Distributed Fair Bandwidth Allocation of a Wireless Base Station

György Miklós
Traffic Analysis and Network Performance Lab, Ericsson Research
H-1037 Budapest, Laborc u. 1., Hungary
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
molnar@ttt-atm.ttt.bme.hu

Abstract. We present a novel 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 can give 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. In this way the scheduler does not need to be aware of the state of the wireless channel for each user. We present a mechanism for user behaviour and illustrate our scheme by simulation results.

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. [5], see [4] 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 [3] 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.

Our approach here is an extension of our previous work. 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.

The paper is organized as follows. Section 2 presents the system architecture. The master scheduling algorithm is described in Section 3. Section 4 shows how a user can optimize for itself the channel usage. Our simulation results are presented 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 [2], 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. It takes as input the resource requests from users transmitter either in the random access channel at the end of frames or earlier in the frame as control information. 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.

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. 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) and others [6]. 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 [1] 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. 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 [1].

Start-time Fair Queuing is defined as follows [1]:

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 [6]. 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, $0 \leq \beta_l \leq 1$ 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, $0 \leq \beta_u \leq 1$. The constant γ, $0 \leq \gamma \leq 1$, 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. Our purpose in this section is to arrive at a method for a user to decide when to relinquish service (i.e. make a resource request of 0). We assume that a user has knowledge of the scheduling algorithm and its constant parameters.

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.

In the following subsections, we present a simple solution to this problem where the user builds a model of the channel estimating the parameters, which enable it to make a prediction of the future expected channel state and decide when to relinquish service.

4.1 Channel model and prediction

We use a simple AR(1) (autoregressive) model. This is a very simple model, yet powerful enough to capture the correlated nature of the channel. Note that the AR(1) model and the quantitative approximation must be regarded as heuristics since we have no way of getting any a priory information on the channel properties and its stationary nature.

We model the success rate in every frame, that is, the fraction of successfully received PDUs out of the transmitted PDUs in every frame. The distribution of errors within a frame is not modeled.

The AR(1) process (established independently for each user) is written as

$$v[t+1] = \rho v[t] + \sigma z,\ s[t] = \pi + v[t] \tag{4}$$

where $v[t]$ is the actual AR(1) process, and $s[t]$ is the process of success rate. t is the frame counter (integer), z is a standard normally distributed random variable, ρ, σ, π are the parameters of the process. We make the simplification that we do not consider to be part of the AR(1) process those frames where the user is not scheduled.

The expected value, variance and correlation of the process $s[t]$ is given as

$$\text{E}\ s[t] = \pi,\ \text{Var}\ s[t] = \frac{\sigma^2}{1-\rho^2},\ \frac{\text{Cov}\ (s[t], s[t+k])}{\text{Var}\ s[t]} = \rho^k \tag{5}$$

We note that the process $s[t]$ could take on values outside the interval $[0, 1]$. However, we are going to use the model only to predict the expected value of the success rate in the future. According to Equation 10 in Subsection 4.1, it will be clear that the prediction cannot take on a value outside the interval $[0, 1]$ provided that π and the measured values of the process are within that interval.

We need to estimate the parameters of the model in such a way that as more and more measurements are available, the accuracy improves, on the other hand, the parameter estimation adapts to long-term, non-stationary changes in the channel behaviour.

To achieve this, we use exponential moving averages in all the parameter estimations with weight ϕ.

Estimations are based on $s_i^*[t]$, success rate measured in the last frame. π is estimated at frame t as

$$\pi[t] \leftarrow \pi[t-1]\phi + s_i^*[t](1-\phi) \tag{6}$$

Estimation of correlation is made indirectly through the estimation of second moment, δ, and first order mixed second moment (i.e. E $(s[t]s[t-1])$), ψ, of the process:

$$\delta[t] \leftarrow \delta[t-1]\phi + (s^*[t])^2(1-\phi),\ \psi[t] \leftarrow \psi[t-1]\phi + s^*[t]s^*[t-1](1-\phi) \tag{7}$$

The variance of the measured process is estimated as

$$\mu[t] = \delta[t] - (\pi[t])^2 \tag{8}$$

The following estimator can be shown to converge to the correlation if the measured process is AR(1):

$$\rho[t] = \frac{\psi[t] - \pi[t]\pi[t-1]}{\sqrt{(\delta[t-1] - (\pi[t-1])^2)(\delta[t] - (\pi[t])^2)}} \tag{9}$$

