Scheduling as an Optimization Problem

Nilo Casimiro Ericsson
Signals and Systems
Uppsala University, PO Box 528
SE-751 20 Uppsala, Sweden
nce@signal.uu.se

ABSTRACT

This paper discusses some optimization algorithms intended for usage in the process of scheduling transmissions between a base station and mobile terminals by allocating time-slots to the different mobiles.

The purpose of the scheduling is to make use of the fast fading characteristics of the radio channel, instead of alleviating the effects with over-pessimistic channel coding. By using information about the individual data streams, together with information about future wireless channel characteristics for the different mobile hosts, it is possible to plan the transmission, so that the requirements meet the limitations.

The algorithms described are compared with respect to throughput, computational complexity, and user demand satisfaction.

INTRODUCTION

In future packet based wireless communication systems, the downlink will be primarily used for data transmission to mobile terminals. An obstacle in this context is the time-variability of the channel. To achieve a high system throughput also over fading channels, adaptive methods for the adjustment of the modulation alphabet and the coding complexity, together with time-slot scheduling, can be used [6].

In the scheduling approach, prediction of different user channels provide a basis for detailed scheduling of the transmission, by combining time-slot allocation and adaptive modulation. This approach can also take into account the desired error-probability and the priority associated with different users, as well as the current traffic situation. Moreover, the frequency band can be used efficiently, since the different users are allocated time-slots when their transmission conditions are predicted to be favorable, allowing them to use a high modulation level. The resulting constant and low (user-specified) error-rate provides the error correcting codes with manageable data, avoiding bandwidth consuming re-transmissions. The main drawback of scheduling in general, is the computational complexity, and the sensitivity to channel prediction errors [4].

In previous studies, a heuristic method for time-slot scheduling called the Robin Hood method, was used. It was chosen for its simplicity. No real analysis of its performance, nor any comparisons with other methods, have been done until now.

In this paper, the scheduling problem is isolated from the rest of the system, in order to investigate suitable algorithms for fast, and efficient, allocations of time-slots to different users.

In the next section it is stated that efficient scheduling is an important issue when discussing future mobile communication systems. The following section gives some additional system aspects, mostly motivating the necessity of including a buffer in the design. Thereafter the problem formulation for this paper is given, simulation results are presented, and some conclusions are drawn.

MOTIVATING SCHEDULING

Adaptivity is crucial in order to obtain spectral efficiency in future mobile communication systems. Real-time predictive scheduling is a possible way to achieve adaptivity.

Spectrum Efficiency

To obtain high data throughputs over wireless channels, we have to act cleverly. The channel quality varies substantially over time, due to radio interference and the mobility of the radio stations. Different types of fading result in a high probability for a bad radio connection some of the time. *Slow fading* can be counteracted by controlling radio transmitter power, or performing handover to another base station. The remedy against *fast fading*, however, is traditionally different types of channel coding and interleaving. In a simplified explanation, the channel coding adds extra information, or controlled redundancy, to the transmitted data, whereas the interleaver spreads the information over time, to make it more robust against error bursts which occur in the fading dips, where the signal-to-interfernce ratio is momentarily low. The channel coding, which is often over-pessimistic, generates much overhead to the wireless system, which in turn wastes precious bandwidth. The scheduling approach is substantially different.

Channel Prediction Works

A central component in the scheduling approach to spectral efficiency, is the channel predictor. It has been demonstrated in [2] that it is possible to predict the channel SINR variations quite accurately several milliseconds into the future. Having these predictions, one per radio link, they can be used together with a target error rate to assign modulation rates, for planning of the transmissions to the dif-

ferent mobile hosts, giving access to the users that have good predicted channel quality. Doing this, we increase our chances of getting the data across the wireless link without error and at a high rate, thus increasing the system throughput, and the spectrum efficiency.

