Channel Predictive Proportional Fair Scheduling

Hans Jørgen Bang, Torbjörn Ekman, and David Gesbert

***Abstract*— Recent work on channel modeling and prediction has shown the feasibility of predicting the mobile radio channel, quite accurately, several milliseconds ahead in time for realistic Doppler spreads. Motivated by these results we consider opportunistic scheduling algorithms that exploit both current and future channel estimates. We demonstrate how this extra channel information can be used to improve the scheduling. Simulations show that the proposed algorithm can improve the inherent tradeoff between throughput, fairness and delay. The current approach builds on proportional fair scheduling but can also be generalized to other criteria.**

***Index Terms*— Channel prediction, fairness, multiuser diversity, proportional fair scheduling.**

I. Introduction

OPPORTUNISTIC scheduling has recently attracted much attention as a means to increase the spectrum efficiency of wireless data networks. By allocating the common radio resource to the users that are currently best able to utilize it the inherent multiuser selection diversity is exploited [1], [2]. There is however a fundamental tradeoff between throughput on one hand and fairness and delay on the other. Techniques that are able to push this tradeoff (improving one without sacrificing the other) are therefore of great interest.

Several channel-aware scheduling techniques for the downlink of wireless data networks have already been proposed in the literature. For a recent survey see [3]. One of the most popular ones is the *Proportional Fair Scheduler* (PFS) [1]. In this approach a time constant parameter is chosen to specify over what time period fairness between the users should be maintained.

In parallel and much independently, recent work on channel modeling and prediction has shown the feasibility of predicting fading channels over horizons of up to 0.25 wavelengths with reasonable accuracy [4]–[6]. Combined with reduced feedback delays and shorter scheduling intervals in proposed 4G systems [7], this opens up the possibility of combining channel prediction with resource allocation. In this paper we examine how channel predictions can help to improve the tradeoff between throughput and fairness. More specifically we develop a framework for scheduling algorithms that rely on both current and future channel estimates. The intuition behind our approach is that future channel estimates, even imperfect ones, are beneficial since they allow the scheduling to be planned over several time slots ahead in time. To the best of our knowledge, this idea has not been investigated before with the exception of [8]. We make the following key points:

- For scenarios where fairness is to be maintained over long periods of time compared to the coherence time of the fading, capacity maximization can be obtained through the use of memoryless schedulers. Thus, channel prediction is of minor interest[1].
- For scenarios where tighter fairness and delay constraints are used, there is a significant gain in terms of throughput to be obtained from a channel prediction-aware scheduler.

Clearly, there are some major challenges associated with prediction-aware scheduling. First, the quality of the predictions degrade rapidly with the prediction horizon. Thus, robustness with respect to prediction errors is crucial. Second, the complexity of the scheduling tends to increase significantly when additional channel information is introduced. In this paper we address both these issues. Specifically, we propose a generalization of the PFS algorithm that incorporates future channel estimates. We demonstrate that this algorithm is capable of increasing the throughput without compromising fairness and delay compared to the standard PFS algorithm.

II. System Model and Channel-Aware Scheduling

A. System Model

We consider the downlink of a single cell with N simultaneously active users served by one base station (BS). The scheduling is organized on a slot by slot basis, i.e. one and only one user is served during any given slot. The scheduler resides at the BS and decides prior to each slot which user the BS shall transmit data to. We use $i^*(k)$ to denote the user scheduled in slot k.

The BS operates at fixed transmit power and employ rate adaption to adjust to instantaneous channel conditions. Our key assumption is that estimates of the users' supported data rates for the current and $L-1$ future slots are available to the scheduler. The supported rate for the ith user in slot $k+l$, as predicted in slot k, will be denoted $\hat{R}_i(k+l|k)$. We use $R_i(k)$ as shorthand for $\hat{R}_i(k|k)$. For an arbitrary vector $\mathbf{i}$ we let $(\mathbf{i})_l$ denote its lth component.

B. Channel-Aware Scheduling

In order to realize a multiuser diversity gain the the scheduling criterion must be a function of the users' current supported

Manuscript received September 20, 2006; revised March 30, 2007 and June 7, 2007; accepted August 31, 2007. The associate editor coordinating the review of this letter and approving it for publication was Y. Ma. This work was presented in part at SPAWC'2005, New York, June 2005.

