Analysis of a Distributed Wireless Fair Scheduling Scheme

György Miklós
Traffic Analysis and Network
Performance Lab, Ericsson Research
H-1037 Budapest, Laborc u. 1., Hungary
Email: Gyorgy.Miklos@eth.ericsson.se
Tel: +36 1 437 7633, Fax: +36 1 437 7219

Sándor Molnár
High Speed Networks Lab,
Dept. of Telecommunications and Telematics,
Technical University of Budapest
H-1117 Budapest, Pázmány P. sétány 1/D, Hungary
Email: molnar@bme-tel.ttt.bme.hu

Abstract

We present an approach to wireless fair scheduling in a distributed architecture. Our scheme is based on a simple extension of wireline fair scheduling by compensation for lost capacity after the losses occur. In addition, the scheduler provides limited compensation for unused capacity. This provides an incentive for users of the location-dependent wireless channel to optimize their channel usage on their own, and in this way the scheduler does not need to be aware of the state of the wireless channel for each user. In this contribution we present the performance metrics by which the system can be analyzed, and show two analytically tractable abstract user behaviours that approximate the locally greedy user behaviour.

1 Introduction

Motivated by the growing use of wireless access technologies, the adaptation of wireline fair queuing algorithms to wireless environments has received increasing attention. Often it is the wireless link that becomes the bottleneck of an end-to-end flow which further emphasizes importance of the topic.

Wireless fair scheduling is different from wireline fair scheduling in several aspects. The scheduler has to take into account not only the quality of service requirements of the individual flow, but also the fact that the quality of the wireless channel may change over time and errors are typically location-dependent. Furthermore, the MAC layer of the specific link may impose additional constraints.

A number of recent papers deal with the adaptation of wireline fair scheduling algorithms to a wireless environment (e.g. [4, 8], see [7] for a comparative overview). They are based on the assumption that errors are typically bursty in nature, such that it is possible for the scheduler to predict the state of the channel for each user. As long as the number of errors are bounded from above, a minimum throughput and a maximum delay guarantee can be given to a flow.

In this paper we start from different assumptions and arrive at a different architecture. We do not assume any bound on the number of errors and accordingly suppose that the quality of the wireless channel will be visible to the end applications that can adapt to the changes in the service quality. Our purpose is not to hide the effects of wireless channel errors but to give partial compensation so that the otherwise unfair service (due to location-dependent errors) is made fairer.

Observe that bandwidth allocation over a lossy channel introduces a fundamental trade-off between fairness and utilisation: users with higher error rates need to be allocated more capacity to get the same fair amount of service but this decreases overall utilization; and vice versa: utilization is increased by allocating more capacity to users with better channels, which makes the allocation less fair. The actual amount of compensation is based on the service provider's choice in this trade-off. Our aim here is to extend wireline fair scheduling algorithms by a simple mechanism for compensation to improve fairness, so that this tradoff between fairness and utilisation can be controlled by the service provider.

In [5] we introduced a simple compensation mechanism where the master scheduler in the base station compensates for losses over the air interface *after they occur*. We analysed the trade-off between fairness and utilisation and the impact of link layer ARQ and TCP traffic.

In [6] we have extended this work into a distributed architecture. The master scheduler in the base station does not have any information or assumptions about the state of the wireless channel. Instead it compensates for channel errors after they occur. We extend the architecture by *allowing the users to defer their transmission to a later time when the channel is temporarily in a bad state*. Our approach is therefore decentralized in the sense that the master scheduler using a simple compensation mechanism is making the scheduling decision without any regard to the wireless channel state. Each user of the channel is responsible by itself for the estimation and prediction of its own channel, and can optimize when to transmit on its own. When the channel is expected to be bad for some time, the user can defer its transmission until the channel is expected to recover, based on the measured channel properties. This is encouraged (but not controlled) by the master scheduler as it allows the users to partially reclaim unused capacity in the future.

This approach offers several advantages. First, we do not need to make any assumptions about the error characteristics of the channel and the master scheduler does not need the prediction of the state of the link. This is useful since it is generally difficult to reliably predict the channel state in the future. Second, our scheme offers a modular implementation and greatly simplifies the master scheduler. Third, it offers a decoupling of functionality: the users' estimation and prediction of the channel can be changed without modifying the master scheduler.