On one hand, the longer the time-frame that the scheduler gets to schedule over, the more optimized the allocation. On the other hand, the further into the future we look, the harder it is to make a correct prediction of the channel quality. The performance of the predictor is crucial for the outcome from utilizing the scheduler. So, in a practical system, a trade-off between predictor performance and scheduling gain, has to be done. The performance for some different link-layer strategies, with unreliable channel predictions, are evaluated in [4].

Quality of Service

Another issue for future data communications over wireless, is quality of service (QoS). Although there is no agreed-upon definition of what QoS really is, it most certainly is improved by increased throughput and decreased delay. The bandwidth over the wireless channel is limited, and relatively narrow, so it has to be utilized efficiently. A way of doing so is to associate each data flow with an economical value, and to give priority to the higher valued flows. This value assignment is not trivial, but once it exists, it can be incorporated into the scheduling so that the scheduler tries to maximize the value of the transmission, and hopefully, the QoS.

So, by matching the higher protocol layer requirements with the physical constraints of the radio channel, we hope to find a time-slot allocation that efficiently utilizes the available spectrum, and maximizes the value of the transmission. Higher layer protocol aspects for Internet communications over error-prone channels are further discussed in [3].

SOME SYSTEM ASPECTS

Apart from the subsystem that makes sure that we have channel predictions for all the ongoing sessions, we need an input buffer, the pupose of which is two-fold: First, we need some mechanism to arrange incoming packets from the wired network in the right order, so that this task is removed from the light-weight mobile host. At the same time, the buffer works as a shock-absorber between the two parts of the network, so that performance variations in either side are hidden from each other. Second, we can use the buffer to estimate the amount of data that has to be transmitted to the different users. We also get the opportunity to analyze the contents of the data flows, so that different values can be assigned to the different flows.

The Buffering Subsystem

The buffer controller is assumed able to submit a status report to the scheduling subsystem, described in the following section, so that the scheduler can make an appropriate decision on which queues to choose for the next transmission frame.

The queues are emptied in a bit-by-bit manner, independently of the individual packet boundaries. The reason for this is to minimize the overhead by filling all the link-layer frames with data. The bit-stream is passed to the link-layer subsystem, along with information about the service requirements. The incoming buffer is described in Figure 1. At the receiving side of the wireless link, the packets have to be re-assembled, before passing them up to the network layer. This can be done since the scheduling decision is transmitted (broadcasted) to the receiving side, and it totally determines which byte belongs to which flow.

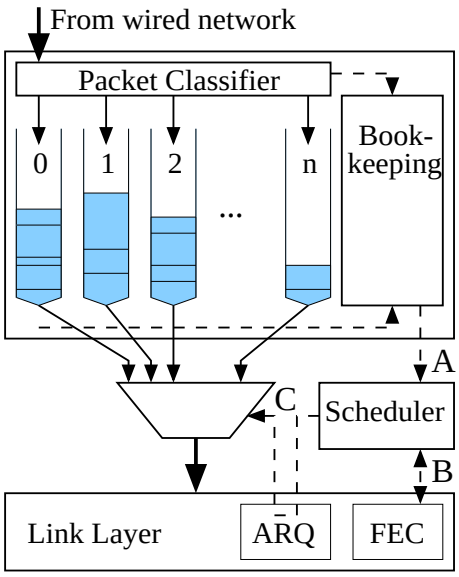

Figure 1: Schematic of the buffer and its queues, and how they interconnect to the scheduler and link layer. The packets arrive at the top and are inserted into their respective queues, restoring order among occasionally arriving out-of-order packets. The buffer regularly submits a status report (A) to the scheduler, containing info about the priorities, the size of the queues, and the required link service, some of which is also passed to the link layer (B). The scheduling decision (C) is updated by the link layer ARQ, and is then used to drain the queues.

The Scheduling Subsystem

The scheduler creates a signaling pipe [8] between the network layer buffer and the link layer service, making them mutually aware of one-another. For instance, the network layer does not ask for a link service whenever there is data to transmit. Instead it notifies the scheduler of the incoming traffic by passing queueing information (A in Figure 1) about the amount of data and type of service that would be preferred by the packets. The scheduler then asks the link layer for a report (B in Figure 1) about how the channel conditions would meet the required service. This can be done since the link layer has access to channel prediction data of all the established connections.