H. Bang is with the UniK-University Graduate Center, University of Oslo, Norway (e-mail:hans@unik.no).

T. Ekman is with the Dept. of Electronics and Telecommunications, Norwegian University of Science and Technology, Norway (e-mail: torbjorn.ekman@iet.ntnu.no).

D. Gesbert is with the Eurécom Institute, Sophia Antipolis, France (e-mail: david.gesbert@eurecom.fr).

Digital Object Identifier 10.1109/TWC.2008.060729.

[1]In a practical system channel prediction must still be employed to obtain estimates of the users' current channel conditions due to non-zero feedback delay

rates. A straightforward approach is to let

$$i^*(k) = \arg \max_{i=1,.,N} f_i(R_i(k)), \tag{1}$$

where $f_i(\cdot)$ is a monotonically increasing function independent of time k. In particular for $f_i(x) = x$ we obtain the Max SNR scheduler which select the user with the highest supported rate directly. However other choices of $f_i(\cdot)$ might lead to better long-term fairness properties. Recently several memoryless channel-aware scheduling policies that maximizes capacity under various long-term fairness criteria have been proposed [2], [9], [10].

A main limitation with memoryless schedulers is that fairness can only be ensured over long time windows compared to the coherence time of the fading. To improve the short term performance the priority of users that has not been selected for a long time must raised. This is exemplified by the PFS algorithm. In this approach the user with the highest supported rate relative to past average throughput is selected in each time slot. Thus, the user scheduled in time slot k is given by

$$i^*(k) = \arg \max_{i=1,.,N} \frac{R_i(k)}{T_i(k)}, \tag{2}$$

where $T_i(k)$ is user i's past average throughput. The average throughputs are updated in each time slot according to

$$T_i(k+1) = (1 - \frac{1}{t_c})T_i(k) + \frac{1}{t_c} R_i(k)\delta(i - i^*(k)), \tag{3}$$

where $\delta(\cdot)$ is the Kronecker delta function and t_c is a predetermined constant. The particular value of t_c determines the time horizon over which the throughputs are computed and gives a tradeoff between long-term throughput and delay. Even though the above scheduling criterion can be easily motivated there exists a second formulation that provides additional insight into the nature of the PFS algorithm [11], [12]. To this end we consider the following system utility function

$$\mathcal{U}(k) = \sum_{j=1}^{N} \log T_j(k). \tag{4}$$

It can then be shown (see *Proposition 2*) that (2) is equivalent to selecting the user that leads to the largest instantaneous increase in $\mathcal{U}$. More precisely this can be formulated as

$$i^*(k) = \arg \max_{i=1,.,N} \mathcal{U}(k+1). \tag{5}$$

This suggests that the underlying goal of the PFS algorithm is to maximize the system utility function $\mathcal{U}$.

III. Prediction-Aware Scheduling

In an attempt to improve the scheduling even further, we now turn to prediction-aware schedulers. This class of schedulers exploit both current rate estimates and predictions of the users' future supported rates. We will in the following refer to these schedulers as being simply predictive.

A. Predictive Block-Based Scheduling

We first consider an approach where adjacent time slots are grouped into non-overlapping blocks of length L time slots. Every Lth time slot (say time slot k) a scheduling vector

$$\mathbf{i}^*(k) = \{i^*(k+l)\}_{l=0}^{L-1} \tag{6}$$

dictating the next L transmissions is computed according to

$$\mathbf{i}^*(k) = \arg \max_{\mathbf{i} \in \mathcal{F}} \mathcal{O}^{(k)}(\mathbf{i}), \tag{7}$$

where $\mathcal{O}^{(k)}(\mathbf{i})$ is an appropriate objective function and $\mathcal{F}$ is the set of feasible scheduling combinations. In time slot $k+L$ a new scheduling vector determining the schedule for the next block of time slots is computed. The main motivation for the above approach is that the scheduling can be better planned if several scheduling decisions are made simultaneously. The objective function should reflect the underlying goals of the scheduling and can be a function of a number of parameters including predictions of the users future rates.