In this paper we introduce performance metrics by which the algorithms can be analysed, and introduce two abstract but simple and analytically user behaviours that help assess the performance of the system. We present a simulation-based performance analysis.

The paper is organized as follows. Section 2 presents the system architecture. The master scheduling algorithm is described in Section 3. Section 4 gives a brief summary on how a user could optimize for itself the channel usage. The perfmance analysis of the system is discussed in Section 5, and Section 6 concludes the paper.

2 System architecture

In this subsection we briefly present the system architecture in which we applied our scheme. An example to such a system is the HiperLAN type 2 wireless LAN system [3], but the compensation algorithms can be adapted to other architectures as well.

Wireless access is provided in a cellular architecture, where an Access Point (AP) serves as the base station (BS) in each cell. APs are connected by a fixed network. Mobile Terminals (MTs) are associated with the AP of the cell they are in. All communication from and to a MT takes place through the AP.

The system uses TDMA/TDD (Time Division Multiple Access/Time Division Duplex) access, that is, all communications in a cell, both uplink and downlink, use the same frequencies, and are multiplexed in time as determined by the AP. The MAC protocol is frame-based, with a fixed-sized frame of 2ms. User data is transmitted in fixed-sized radio packets (referred to a PDUs, protocol data units) with 48 bytes of payload. Lost packets may be retransmitted as determined by the ARQ (automatic repeat request) protocol.

Each frame consists of a downlink and an uplink part. At the beginning of each frame, the AP announces how uplink and downlink transmissions are allocated within the current frame. Therefore communication is contention-free, with the exception of a short random access channel at the end of each frame, where the MTs can transmit control information to the AP.

The process of scheduling transmission resources in a MAC frame takes place as follows. The master scheduler within the AP takes as its input the number of packets queued for transmission for each user of the air interface. For downlink traffic this is readily available at the AP, while for uplink traffic this information is sent to the AP in control messages.

The master scheduler can run its scheduling algorithm frame by frame to determine how the resources are allocated. This is illustrated in Fig. 1. The first row represents the implementation of the scheduling process. It takes as input the resource requests from users transmitter either in the random access channel at the end of frames (shown in the figure) or earlier in the frame as control information (not shown). Depending on the implementation the scheduling process can incur one or more frames of delay which is not considered here. The output of the scheduler are the resource grants announced at the beginning of the new frame.

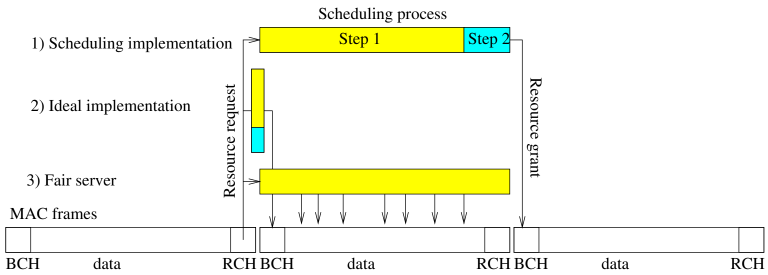


Figure 1: Scheduling. The horizontal direction represents time, with the MAC frames shown at the bottom. Each frame begins with a BCH (broadcast channel) where the frame structure is announced; frames end with a random access channel (RCH).

When applying a fair queuing algorithm in the master scheduler, we can think of the scheduling process in two steps. Based on the resource requests from the users, a virtual server running a fair queuing algorithm serves the users in the first step, and the amount of total service is determined for each user for the next frame. This is the amount of allocation each user will get in the frame, but the actual ordering of allocation within the frame may be changed in the second step. Most likely allocations for a given user will be grouped together to reduce overhead, and downlink and uplink transmissions will also be grouped together. The third row illustrates the fair server, as it makes the scheduling decisions during the frame. We address only this step in this paper, so it must be kept in mind that the actual transmissions may take place at a time of maximum one frame apart from when the server made the service.

3 Master scheduling algorithm

There are a number of wireline fair scheduling algorithms such as Weighted Fair Queuing (WFQ) [9] and others [10]. We have chosen Start-time Fair Queuing (SFQ) for our scheme, which will be motivated in Subsection 3.2. First, we introduce SFQ.