PROBLEM FORMULATION

The problem discussed and hopefully solved in this paper deals with methods for (sub)optimally allocating time-slots to users. The allocation is based on the users' requirements and their predicted wireless channel conditions. The channel conditions are translated, via the target bit error rate (BER), to an allowed modulation format. This translation results in an array of size $U \times S$, where U is the number of active users, and S the number of time-slots in the scheduling window. Each entry in the matrix is the allowed modulation format ($R = log_2 M$, where $M = 2$ for BPSK, $M = 4$ for QPSK, i.e. the number of bits per symbol) to meet the target error rate for a predicted channel quality in each time-slot. This array will hereafter be referred to as the *constraint matrix*.

From the other side of the scheduler, the throughput requirements for each user are reported in a vector of size $1 \times U$ (along with some kind of priority for each data flow). This vector will be referred to as the *requirement vector*. Now, the task for the scheduler is to make an allocation of time-slots to the users, so that as many as possible experience a good service. The throughput requirements are represented by a number for each user. These numbers are calculated from the amount of data currently in the input buffer, normalized by the time-slot size used in the radio link layer. So, for instance, if a user has a throughput requirement of 12, it means that it would be satisfied by 4 time-slots with uncoded 8-PSK modulation ($R = 3$).

The output from the scheduler is a vector with one entry for each time-slot ($1 \times S$), where each entry is the user number for the user that gets to transmit in a particular time-slot. This array is hereafter referred to as the *decision vector*. So, the problem is to generate a good-enough decision vector, from the given constraint matrix, and the requirement vector.

The decision vector can also be translated into a binary matrix of the same dimensions as the constraint matrix ($U \times S$), having one "1" for each time-slot. The "1" is in the location of the user that was allocated that time-slot, and the "0"s are in the other locations. This matrix is the *allocation matrix*.

Discrete Optimization

The set of solutions to, and constraints on the scheduling problem are made up of discrete values in a finite space. This means that all solutions could be found and classified by an exhaustive search, and the best one chosen. A disadvantage, however, is that the optimization problem becomes very complex to solve in an efficient way.

Cost Functions

The cost functions we use in our optimization should somehow reflect our goal with the scheduling. The most important goal is user satisfaction. (In this study, such quantities as *revenue maximization* are only dealt with indirectly, since they would require some pricing policy, and quantization of future goodwill, etc.) If all users are satisfied with the received service, we could assume that they happily pay for it. One way of quantizing the momentary user satisfaction is by evaluating the difference vector between the requirement vector and the resulting throughput from a scheduling decision vector. A negative value would reflect that the user did not get enough bandwidth to transmit all his data. A positive value means that some of the channel bandwidth will be wasted by letting time-slots travel without data. So, an optimal allocation in this sense would result in a difference of zero.

User satisfaction is not only a question of throughput, but also of delay. Delay requirements can be incorporated into the cost function by introduction of priorities. Low-delay applications are associated with a high priority, and latency-insensitive data is given a low priority. Priorities may also change with time, reflecting the increasing urge of transmitting a pending real-time packet.

A nice cost function that includes all the considerations mentioned above, is the weighted squared norm of the difference vector, expressed as

$$J = \|a - r\|_P^2 = \sum_u P_u (a_u - r_u)^2 \qquad (1)$$

where a_u is the allocated bandwidth to user u, and r_u is its required bandwidth, according to the buffer status report (see Figure 1). The weight P_u is a monotonically increasing function of the priorities for the different users, that might even be time-varying.