B. Predictive Proportional Fair Scheduling

In this subsection we extend the PFS algorithm to a predictive scheduling scenario. We first define the following two quantities

$$\tilde{T}_i(k|\mathbf{i}) = (1 - \frac{1}{t_c})T_i(k) + \frac{1}{t_c} \sum_{l=0}^{L-1} w_l \hat{R}_i(k+l|k)\delta(i - (\mathbf{i})_{l+1}) \tag{8}$$

$$\mathcal{O}_{pf}^{(k)}(\mathbf{i}) = \sum_{i=1}^{N} \log \tilde{T}_i(k|\mathbf{i}), \tag{9}$$

where $w_0, \ldots, w_{L-1}$ are positive weights in (8). With $\mathcal{O}_{pf}^{(k)}(\mathbf{i})$ as objective function we can directly adopt the block-based scheduling strategy. Note that for $w_0 = 1$ and $L = 1$ we obtain (5) and hence the PFS algorithm as a special case. For $L > 1$ we obtain a family of predictive schedulers depending on the particular combination of weights w_l. The next result illustrates an even deeper connection with the PFS algorithm.

Proposition 1: Assume that the rate predictions are perfect and let

$$\mathbf{i}^*(k) = \arg \max_{\mathbf{i} \in \mathcal{F}} \mathcal{O}_{pf}^{(k)}(\mathbf{i}), \tag{10}$$

with $w_l = (1 - \frac{1}{t_c})^{-l}$ for $l = 0, \ldots, L-1$. Then $\mathbf{i}^*(k)$ maximizes $\mathcal{U}(k+L)$.

Proof: See Appendix 1. ■

Thus, for the weights w_l specified in *Proposition 1* the scheduling vector $\mathbf{i}^*(k)$ is chosen to maximize $\mathcal{U}(k+L)$. Observe that this is a natural generalization of the standard PFS algorithm where $i^*(k)$ maximizes $\mathcal{U}(k+1)$. Even though this seems to justify this particular choice of weights one important remark must be made. Since $(1 - \frac{1}{t_c})^{-1} > 1$ we can infer from (8) that increasing emphasis is put on more distant rate predictions. In the case of poor prediction accuracy this might not be the best practical design. Generally the optimum weights will be a function of the prediction quality.

C. Robust Predictive Scheduling

In the block based-scheduling strategy a schedule is fixed for a block of future time slots. Although conceptually simple there are certain drawbacks to this approach that we now address.

Most notably we rely on imperfect estimates of the users' supported rates. Although reliable short range predictors have been proposed, the predictions degrade rapidly for longer prediction ranges [4]–[6]. This is a common problem to all predictive schedulers, but to fix the schedule several time slots ahead might result in additional sensitivity to prediction errors. A better and more robust approach is to re-optimize the schedule in each time slot as the predictions are updated. Furthermore, estimates of the users' supported rates for the $L-1$ subsequent time slots are available in each slot, but only the scheduling decision for the first slot within a block utilizes this prediction range fully. In fact, the scheduling decision for the last slot within a block is independent of the rate estimates in the slots subsequent to it. A final concern is that of the complexity of the algorithm. In order to compute an optimal scheduling vector according to (7) an exhaustive search will generally be required. With a total of $|\mathcal{F}| = N^L$ potential scheduling combinations this is only practical for very short scheduling horizons.

Motivated by the first two points we propose an strategy where the schedule is re-optimized in each time slot in a receding horizon fashion. The algorithm is summarized below.

In each time slot k:

1) Update the rate predictions, $\hat{R}_i(k+l|k)$.
2) Search for $\mathbf{i}^*(k)$ that maximizes $\mathcal{O}^{(k)}(\mathbf{i})$.
3) Schedule the user given by the first component of $\mathbf{i}^*(k)$, i.e $i^*(k) = (\mathbf{i}^*(k))_1$.

Observe that a full schedule is computed in each slot, but only the scheduling decision for the current slot is implemented. In this way all available channel information is utilized in each time slot and for each scheduling decision. We further note that the joint computation of all components of the scheduling vector greatly affects the final outcome even though only the first component is directly used.