3.1 Start-time Fair Queuing

Start-time Fair Queuing [2] greatly reduces the computational complexity of WFQ by avoiding the need to simulate the fluid server in real time. Virtual time in SFQ is derived from the start tag of the packet in service. Another advantage of this method is that SFQ is applicable to variable rate servers without a need to take the server rate into account in the virtual time computation. [1]. The price paid for this simplicity is that the delay guarantee increases with number of flows. The fairness, throughput and delay properties of SFQ are analyzed in [2].

Start-time Fair Queuing is defined as follows [2]:

1 Packet j of flow i is stamped with a start and finish time S_i^j and F_i^j, respectively, upon arrival at time A_i^j.

$$S_i^j = \max\{v(A_i^j), F_i^{j-1}\},\ F_i^j = S_i^j + \frac{L_i^j}{w_i} \quad \text{for } j \geq 1 \tag{1}$$

where F_i^0=0, w_i is the weight of flow i, L_i^j is the length of packet j of flow i, and $v(t)$ is the virtual time at time t.

2 The server virtual time is initially 0. During a busy period the server virtual time at time t, $v(t)$, is defined to be equal to the start tag of the packet in service at time t. At the end of a busy period, $v(t)$ is set to the maximum of finish tag assigned to any packets that have been serviced by time t.

3 Packets are served in increasing order of start tags; ties are broken arbitrarily.

3.2 Fair Queuing in the AP master-scheduler

It is possible to use any of the fair scheduling algorithms in the first step of the AP master scheduler [10]. A number of differences from wireline fair queuing must be noted however:

(a) Scheduling has a granularity of a *user*, which may be coarser than the granularity of *flows*. *(b)* The scheduling process has no immediate information about the arrival of packets or the immediate backlogged status of queues. All that the server can take into account is whether users are satisfied or unsatisfied as compared to the resource requests that users make on a frame by frame basis. This is why we use the terms satisfied or unsatisfied, instead of backlogged or unbacklogged, for the status of the users in the scheduling process. *(c)* The scheduler is only responsible for the amount of allocations to the users, and the users of the air interface themselves can decide how to utilize the allocated capacity for new transmissions and retransmissions. Since the scheduler is in no control of the individual packets, it is better to use per user state information in the scheduling process, rather than per packet information (as is traditional with the start and finish time tags in fair queuing algorithms). *(d)* The computation of the system virtual time (i.e. the simulation of the fluid server) is made easier by the fact that users can become unsatisfied (i.e. equivalent of backlogged) only at frame boundaries, which must be also packet service boundaries. This follows that the satisfied status of a user changes only after the service of a packet. *(e)* The unsatisfied status of a user may change to satisfied even without the requested amount of service provided by the scheduler because the user may give up further transmission attempts of some packets at the expiration of some timer, or because the user may decide to defer its transmission to a later frame, as discussed later. When the virtual fluid server is simulated, this requires modifications since the set of unsatisfied users can change in the past in the sense that virtual service was given to a user but the corresponding real service cannot be given later. To solve this problem, the real service to a user that is no longer unsatisfied can be given to another user as a "present". This is not discussed further here.

Here we apply SFQ to our study. The choice is motivated by the fact that SFQ is not sensitive to the changes of the satisfied status of the users, i.e. the computation of the virtual time or the selection of the next user does not have to be modified as users become satisfied and unsatisfied (see (e) above), and that SFQ is computationally simple, yet provides fairness, throughput and delay guarantees. Although a computationally more expensive algorithm such as WFQ could provide tighter delay bounds, the frame-based architecture itself introduces a delay in any case which does not motivate a computationally more expensive algorithm.

We adapt SFQ to the master-scheduler of the AP in the following way.

1 A start and a finish tag, S_i and F_i, are associated with each *user*, corresponding to the virtual start and finish time of the packet at the head of the queue. When a new packet enters the head of the queue at time t (i.e. a packet has been served, or the user becomes unsatisfied), then the new values are computed from the old value of the finish time, F_i', as

$$S_i \leftarrow \max\{F_i', v(t)\},\ l \leftarrow \frac{L}{w_i},\ F_i \leftarrow S_i + l \tag{2}$$

where initially $F_i' = 0$, w_i is the weight of flow i, L is the length of all packets, l is the normalized length of the packet, and $v(t)$ is the virtual time at time t.