A different way of regarding this is from the point of view of system throughput. We would like to maximize the system throughput, with the (somewhat "soft") constraints of also keeping as many as possible of the users satisfied with their received service. This viewpoint results in a more difficult problem to solve, suggesting linear programming solutions, since throughput is a linear function of the allocation matrix. The difficulty is in the inclusion of the constraints: The constraints are "soft" in the sense that they need not necessarily be fully met. However, a linear program can only take "hard" constraints into account, leaving us with the option of starting with a solution to an unconstrained maximization and introduce the constraints sequentially, in a narrowing fashion. This approach has been avoided, due to the unattractiveness of the problem formulation.

SCHEDULING ALGORITHMS

A number of different methods for time-slot allocation have been implemented for comparison with the suggested Robin Hood algorithm. Scheduling is supposed to be kept simple, but still make a substantial contribution to improving the over-all performance of the communications system.

Optimal Allocation by Exhaustive Search

This is not a viable solution to the scheduling problem, since the optimal allocation performs a search through all combinations of time-slot allocations, with no clever search algorithm. This is just intended for comparison with the other algorithms, to show what can actually be obtained by a "perfect" optimization.

What this search algorithm does, is simply to systematically run through all possible combinations of the decision vector, saving the "best-this-far" vector.

Lagrange Formulation

This solution to the scheduling problem is based on equation (1), realizing that the constraint on the allocation vector being binary, can be introduced using Lagrangian multipliers [5]. The problem then boils down, through calculus of variation [1], to solving a system of $2US$ non-linear (Uth order polynomial) equations, U being the number of active users, and S the number of time-slots in the scheduling window. The solution could be found numerically using e.g. the Newton-Raphson algorithm [7], but a more clever algorithm is sought, since the problem has an attractive structure that could be exploited. The problem might even perhaps be formulated so that a closed form solution can be found. This would however require a different formulation from the one outlined here, due to the high order of the constraint equations.

Controlled Steepest Descent

This could be regarded as a first attempt at trying to find the optimal solution to the Lagrangian formulation of the problem. We here make use of the cost function (1), but instead of introducing the binary allocation vector as a constraint, we consciously restrict ourselves to the binary feasible solution space in our steepest-descent path. A good initial guess is needed, so first, each time-slot is allocated to the user that has the highest throughput in that time-slot, a simple throughput maximization without constraints. Second, for each time-slot and for each user, the change in over-all user satisfaction (1) is calculated in the case that the time-slot is given to that particular user. After all the possible transactions have been evaluated, the one giving the highest increase in over-all user satisfaction (or equivalently, the biggest decrease in over-all user pain) is executed.

A drawback of this approach is the number of sum-of-squares evaluations needed in order to take the appropriate next step. Simplifications could include e.g. taking more than one step at each iteration, or a simpler update of the cost function.

Robin Hood

This is a simplification of the Controlled Steepest Descent algorithm outlined in the previous section, but without the quadratic cost function. Again, the scheduler perfoms the scheduling in two rounds. In the first round, each time-slot is simply allocated to the user that can transmit at the highest rate in that time slot (unconstrained maximization). In the second round, time-slots are redistributed from users that have been over-supplied (rich), to users that have been under-supplied (poor), with respect to their required throughput. We call this equalization to user satisfaction the Robin Hood principle: To take from the rich, and give to the poor. This algorithm works as follows:

1. Find the rich and poor users by comparing their allocations to their amount of data in the queues
2. Loop until either no more rich or no more poor users exist:
   (a) For the richest user, find its worst time-slot, in the meaning of lowest transmission rate
   (b) Among the poor users, find the best user in that time-slot, and give the time-slot to him. In case two poor users have the same transmission rate, choose the one with the higher priority
   (c) Update the rich and poor variables

A crucial requirement for the algorithm to converge is that never must any rich users become poor, or vice versa. This is realized by having a gap between the rich and poor domains, too big to be crossed by one re-distribution step.

Best First

This approach, which is the simplest of them all, simply goes through the time-slots in chronological order, giving them one by one to the user that has the highest throughput in that time-slot, provided the user is under-supplied. In case the user is satisfied, the time-slot goes to the user with the second-best throughput, and so on.