Clearly, this modification does not reduce the computationally complexity compared to the original block based version. However, the schedule from the previous time slot provides an excellent starting point for an iterative search of a new updated schedule in the current time slot. Based on this idea we next present a low complexity algorithm that renders at least locally optimum scheduling vectors $\mathbf{i}^*(k)$. Note that we will still use $\mathbf{i}^*(k)$ to denote the computed scheduling vector even though it might differ from (7).

D. A Suboptimal Algorithm for Obtaining $\mathbf{i}^*(\mathbf{k})$

As opposed to an exhaustive search we propose an iterative search algorithm, based on cyclic coordinate ascent. To initialize the search we use the schedule from the previous time slot. Then in each subsequent iteration one component of the scheduling vector is updated with the other components held fixed. This process is repeated in a cyclic fashion until we converge to solution. To describe the algorithm we use the following notation:

- $\mathbf{i}^n(k) = \big(i_1^n(k), \ldots, i_L^n(k)\big)$ denotes the computed scheduling vector after n iterations.
- For an arbitrary vector $\mathbf{i} = \big(i_1, \ldots, i_L\big)$, $\mathbf{i} \overset{l}{\leftarrow} i$ denotes the vector $\mathbf{i}$ with the lth component exchanged with i. Thus $\mathbf{i} \overset{l}{\leftarrow} i = \big(i_1, \ldots, i_{l-1}, i, i_{l+1}, \ldots, i_L\big)$.

TABLE I
ITERATIVE ALGORITHM FOR OBTAINING $\mathbf{i}^*(k)$

1) **Initialization**
To initialize the algorithm let
$$\mathbf{i}^0(k) = \big(i_2^*(k-1), i_3^*(k-1), \ldots, i_L^*(k-1), 1\big).$$
2) **Main iteration**
At each iteration recompute one component of the scheduling vector. For the $(n+1)$th iteration let
$$\mathbf{i}^{n+1}(k) = \mathbf{i}^n(k) \overset{l}{\leftarrow} i_l^{n+1}(k),$$
where $l = L - (n \bmod L)$ and
$$i_l^{n+1}(k) = \arg\max_{i=1,..,N} \mathcal{O}^{(k)}\big(\mathbf{i}^n(k) \overset{l}{\leftarrow} i\big). \qquad (11)$$
3) **Termination**
When $\mathbf{i}^n(k) = \mathbf{i}^{n-L}(k)$ we have $\mathbf{i}^m(k) = \mathbf{i}^n(k)$ for all $m \geq n$ and we have converged to a solution.

The resulting algorithm is given in Table I. Observe that, for each iteration, we either obtain the same or a better solution in the sense $\mathcal{O}^{(k)}\big(\mathbf{i}^{n+1}(k)\big) > \mathcal{O}^{(k)}\big(\mathbf{i}^n(k)\big)$. Hence the algorithm will necessarily converge since there are only a finite number of scheduling combinations. The solution will be locally optimal in the sense that it can not be improved by changing any single component. We can expect fast convergence as limited amount of new channel state information is introduced in each time step and we use the schedule computed in the previous time step to initialize the search[2]. The exact rate of convergence will depend on the prediction quality and the Doppler spread, but will be roughly linear in the product LN.

We next present a useful result that shows that the computational complexity can be further reduced with (9) as objective function.

Proposition 2: Let $\mathcal{O}_{pf}^{(k)}(\mathbf{i})$ be used as objective function in (11). Then (11) is equivalent to

$$i_l^{n+1}(k) = \arg\max_{i=1,..,N} \frac{\hat{R}_i(k+l-1|k)}{\bar{T}_i(k+L|\mathbf{i}^n(k) \overset{l}{\leftarrow} 0)}. \qquad (12)$$

Proof: See Appendix. ■

Thus, according to the above result, the complexity per iteration is equal to that of the standard PFS algorithm per slot. Nevertheless, this results in considerable computational savings compared to explicitly evaluating the function $\mathcal{O}_{pf}^{(k)}\big(\mathbf{i}^n(k) \overset{l}{\leftarrow} i\big)$ as suggested in (11).