Given a current measurement of the success rate $s^*[t]$ and the estimations $\pi[t]$ and $\rho[t]$, the success rate for the frame $t+k$ can be predicted as

$$\hat{s}[t+k] = (s^*[t] - \pi[t])(\rho[t])^k + \pi[t] \tag{10}$$

4.2 User decision

Based on the channel measurements and predictions, a user must decide whether to relinquish transmission in the next frame in the hope of more efficient transmission later. Our criterion aims at maximizing throughput for the user. (Here we do not consider explicit delay guarantees for a user, though they may be incorporated as well by imposing additional rules that do not allow a user to defer transmission when the delay bound could be violated.)

In order to be able to provide a decision function that is applicable by a user without any explicit information about other users or the future, we make several simplifying assumptions. We assume that compensation is not yet limited by the maximum lag; and we do not consider changes in the success rate during compensation for unused service; furthermore we approximate the service given to a user for a compensation of x to be $x\pi[t]$, that is, success rate during compensation is approximated by the estimated average success rate $\pi[t]$. Our decision will be based on the expected service with compensation for one PDU in a frame.

If the user decides to transmit in the next frame, the expected service per packet in the frame is $L\hat{s}[t+1]$ (where L is the packet length), and the expected lost service is $L(1-\hat{s}[t+1])$. The expected compensation is $L(1-\hat{s}[t+1])\beta_l$ giving an expected service of $L(1-\hat{s}[t+1])\beta_l\pi[t]$ since we assume that the success rate during compensation for lost resources is $\pi[t]$. So the expected service per PDU is $L(\hat{s}[t+1] + (1-\hat{s}[t+1])\beta_l\pi[t])$. If the user defers transmission to a later frame $t+d$, the total service received instead of the allocation of a PDU in the current frame is less by the factor β_u

since the service is being compensated for unused capacity. So the expected service per PDU is $L\beta_u(\hat{s}[t+d]+(1-\hat{s}[t+d])\beta_l\pi[t])$. Transmission is relinquished in the current slot if the expected service per PDU is increased:

$$L(\hat{s}[t+1]+(1-\hat{s}[t+1])\beta_l\pi[t]) < L\beta_u(\hat{s}[t+d]+(1-\hat{s}[t+d])\beta_l\pi[t]) \qquad (11)$$

which gives

$$\frac{\hat{s}[t+1]}{\beta_u}+\frac{\frac{1}{\beta_u}-1}{\frac{1}{\beta_l\pi[t]}-1} < \hat{s}[t+d] \qquad (12)$$

Deferring transmission in the current frame is useful when the channel model suggests that at a later frame $t+d$ the expected success rate fulfills the above equation.

We introduce a threshold-based decision criterion for a user using an upper threshold θ_1 and a lower threshold θ_2 on the expected success rate for the next frame $\hat{s}[t+1]$. We assume a greedy user meaning a user that has always data to transmit, so that it can use any additional capacity provided to it. (TCP traffic is an example of greedy traffic.)

The upper threshold determines the sensitivity of the algorithm and it is specified through the constant ω which gives the thresholds relative position with respect to the average success rate and its standard deviation:

$$\theta_1 = \pi[t] - \omega\sqrt{\mu[t]} \qquad (13)$$

The lower threshold is determined in such a way that if the expected success rate in the next frame falls below the lower threshold and the user waits until the expected success rate reaches the upper threshold, then the user is expected to receive more service compared to not deferring transmission. This follows that the relationship between θ_1 and θ_2 can be obtained by applying eq. (12):

$$\theta_2 = \beta_u\left(\theta_1 - \frac{\frac{1}{\beta_u}-1}{\frac{1}{\beta_l\pi[t]}-1}\right). \qquad (14)$$

The rule is that we begin deferring transmission when the expected success rate falls below θ_2, and re-start when the expected success rate is above θ_1.

4.3 Adaptive threshold calculation

The previous subsection described how the thresholds θ_1 and θ_2 can be computed given ω, which determines sensitivity of the thresholds. A high value of ω results in lower thresholds and therefore less frequent relinquishing of service. A low value of ω on the other hand causes the user to relinquish service more often.

To reach the highest throughput and most efficient resource utilisation, a user should relinquish service as often as possible so that during transmission, the channel behaviour is as good as possible. However, the speed of compensation in the master-scheduler (determined by the parameter γ) poses an upper limit on the frequency of relinquishing service: the average rate of necessary compensation generated by the user should not exceed the rate of compensation that the scheduler can provide.