2 The server virtual time is initially 0. During a busy period the server virtual time at time t, $v(t)$, is defined such that when a packet from user i enters service, the system virtual time becomes S_i, the start tag corresponding to the packet. When the system is idle the virtual time is increased linearly such that $dv(t)/dt = C/\sum_{i\in\mathcal{U}} w_i$ where $\mathcal{U}$ is the set of all users.

3 Packets are served in increasing order of start tags; ties are broken arbitrarily.

Note that the start and finish tag computation was modified in so far as only the start and finish tag of the first packet of a user is maintained. (But even when the last packet of a user is served, its finish tag, denoted by F_i', must be maintained.) The virtual time computation was modified in that for an idle server, the virtual time is incremented linearly so that it reflects the increase in the minimum amount of virtual service that each user could have been given if it were unsatisfied (as will be discussed later).

3.3 Compensation of lost and unused resources

Location dependent errors on the wireless channel can alter the amount and distribution of resources utilized by the users, which motivates the need for a compensation mechanism in the scheduler. We extend the scheduler by two types of compensation: *compensation of lost and unused resources*.

Lost resources correspond to the packets which were not received correctly. They can become known to the AP scheduler through the ARQ protocol after the errors occur. By compensating for lost resources the differences of user goodput due to the different radio channel conditions can be reduced.

Unused resources correspond to the amount of service a user could have got while it did not request resources. By compensating unused resources later we give incentive to the users to monitor the state of their wireless links themselves and defer transmission when the link is temporarily in a bad condition.

To implement the compensation, we introduce the state variable *lag* for each user, denoted by n_i, to represent the amount of normalized service that the user should get in compensation. The constant β_l, $0 \le \beta_l \le 1$ represents the amount of compensation for lost resources. $\beta_l = 1$ means that all of the lost capacity is compensated later, $\beta_l = 0$ means no compensation for lost capacity. Similarly, the constant β_u represents the amount of compensation for unused resources, $0 \le \beta_u \le 1$. The constant γ, $0 \le \gamma \le 1$, is called the speed of compensation, and determines the increase of allocations when a user is being compensated. $\gamma = 0$ corresponds to no compensation, whereas $\gamma = 1$ corresponds to immediate compensation, where a user is compensated before any other users can get more allocations.

The scheduling algorithm is extended as follows.

4 After each error of a packet of length L on the wireless channel for user i as reported by the ARQ protocol, its lag is incremented by β_l times the normalized service that was lost:

$$n_i \leftarrow \min\{n_i + \beta_l L/w_i, n_{max}\}.$$

5 When user i becomes unsatisfied at the beginning of a frame at time t after being satisfied, its lag is incremented by β_u times the normalized service that the user missed:

$$n_i \leftarrow \min\{n_i + \beta_u \max\{v(t) - F_i', 0\}, n_{max}\}$$

where $v(t)$ is the virtual time at time t, and F_i' is the finish time of user i before it became satisfied. Since $v(t)$ can be interpreted as the normalized fair amount of service that each user could have received up to time t, $v(t) - F_i'$ is the amount of normalized service that the user missed while it was satisfied.

6 Equation eq. (2) is modified as follows.

$$l \leftarrow \frac{L}{w_i}, l_c \leftarrow \min\{n_i, \gamma l\},\ F_i \leftarrow S_i + l - l_c, n_i \leftarrow n_i - l_c, \tag{3}$$

where l is the normalized length of the packet, l_c is the normalized compensation given during the service of the packet.

This means that while a user is being compensated, the normalized service given when a single packet is served is artificially decreased by γ times the normalized length of the packet. This can also be interpreted as increasing the weight of the user from w_i to $w_i/(1-\gamma)$. The lag is decreased by the amount that was used up for the compensation.

The amount of lag is maximized by n_{max}. The purpose of this limit is to allow the compensation of lost and temporarily deferred transmissions but limit the amount of compensation for a users with little or no traffic for a long period of time.