[2]The last component of $\mathbf{i}^0(k)$ is set to 1. This particular value is arbitrary and will not affect subsequent iterations.

IV. Numerical Results and Discussion

In this section, we evaluate the robust version of the predictive PFS algorithm together with the iterative algorithm in Table I numerically. For a neutral design we let $w_l = 1$ for $l = 0, \ldots, L-1$. Note however that the optimum weights are likely to be dependent on the quality of the predictions. The simulation results are obtained for Rayleigh fading channels with time correlations given by Jakes' model. All users have symmetrical channels with average SNR 0 dB and time slot Doppler frequency product 0.01. This means that the terminals move one wavelength in 100 time slots, e.g. 5.5 km/h with a time slot length of 1 ms in the 2 GHz band. We use the Shannon Capacity to estimate the supported rates and we assume that the BS always has data to send to each user.

In order to generate realistic estimates of the users' future supported rates we use a linear FIR (Finite Impulse Response) MMSE predictor with 128 coefficients to predict the future complex fading gains from past noisy observations of the channel. The channel power gain used in the Shannon Capacity is obtained as the absolute square of the predicted complex fading gain. The quality of the prediction will depend on the Channel to Estimation Error Ratio which is set to 20 dB. The NMSE (Normalized Mean Square Error) for the complex fading gain prediction ten time slots ahead is roughly 10^{-2} but for one step ahead it is only 10^{-3}. Thus, the error in the estimated rate for one time slot ahead can be disregarded.

A. Qualitative Example of Predictive Scheduling

We first compare the performance of the predictive and standard PFS algorithm with a qualitative example. Consider a scenario with 15 users, $t_c = 100$ and $L = 21$. Fig. 1 shows a snapshot of the supported rates for one user, where a superimposed cross corresponds to an allocated slot. The standard algorithm is shown to the left and the predictive version to the right. By inspection there appears to be significant increase in performance by using prediction since the allocated slots are clustered more tightly around fading peaks. Fig. 1 also indicates that potential gain in performance by doing full searches for the optimum scheduling vectors is very limited.

B. Long-Term Throughput

We next compare the total long-term throughput as a function of the parameter t_c for the standard and the predictive PFS algorithm with 15 users. The prediction horizon $L-1$ is set to 10 slots for the predictive algorithm. On average 20 iterations were required to compute $\mathbf{i}^*(k)$ in each time slot. It can be seen from Fig. 2 that there is an increase in throughput with the predictive algorithm for all values of t_c. Note however, that the largest gains (20%) occur for smaller t_c values. This is intuitive because for large values of t_c both algorithms approach the Max SNR scheduler.

C. Short-Term Throughput

Depending on the application the throughput over some finite time scale might be the key metric of interest. To quantify this performance let $T_i(w)$ denote the throughput for user i over a window of w time slots. We then define the

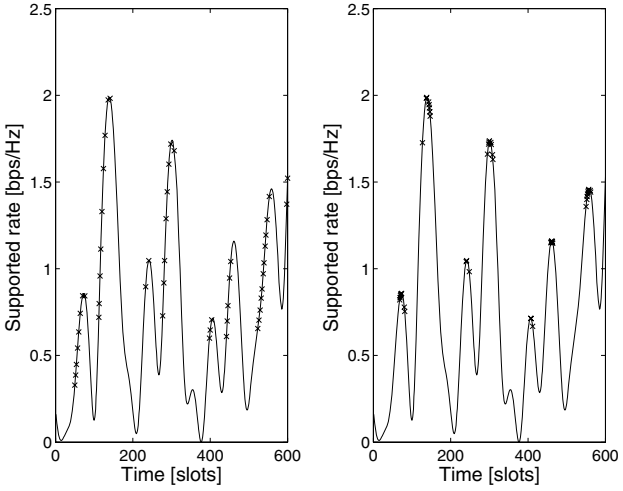

Fig. 1. Snapshot of supported rates for one user. A superimposed cross corresponds to an allocated slot with the standard (to the left) and the predictive PFS algorithm (to the right). The predictive algorithm leads to higher throughputs since the allocated slots are more tightly clustered around the fading peaks.

