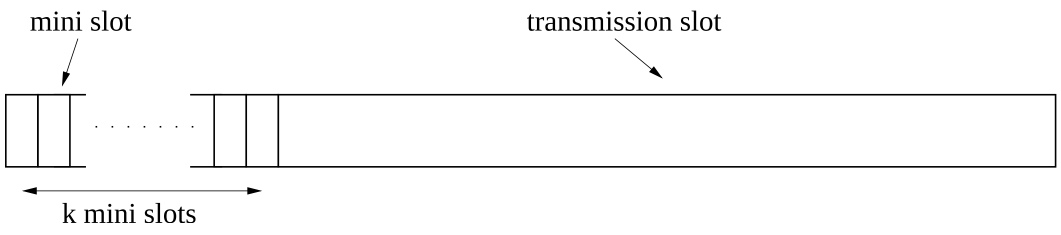

Fig. 1. Structure of a time unit.

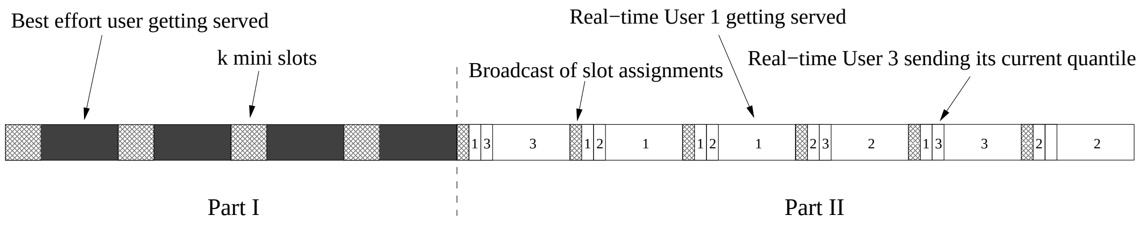
Best effort user getting served
k mini slots
Real−time User 1 getting served
Broadcast of slot assignments
Real−time User 3 sending its current quantile
Part I
Part II

Fig. 2. Example of the JPOS scheme with frame size of 10 time units and 3 real-time users having 2 tokens each.

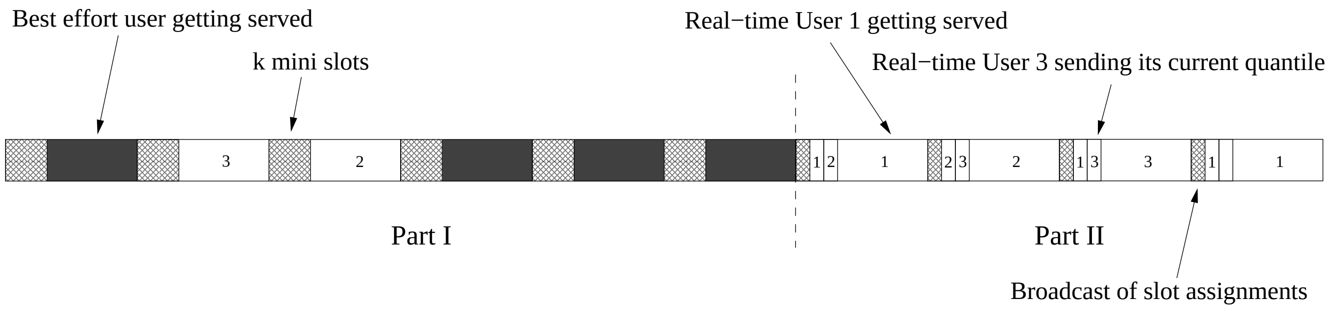
Best effort user getting served
k mini slots
Real−time User 1 getting served
Real−time User 3 sending its current quantile
Part I
Part II
Broadcast of slot assignments

Fig. 3. Example of the heuristic based JPOS scheme with frame size of 10 time units and 3 real-time users having 2 tokens each.

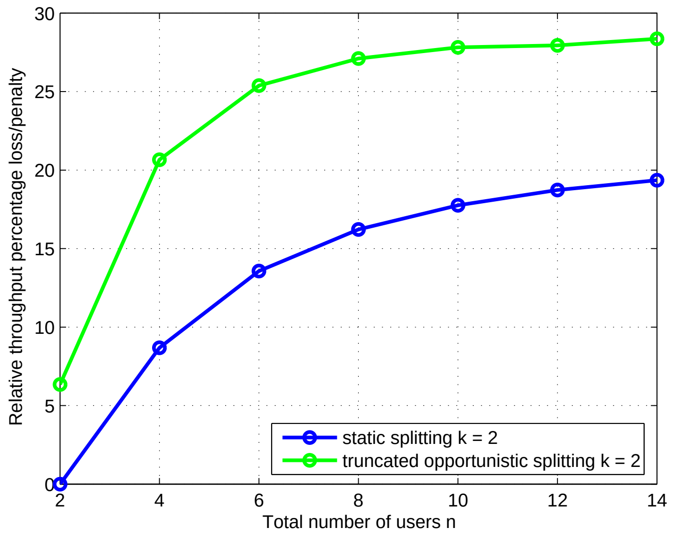

Fig. 4. The relative percentage throughput loss due to static splitting and truncated form of opportunistic splitting for $k = 2$ mini slots and an increasing number of users.