As seen from the implementation of compensation a user is being compensated in such a way that its weight is in effect increased from w_i to $w_i/(1-\gamma)$. This is why the admission criterion for a new flow (which is in the case of SFQ $\sum_{i\in\mathcal{U}} w_i \leq C$) becomes $\frac{1}{1-\gamma}\sum_{i\in\mathcal{U}} w_i \leq C$.

4 User behaviour

The previous section has presented the master scheduling algorithm that provides two kinds of compensation to the users: compensation for lost and unused resources. Each user can observe the quality of its own channel through measurements and can decide how much capacity to request. Here we make the following simplifying assumptions: we assume that the success of the transmissions of a user is known immediately after the transmissions through the use of an ARQ mechanism on the link layer, there is no delay associated with the capacity requests, and that resource requests are never lost. A user either makes a resource request according to the packets waiting for transmission in its buffers, or makes a resource request of zero, thereby relinquishing service to a later frame.

Relinquishing service in a given frame can be useful for the mobile because channel behaviour is typically positively correlated, so when a user observes bad channel, it is likely that the channel will continue to be bad for some time.

We have presented a possible implentation of user behaviour in [6]. We summarize the algorithm here but do not detail it due to space limitations. The interested reader is referred to [6] for an elaboration and example.

The user behaviour is based on an AR(1) model of the radio channel. It is chosen as a heuristics model for a positively correlated chanel. It is simple enough to be handled computationally, yet powerful enough to capture the positively correlated nature of the channel.

The AR(1) process itself models the sequence of success ratios per frame, that is, the fraction of PDUs successfully sent by the users. By making use of the ARQ feedback, a user is able to determine the actual success ratios, and by that, estimate the parameters of the AR(1) channel model. This model enables the user to make predictions about the expected future change of the channel given the current measured channel quality.

Based on this model and prediction, a threshold-based decision rule is established. We have two thresholds to use a hysteresis. If the current success rate falls below the lower threshold, the user relinquishes services until the expected success rate rises above the upper thresholds. The two thresholds are set relative to each other in such a way that we expect a gain in throughput by the compensation mechanism. By changing the value of the thresholds, it is possible to alter the sensitivity of the algorithm, i.e. the frequency of relinquishing service in a frame.

We have also implemented an adaptive threshold calculation to set the sensitivity of the user algorithm automatically based on a one-bit feedback from the master scheduler. This method dynamically changes the threshold levels in such a way that the amount of compensation to a user approaches but does not exceed the amount that the scheduler can provide.

5 Performance analysis

There are a number of ways in which the performance of the proposed methods can be examined. The difficulty in the performance analysis is that the performance of the scheduling algorithm and user behaviour are closely coupled. Nevertheless it is possible to look at the performance of user behaviour separately.

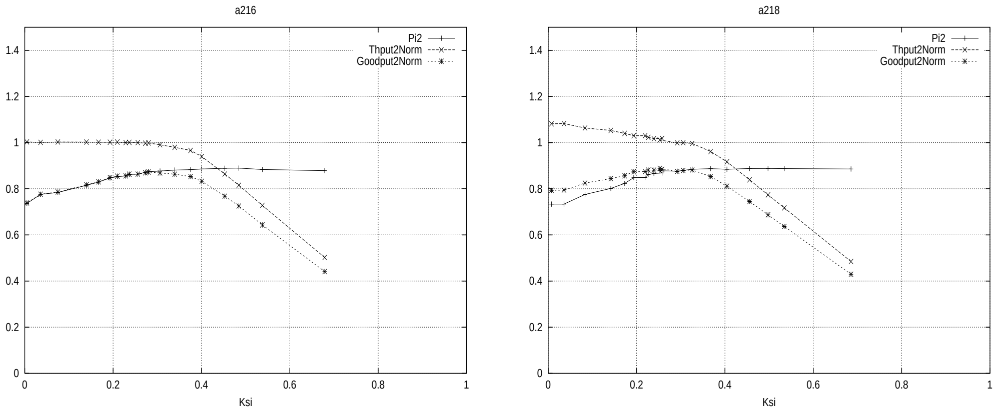
5.1 Performance of user behaviour

The net effect of the user relinquishing service is that, by making use of the positive correlation in the channel, the user can avoid transmitting when the channel is temporarily bad, and therefore the success rate of transmission is increased. This can be quantified by the plot of the user's observed success rate versus the sensitivity of the algorithm, i.e. the frequency of relinquishing service.