lowest achieved throughput (LAT) with outage probability ϵ according to

$$\Pr\{\mathcal{T}_i(w) < \mathrm{LAT}_i(\epsilon)\} < \epsilon. \tag{13}$$

Thus the probability that $T_i(w)$ falls below $\mathrm{LAT}_i(\epsilon)$ is given by ϵ. As w approaches infinity $\mathrm{LAT}_i(\epsilon)$ approaches the long-term throughput of user i, but for smaller time windows $\mathrm{LAT}_i(\epsilon)$ will be considerably less. $\mathrm{LAT}_i(\epsilon)$ is an interesting measure since it gives a good indication of the continuity of the incoming data flow and is tightly related to the maximum delay between two received packets for a user. Since we assume all users to have statistical identical channel we will drop the user-index i.

Fig. 3 shows LAT(0.01) as a function of window size for the Round-Robin algorithm, the standard PFS algorithm and the predictive PFS algorithm with $t_c = 100, 1000$. The Max SNR algorithm is not included as $\mathrm{LAT}(0.01)$ is zero for all window sizes in the considered range. Observe that the predictive PFS algorithm performs significantly better than standard PFS algorithm, for all but very small window sizes, for $t_c = 100$. This also hold true for $t_c = 1000$ although the improvement with the predictive algorithm is more modest.

D. Fairness

To quantify the degree of fairness between the users we use Jain's fairness index[13]. Let x_i be the resource of interest allocated to user i. Jain's fairness index is then defined as

$$J = \frac{\left(\sum_{i=1}^{N} x_i\right)^2}{N \sum_{i=1}^{N} x_i^2}. \tag{14}$$

Jain's fairness index measures the spread in the allocated resources and will always be in the range $1/N$ to 1. It is easily be verified that $J = 1$ indicates absolute fairness and $J = 1/N$ indicates no fairness, i.e all resources are allocated to one single user.

Since we assume all users to have statistical identical channels we can expect fairness between the users over longer

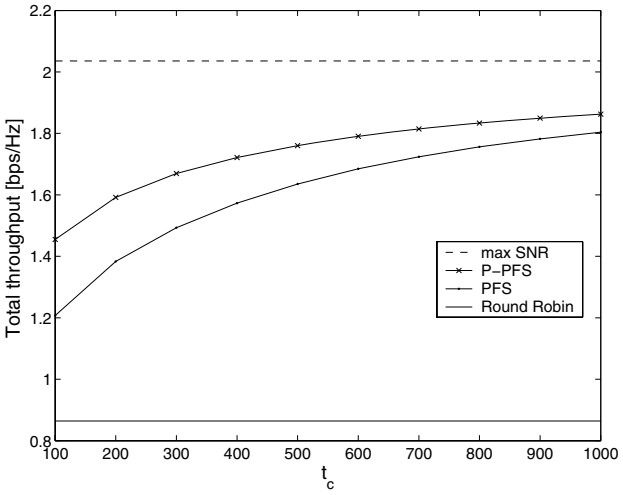

Fig. 2. Total long-term throughput as function of t_c for the predictive and the standard PFS algorithm with 15 users. The prediction horizon $L-1$ is set to 10 slots for the predictive PFS algorithm.

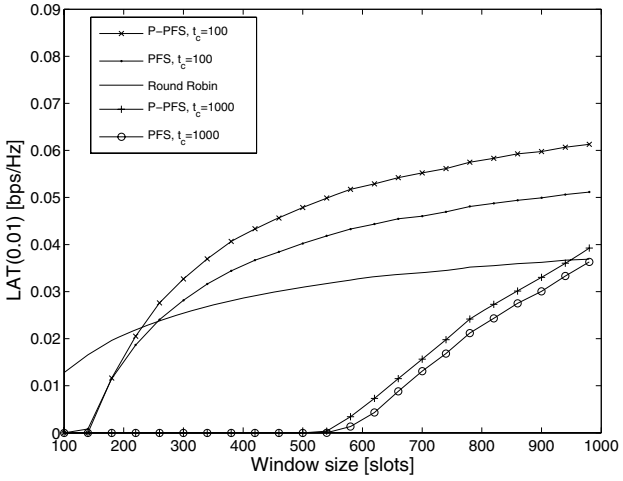

Fig. 3. LAT(ϵ) as a function of window size for $\epsilon = 0.01$ in a system with 15 users. The probability that the throughput of a single user falls below LAT(ϵ) over the specified window size equals ϵ. The prediction horizon $L-1$ is 10 slots for the predictive PFS algorithm.