The optimal value of ω depends on the speed of compensation, γ, but also on the behaviour of other users, whether they request resources or not and how much compensation they receive. This is why we have chosen to implement an adaptive algorithm for setting ω based on feedback from the scheduler in the AP. This adaptive algorithm can be regarded as optional in the scheme: without it, a value for ω can be set as a constant.

The algorithm works as follows. In the broadcast channel of each frame, the AP gives a one-bit feedback to all the users which is set when the user has nonzero lag. Each user monitors this bit and computes an exponential moving average (denoted by f_i) for the ratio of frames where the user has nonzero lag at the scheduler. When f_i is too small this means that the scheduler can still provide more compensation. In this case the user decreases ω in orger to get more compensation for unused resources, so that user is active when the channel is better. When f_i is too large this means that there is a danger that the lag is too high and reach its upper limit, i.e. the scheduler can not provide the required amount of compensation. In this case ω is increased.

5 Simulation study

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 8 PDUs/frame where a PDU can carry 48 bytes, giving a total system capacity of 1.5Mbps. This capacity is shared by two users, each having an average PDU loss rate of 25%, but independent channel error models. User 1 has independently distributed losses, while User 2 has bursty losses according to an ON-OFF Markovian model. The average OFF period is 100 PDU transmission times, the loss rate is 95 % in the OFF period, 5% in the ON period. Both users are greedy.

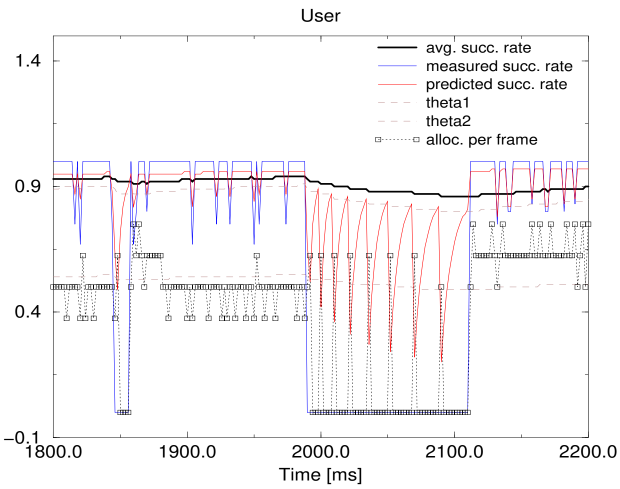

Fig. 1. User behaviour when the channel has bursty errors.

Figure 1 shows a trace from a simulation for User 2, having bursty losses. The figure plots the time evolution of a number of parameters computed each frame. The measured success rate is the ratio of the successful PDU deliveries to all PDU transmissions in the last frame. The average and predicted success rates are computed as described in Subsection 4.1. The thresholds θ_1 and θ_2 (Subsection 4.2) are also plotted. The user's decision and the behaviour of the scheduler is seen on the amount of allocation per frame. In the figure, this gives the number of PDUs allocated in the frame to the user as a fraction of the total number of PDUs per frame (8 in our setup). When the user decides to relinquish service in a frame, this is seen by the lack of allocation per frame.

The simulation illustrates the user behaviour and compensation (we used the parameters $\gamma = 0.5, \beta_l = 0.5, \beta_u = 0.8$ here). In the figure we can see two periods when the user relinquishes service, one at 1850ms, the other from 1980ms to 2120ms. The latter is a long OFF period in which the user re-starts several times. Note the increased amount of allocation after a user relinquished its service as a result of compensation for unused capacity in the scheduler.

Figure 2 shows the effect of ω on the throughput of connections and system utilisation. In this simulation there was full compensation for unused capacity ($\beta_u = 1$), no compensation for lost capacity ($\beta_l = 0$) and the speed of compensation $\gamma = 0.5$. The figure shows the following throughput quantities normalized by the system capacity: the total allocation for both connections (SumAlloc), the total goodput of the two connections (SumGoodput), the allocated capacity from the two connections (Alloc1 and Alloc2) and the goodput (successfully carried traffic) for the two connections. The graphs DeferRate1 and DeferRate2 show the fraction of frames where User 1 and 2 decided to relinquished transmission.