We have performed a simulation study to assess the performance of our algorithms. We have implemented our scheme in a packet-level simulation environment. The simulated architecute conforms to the architecture presented in Section 2 with frame based MAC protol with a frame length of 2ms. In our simulations, the capacity of the system was 32 PDUs/frame where a PDU can carry 48 bytes, giving a total system capacity of 6Mbps. This capacity is shared by several users. Users 1,3,5,... have a "good" channel with a PDU loss rate of 5% and independent losses. Users 2,4,6,... have a "bad" channel with bursty losses according to an ON-OFF Markovian model with and average loss rate of 25%. The average OFF period is 100 PDU transmission times, the loss rate is 95 % in the OFF period, 5% in the ON period. All users are greedy, which means that they transmit as many PDUs as possible.

Figure 2 shows the simulated performance of the user algorithm as a function of ξ, the frequency of relinquishing service. In this simulation there were 4 users, two with "good" channel and two with "bad" ones. The figure shows the success rate observed by the second user at it changes the sensitivity of its algorithm. (The other users employ automatic sensitivity setting.) It also shows the amount of allocation and amount of successfully carried traffic (goodput) normalized by the fair share of the user (i.e. 1/4 of the total capacity). In the simulations, $\gamma = 0.5$, β_u=1, and (a) $\beta_l = 0$ (b) $\beta_l = 0.5$.

The amount of improvement for the channel success rate shows the performance of the algorithm. (Note that this improvements are dependent on the parameter settings, the number



(a) (b)

Figure 2: Performance of user algorithm as a function of ξ, the frequency of relinquishing service. $\gamma = 0.5$, β_u=1, (a) $\beta_l = 0$ (b) $\beta_l = 0.5$.

of users, and channel properties.) We can observe that the algorithm works successfully by improving the success rate from 0.75 to 0.88.

The amount of allocation remains close to the fair share up to about $\xi = 0.35$, and then begins to decline. This decline is due to a saturation effect: above the saturation point, the master scheduler can not provide the compensation for unused resources during the time when the user is requesting service. Below the saturation point, the scheduler provides full compensation for the unused resources, as illustrated in Figure 2 (a) by the horizontal line at 1. Figure (b) on the other hand shows that this user is allocated even more resources than its fair share below the saturation point. This is because in this simulation, there is compensation for lost resources ($\beta_l = 0.5$) and the more errors there are on the channel, the more extra allocation the user receives.

The declining curve of the amount of allocation to the user multiplied by the increasing curve of improving transmission success rate gives us the change of user goodput (i.e. successfully carried traffic). The user is looking for the maximum of this curve in order to optimize its throughput locally. We call it ”locally greedy” user behaviour. Indeed, the adaptive sensitivity setting for this user finds the point at $\xi = 0.25$, which gives a nearly optimal local throughput performance.

Besides the locally greedy user behaviour, we introduce here two simple other behaviours as well for analytical purposes. The first is the ”no algorithm” case, that is, when the user uses no relinquishing of service. This simply corresponds to the $\xi = 0$ case.

The second is the ideal algorithm case which is interpreted as follows. In the simulation example, the user algorithm has been able to increase the success rate from 0.75 to 0.88. We can say that an upper limit on the achievable success rate is 0.95, since this is the success rate in the ON state in our actual channel model. So the quicker and close the success rate approaches 0.95 as ξ is increased from 0 to 1, the better the algorithm is. The limiting case is the one in which the success rate reaches 0.95 as ξ grows from 0 to an infinitesimal small value. This takes us to the ideal limiting case, for which $\xi = 0$ and the success rate is 0.95.

In general, we say that an ideal algorithm is one in which for $\xi = 0$, the success rate is no smaller that what can be achieved by any $\xi > 0$. Such an algorithm is naturally impossible to implement or even approach, but it provides a valuable abstraction for the analysis of the system.