time scales. To measure the degree of fairness over shorter time intervals we let $x_i = \mathcal{T}_i(w)$ and compute the mean over multiple realizations. Note that by adjusting the window size w the time scale that the fairness is computed over is adjusted accordingly.

Fig. 4 shows the average Jain's fairness index as a function of window size for the standard and predictive PFS algorithm with $t_c = 100, 1000$. Observe that the level of fairness is virtually the same for the two algorithms. There is a negligible reduction in fairness for $t_c = 100$ and a slight increase in fairness for $t_c = 1000$ with the predictive algorithm.

V. Conclusion

We have extended the PFS algorithm to a scenario where estimates of the users' future rates are available. At a reasonable increase in complexity and without compromising fairness or delay the total throughput was significantly increased compared to the standard PFS algorithm. The largest gains occurred when tighter fairness and delay requirements were imposed.

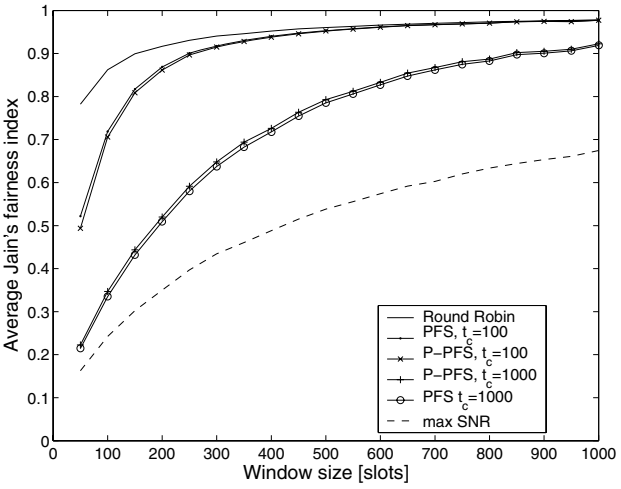

Fig. 4. Jain's fairness index as function of window size for the predictive and the standard PFS algorithm with 15 users and $t_c = 100, 1000$.

Appendix

A. Proof of Proposition 1

Let $w_l = (1-\frac{1}{t_c})^{-l}$ for $l = 0, \ldots, L-1$. We first note that

$$(1-\frac{1}{t_c})^{L-1}\tilde{T}_i(k|\mathbf{i}) = (1-\frac{1}{t_c})^{L}T_i(k) + \sum_{l=0}^{L-1}(1-\frac{1}{t_c})^{L-l}\hat{R}_i(k+l|k)\delta(i-(\mathbf{i})_{l+1}) \quad (15)$$

equals $T_i(k+L)$ given that the rate predictions are perfect and that user $(\mathbf{i})_l$ is scheduled in time slot $k+l-1$ for $l = 1, \ldots, L$. This is easily verified by solving (3) as a difference equation. Observe next that

$$\begin{aligned}\mathbf{i}^*(k) &= \arg\max_{\mathbf{i}\in\mathcal{F}} \mathcal{O}_{pf}^{(k)}(\mathbf{i}) \\ &= \arg\max_{\mathbf{i}\in\mathcal{F}} \sum_{i=1}^{N}\log\tilde{T}_i(k|\mathbf{i}) \\ &= \arg\max_{\mathbf{i}\in\mathcal{F}} \sum_{i=1}^{N}\log\left((1-\frac{1}{t_c})^{L-1}\tilde{T}_i(k|\mathbf{i})\right)\end{aligned} \quad (16)$$

This proves that $\mathbf{i}^*(k)$ maximizes $\mathcal{U}(k+L)$ if the rate predictions are perfect.