This approach can also be regarded as an *nonpredictive scheduling*, only taking present conditions into account by not looking at future possible allocations when assigning a time-slot to a user.

SIMULATION RESULTS

In this section, the performances of three different approaches are presented. Three measures are compared, namely, the *throughput*, the *user satisfaction*, and the *computing complexity*. The size of the scheduling problem in these simulations is 9 users, competing for 48 time-slots. The traffic load is adjusted to be a little more than can be accomodated in the system, to make the scheduling problem interesting. The number of possible solutions to this problem is $9^{48} \approx 6.4 \cdot 10^{45}$, which clearly excludes the possibility for an exhaustive search, especially since we make 1000 runs with independent channel properties to get some statistics. In the simulations, the channel only affects the value of the possible modulation format for a given target error rate. There are no channel transmission simulations involved. The throughput performance in Figure 2, only reflects the *allocated* bandwidth.

The results are presented in Figures 2 through 4 by means of histograms, displaying the distribution of the simulation outputs.

In Figure 2 we see the throughput resulting from (top to bottom) Best First, Robin Hood, and Controlled Steepest Descent, and in Figure 3 the resulting *differences* in user satisfaction, named *unfairness*:

$$\begin{aligned} U &= d_{max} - d_{min} \\ &= \arg\max_u \|d_u\|_P^2 - \arg\min_u \|d_u\|_P^2 \end{aligned} \quad (2)$$

where $d_u = a_u - r_u$, see (1). There is, as can be seen from the histograms and the average values, a trade-off between fairness and throughput.

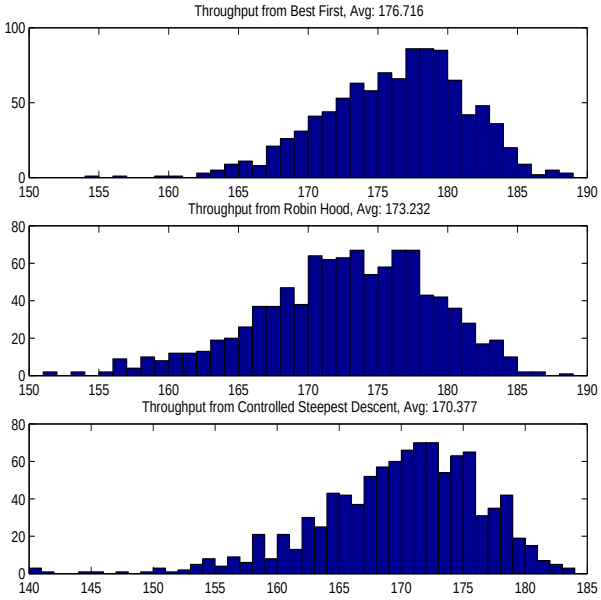

Figure 2: Resulting throughput for different optimization approaches. The y-axis shows the number of schedules resulting in the throughput on the x-axis. The more schedules on a higher x-value, the better.

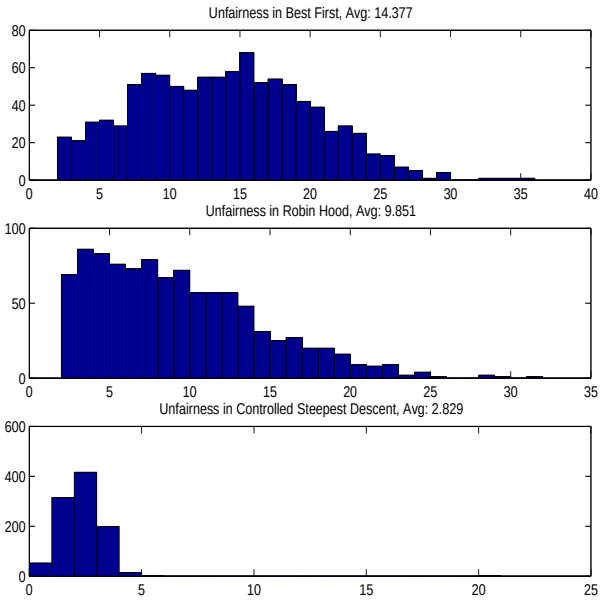

Figure 3: Unfairness among users for different optimization approaches. The y-axis shows the number of schedules resulting in the unfairness on the x-axis. The more schedules on a lower x-value, the better.