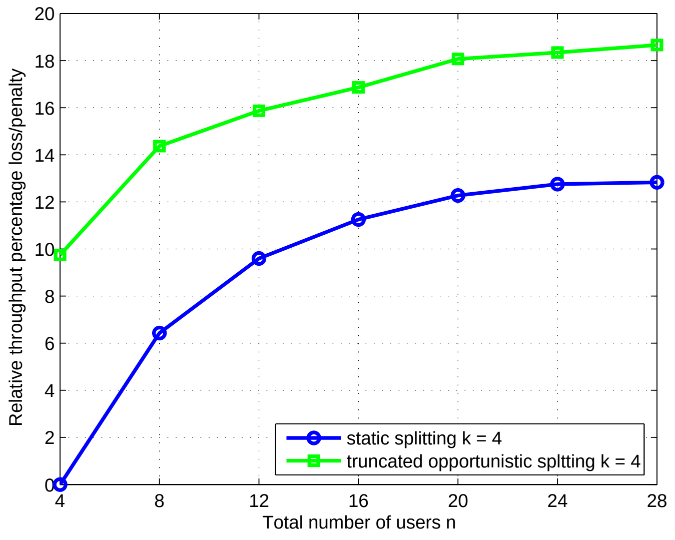

Fig. 5. The relative throughput percentage loss due to static splitting and truncated form of opportunistic splitting for $k = 4$ mini slots and an increasing number of users.

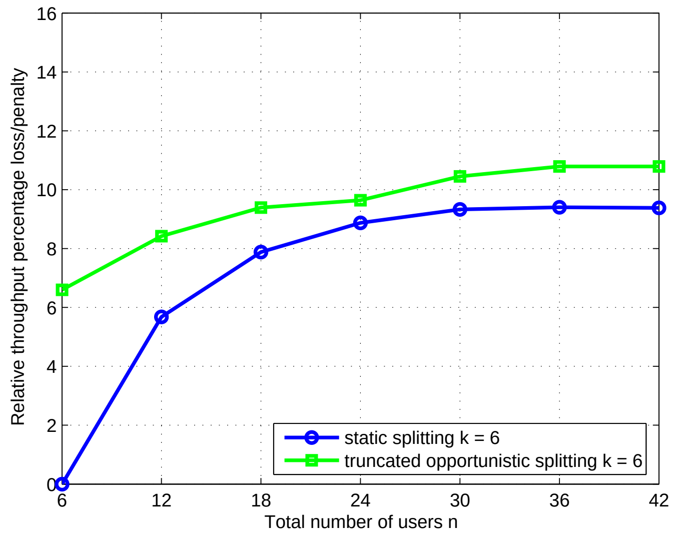

Fig. 6. The relative throughput percentage loss due to static splitting and truncated form of opportunistic splitting for $k = 6$ mini slots and an increasing number of users.

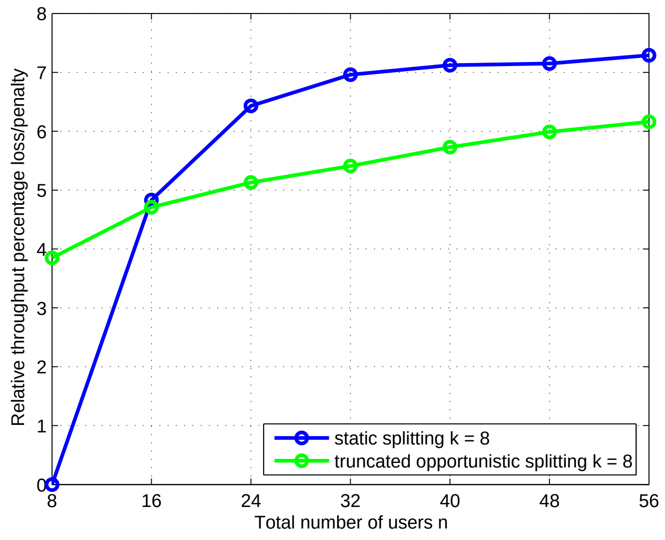

Fig. 7. The relative throughput percentage loss due to static splitting and truncated form of opportunistic splitting for $k = 8$ mini slots and an increasing number of users.

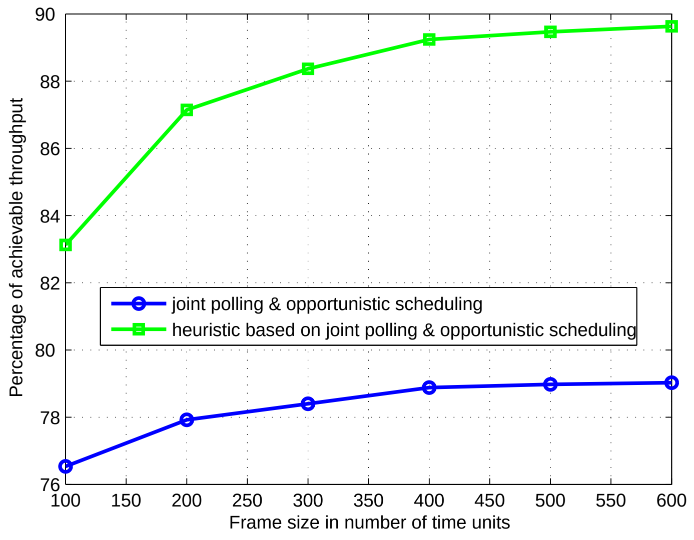

Fig. 8. Percentage of achievable throughput realized by the JPOS scheme and it heuristic modification.