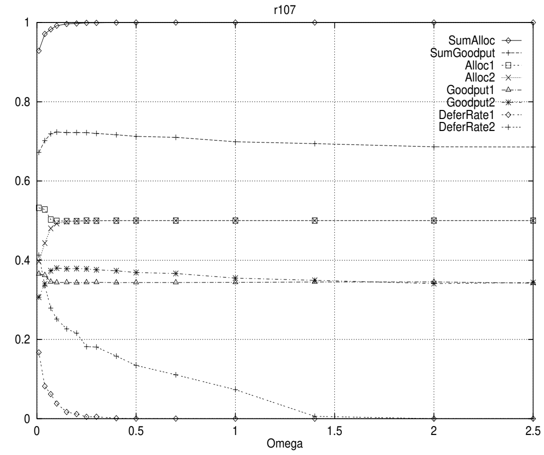

Fig. 2. Effect of ω, $\gamma = 0.5$, $\beta_u = 1$, $\beta_l = 0$.

The figure clearly shows a maximum for the total goodput and the goodput for User 2 (the one with bursty channel loss model) at near $\omega = 0.15$. User 1 can not achieve any improvement since its channel losses are independent. For higher ω, the gain of User 2 is less becuase the user relinquishes transmission less frequently, therefore it does not avoid all the bad channel states that it possibly could. This clearly shows the advantage of using the compensation method for unused capacity. For smaller ω, the user relinquishes more service that the scheduler can compensate. As a result, the allocation to User 2 is decreased, and this, of course, results in goodput decrease.

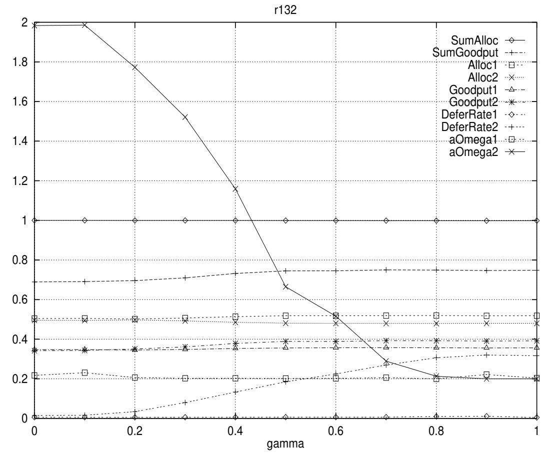

Fig. 3. Effect of γ, $\beta_u = 1$, $\beta_l = 0.5$.

Figure 2 illustrates that there exists an optimal value of ω motivating the use of the adaptive ω calculation approach described in Subsection 4.3.

Using that algorithm to set ω adaptively, Figure 3 shows the effect of changing the speed of compensation, γ, while $\beta_u = 1$ and $\beta_l = 0.5$. (The average value of ω for the users are shown by aOmega1 and aOmega2.) As the speed of compensation increases, there is time for more frequent relinquishing of service for User 2 (increase of deferRate2). This is achieved through the decrease of ω. Note that as γ increases from 0 to about 0.6, there is significant increase in the total system goodput as well as the goodput of User 2 and also User 1. The increase of goodput for User 2 is explained by its better utilisation of the allocated capacity, while the slight increase of goodput for User 1 is explained by the increase of allocation provided to it due to the compensation for lost capacity.

6 Conclusions

We have presented a scheme for the resource allocation of the capacity of a frame-based wireless base station with its performance evaluation. The new contributions of our work are (a) presenting a modular architecture where the fair master-scheduler does not explicitly take into consideration the channel state of individual users; (b) showing how a wireline fair scheduling algorithm can be simply extended to compensate for lost capacity after losses occur; (c) extending the master-scheduler with compensation for unused capacity, so user can optimize for its own when to relinquish service in the hope of more service later; (d) showing one possible way of user behaviour; and (e) showing by simulation that the pair of master scheduler and user decision rule can work together to improve both the total system utilization and the goodput of individual users.

References

1. 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.
2. HiperLAN/2 Global Forum. http://www.hiperlan2.com/.
3. György Miklós and Sándor Molnár. Fair allocation of elastic traffic for a wireless base station. In *IEEE GLOBECOM'99*, December 1999.
4. 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.
5. T. S. Eugene Ng, Ion Stoica, and Hui Zhang. Packet fair queueing algorithms for wireless networks with location-dependent errors. In *INFOCOM'98*.
6. Hui Zhang. Service disciplines for guaranteed performance service in packet-switching networks. *Proceedings of the IEEE*, 83(10), October 1995.