5.2 System performance

The two abstract cases introduced above, the no algorithm and ideal algorithm cases, allow us to derive system performance metrics. In both of these cases, $\xi = 0$, i.e. the user does not give up service by itself. This follows that the total system capacity is distributed among the users based on their weights. However, we have to take into account the compensation mechanism for lost capacity.

Let us denote by Π_i the transmission success rate of user i. The normalized service without compensation to the user is L/w_i for one PDU. With probability $1 - \Pi_i$, the transmission is unsuccessul, in which case the lag of the user is increased by $\beta_l L/w_i$.

In order to derive the effect of compensation for lost capacity, we have to find the set of parameters where the compensation is saturated. This happens when the amount of lag generated is at least as much as the maximum amount of compensation that the scheduler can give to the user. The expected increase of the lag for one PDU is $(1 - \Pi_i)\beta_l L/w_i$, and the maximum compensation is given when the user gets compensation during the service of all PDUs, in which case the compensation if $\gamma L/w_i$ per PDU. This follows that compensation is saturated when $\gamma < (1 - \Pi_i)\beta_l$.

In this case, the normalized service with compensation for one PDU is $L/w_i - \gamma L/w_i$. If the compensation is not saturated, then the compensation equals the amount of lag generated, so normalized service for one PDU is $\Pi_i L/w_i + (1 - \Pi_i)(1 - \beta_l)L/w_i$. Taking the two cases together, the normalized service to the user is compensated in such a way that it can be modeled by the following modified weight of user i:

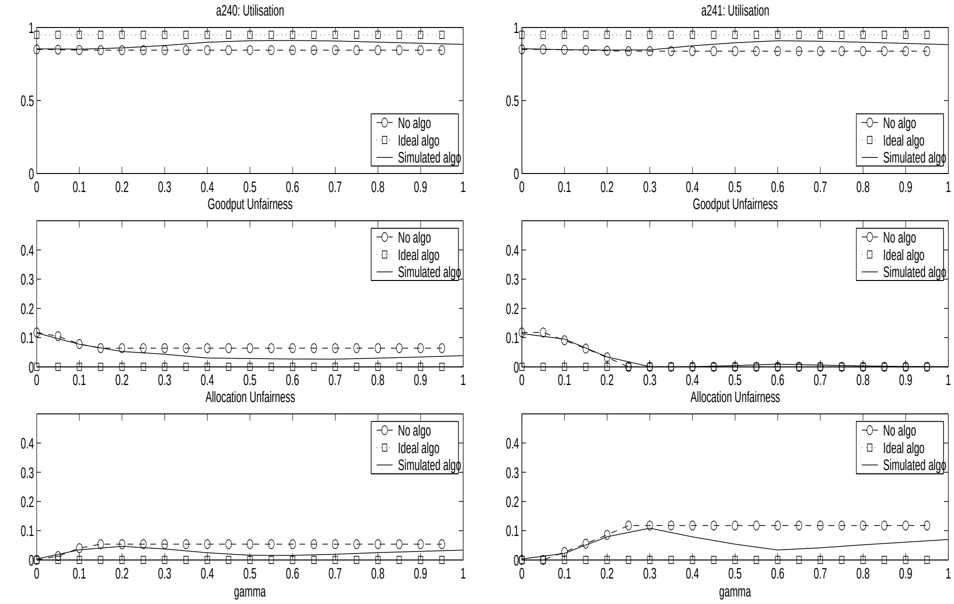
$$w'_i = \frac{1}{1 - \min\{\gamma, \beta_l(1 - \Pi_i)\}} w_i. \tag{4}$$

The total system capacity, C, is distributed among the users according to equation 4. Using $W' = \sum_i w'_i$, the allocation and goodput to user i (assuming greedy users) are

$$a_i = \frac{w'_i}{W'} C \text{ and } g_i = \Pi_i a_i. \tag{5}$$

The total system performance is characterized by the following metrics: the system utilization, defined by $U = \sum_i g_i/C$, the goodput unfairness $V_g = \sigma\{g_i/w_i\}/E\{g_i/w_i\}$, and allocation unfairness $V_a = \sigma\{a_i/w_i\}/E\{a_i/w_i\}$, that is, the coefficient of variation of normalized goodput and allocation values. The higher the coefficient of variation, the higher the unfairness and the worse the fairness is.