B. Proof of Proposition 2

Observe that

$$\tilde{T}_j(k+L|\mathbf{i}^n(k) \overset{l}{\leftarrow} i) = \hat{T}_j(k+L|\mathbf{i}^n(k) \overset{l}{\leftarrow} 0) \quad (17)$$

for $j \neq i$ and

$$\tilde{T}_j(k+L|\mathbf{i}^n(k) \overset{l}{\leftarrow} i) = \hat{T}_j(k+L|\mathbf{i}^n(k) \overset{l}{\leftarrow} 0) + \frac{1}{t_c}w_l\hat{R}_j(k+l-1|k) \quad (18)$$

for $j = i$. We can therefore write

$$\mathcal{O}_{pf}^{(k)}\left(\mathbf{i}^n(k) \overset{l}{\leftarrow} i\right) = \sum_{j=1}^{N} \log \tilde{T}_j(k|\mathbf{i}^n(k) \overset{l}{\leftarrow} 0) + \log\left(1 + \frac{\hat{R}_i(k+l-1|k)}{\tilde{T}_i(k+L|\mathbf{i}^n(k) \overset{l}{\leftarrow} 0)}\right). \quad (19)$$

This completes the proof since only the last term in the expression above depends on i and $\log(\cdot)$ is a monotone increasing function.

REFERENCES

[1] P. Viswanath, D. Tse, and R. Laroia, "Opportunistic beamforming using dumb antennas," *IEEE Trans. Inform. Theory*, vol. 48, no. 6, pp. 1277–1294, 2002.

[2] X. Liu, E. Chong, and N. Shroff, "Opportunistic transmission scheduling with resource-sharing constraints in wireless networks," *IEEE J. Select. Areas Commun.*, vol. 19, no. 10, pp. 2053–2064, 2001.

[3] X. Lin, N. B. Shroff, and R. Srikant, "A tutorial on cross-layer optimization in wireless networks," *IEEE J. Select. Areas Commun.*, vol. 24, no. 8, pp. 1452–1463, Aug. 2006.

[4] A. Duel-Hallen, S. Hu, and H. Hallen, "Long-range prediction of fading signals," *IEEE Signal Processing Mag.*, vol. 17, no. 3, pp. 62–75, 2000.

[5] R. Vaughan, P. Teal, and R. Raich, "Short-term mobile channel prediction using discrete scatterer propagation model and subspace signal processing algorithms," in *Proc. VTS*, Sep. 2000, pp. 751–758.

[6] T. Ekman, "Prediction of mobile radio channels, modeling and design," Ph.D. dissertation, Uppsala University, Sweden, 2002 [Online]. Available: http://www.signal.uu.se/Publications/abstracts/a023.html

[7] M. Sternad, T. Svensson, and G. Klang, "The WINNER b3g system MAC concept," in *Veh. Technol. Conf., VTS-Fall 64th VTC*, Sep. 2006, pp. 1–5.

[8] N. C. Ericsson, A. Ahlen, S. Falahati, and A. Svensson, "Hybrid type-II ARQ/AMS supported by channel predictive scheduling in a multi-user scenario," in *Veh. Technol. Conf. IEEE VTS-Fall 52nd VTC*, Sep. 2000, vol. 4, pp. 1804–1811.

[9] S. Borst, "User-level performance of channel-aware scheduling algorithms in wireless data networks," *IEEE/ACM Trans. Networking*, vol. 13, no. 3, pp. 636–647, 2005.

[10] D. Park, H. Seo, H. Kwon, and B. G. Lee, "Wireless packet scheduling based on the cumulative distribution function of user transmission rates," *IEEE Trans. Commun.*, vol. 53, no. 11, pp. 1919–1929, 2005.

[11] H. Kushner and P. Whiting, "Convergence of proportional-fair sharing algorithms under general conditions," *IEEE Trans. Wireless Commun.*, vol. 3, no. 4, pp. 1250–1259, July 2004.

[12] H. Kim and Y. Han, "A proportional fair scheduling for multicarrier transmission systems," *IEEE Commun. Lett.*, vol. 9, no. 3, pp. 210–212, Mar. 2005.

[13] R. Jain, *The Art of Computer Systems Performance Analysis*. New York: John Wiley and Sons, 1991.