The last figure requires some more explanation, since the computation complexity is not the same for a step of the Controlled Steepest Descent as for a step of the other two algorithms. For each step in the Steepest Descent algorithm, 9×48 sums of 48 squares are calculated, whereas in the other two algorithms, merely a maximum value in a 9-element vector is found for each iteration. So, in comparison, this version of the Controlled Steepest Descent is prohibitedly complex, but it is also extremely fair, and will nicely include the priorities of different data flows.

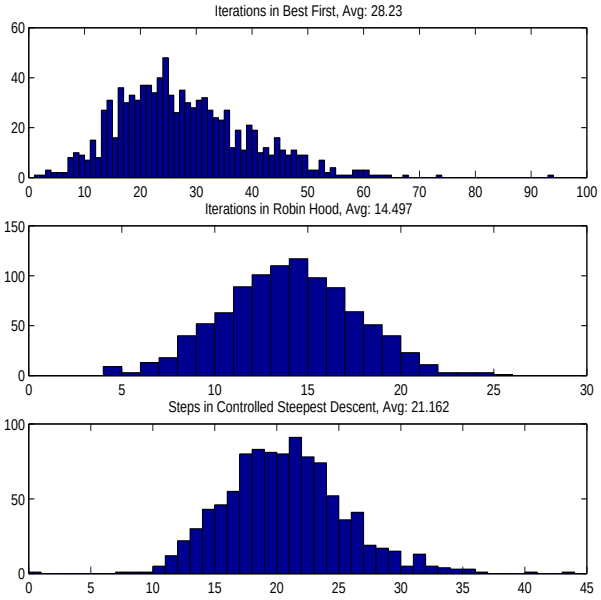

Figure 4: Required iterations for different optimization approaches. The y-axis shows the number of schedules running over the number of iterations on the x-axis. The more schedules on a lower x-value, the better.

CONCLUSIONS AND FUTURE WORK

It is interesting to search for possible closed form solutions to the quadratic problem, and this is under current investigation. Until then, we have some approximate iterative methods of varying complexity to take on the the task of schedule optimization. A trade-off between fairness of the allocation and system throughput has to be made. Moreover, fairness provision seems to require more computation than throughput maximization, due to the quadratic nature of the fairness optimization.

References

[1] A. E. Bryson and Y.-C. Ho. *Applied Optimal Control*. Hemisphere Publishing Company, 1975.

[2] T. Ekman. *Prediction of Mobile Radio Channels*. Licentiate Thesis, Uppsala University, Dec. 2000.

[3] A. Ewerlid. Reliable Communication over Wireless Links. In *Proc. PCC Workshop*, Nynäshamn, Sweden, Apr. 2001.

[4] S. Falahati and N. C. Ericsson. Hybrid type-II ARQ/AMS and Scheuling using Channel Prediction for Downlink Packet Transmission on Fading Channels. In *Proc. PCC Workshop*, Nynäshamn, Sweden, Apr. 2001.

[5] D. G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley Publishing Company, 1984.

[6] M. Naijoh, S. Sampei, N. Morinaga, and Y. Kamio. ARQ Schemes with Adaptive Modulation/TDMA/TDD Systems for Wireless Multimedia Communication Services. In *Proc. IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, volume 2, pages 709–713, Helsinki, Finland, Sept. 1997.

[7] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes - The Art of Scientific Computing*. Cambridge University Press, 1989.

[8] G. Wu, Y. Bai, J. Lai, and A. Ogielski. Interactions between TCP and RLP in Wireless Internet. In *GLOBECOM*, pages 661–666, Rio de Janeiro, Brazil, December 1999.