For a specific channel, we can use the following success ratios: $\Pi_{i,avg}$ for the average success rate of the channel, and $\Pi_{i,max}$ for the maximum success rate that can be achieved by any user behaviour. Such an upper limit can be interpreted, for example, for a Markovian channel by the success rate in the best state. By using the success rates $\Pi_{i,avg}$ and $\Pi_{i,max}$ for Π_i, the utilization and unfairness metrics can be derived for the no algorithm and ideal algorithm cases.



(a) (b)

Figure 3: System performance, (a) $\beta_l = 0.5$ (b) $\beta_l = 1$, four users.

Figure 3 (a) and (b) show these metrics for the case of four users, $\beta_l = 0.5$ and $\beta_l = 1$, respectively, and the same parameters as in subsection 5.1. The figures show the two abstract cases (no algorithm and ideal algorithm) together with the simulated locally greedy case. The figure shows that these abstract limiting cases provide a good hint for the actual system performance.

We can see that in both cases (a) and (b), total system utilization is significantly increased as γ goes from zero (no compensation) to 0.7. Further increase of γ does not increase the utilization, in fact, it can slightly decrease it. This is attributed to the fact that the specific user behaviour considered in this paper works better when the speed of compensation is not too fast (i.e. γ is lower than 1).

The fairness curves show that the algorithm, while keeping the increase in utilization, can significantly improve the goodput fairness of the system. It is seen that the fairness in goodput and allocation run counter to each other: improving one makes the other worse. For $\beta_l = 1$, almost complete fairness can be achieved for $\gamma > 0.3$. The figure clearly illustrates that it is possible to improve utilization and fairness simultaneously. This is possible because our architecture encourages efficient use of resources.

6 Conclusions

We have presented a scheme for the resource allocation of the capacity of a frame-based wireless base station. The scheme provides a modular architecture where the fair master-scheduler does not explicitly take into consideration the channel state of individual users. A traditional wireline scheduling algorithm in the master scheduler is extended to give compensation for unused and lost capacity, which gives an incentive to the users to optimize their channel usage on their own, without the explicit help of the base station.

In this contribution we have introduced metrics by which the performance of the user algorithm can be assessed, and we have defined the utilization and fairness metrics by which the system can be analyzed. We have introduced two simple abstract user behaviours, and we have implemented the locally greedy user algorithm in a packet-based simulator. We have shown that these abstract user behaviours can be handled analytically and provide a good approximation for the simulated locally greedy behaviour.

In the future we will continue our investigation by introducing new user behaviour methods, and extend the performance analysis by investigating global as well as local optimizations.

References

[1] Jon C. R. Bennett and Hui Zhang. Hierarchical packet fair queueing algorithms. *IEEE/ACM Transactions on Networking*, 5(5):675–689.

[2] Pawan Goyal, Harrick M. Vin, and Haichen Cheng. Start-time fair queuing: A scheduling algorithm for integrated services packet switching networks. *IEEE/ACM Transactions on Networking*, 5(5):690–704, October 1997.

[3] HiperLAN/2 Global Forum. http://www.hiperlan2.com/.

[4] Songwu Lu, Thyagarajan Nanadagopal, and Vaduvur Bharghavan. A wireless fair service algorithm for packet cellular networks. In *ACM MOBICOM'98*, August 1998.

[5] György Miklós and Sándor Molnár. Fair allocation of elastic traffic for a wireless base station. In *IEEE GLOBECOM'99*, December 1999.

[6] György Miklós and Sándor Molnár. Distributed fair bandwidth allocation of a wireless base station. In *Networking2000*, May 2000.

[7] Thyagarajan Nanadagopal, Songwu Lu, and Vaduvur Bharghavan. A unified architecture for the design and evaluation of wireless fair queueing algorithms. In *ACM MOBICOM'99*, August 1999.

[8] T. S. Eugene Ng, Ion Stoica, and Hui Zhang. Packet fair queueing algorithms for wireless networks with location-dependent errors. In *INFOCOM'98*.

[9] Abhay K. Parekh and Robert G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single node case. *IEEE/ACM Transactions on Networking*, 1:344–357, June 1993.

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