Fairness and Aggregation: A Primal Decomposition Study *

André Girard[1], Catherine Rosenberg[2], and Mohammed Khemiri[3]

[1] INRS-Télécommunications Place Bonaventure, 900 de la Gauchetière Ouest, Niveau C, C.P. 644, Montréal (Qué) Canada H5A 1C6 andre@inrs-telecom.uquebec.ca
[2] Department of Electrical and Computer Engineering, 1285 Electrical Engineering Building, Purdue University, West Lafayette IN 47907 - 1285 USA
cath@ecn.purdue.edu
[3] Ecole Polytechnique, Paris, France

Abstract. We examine the fair allocation of capacity to a large population of best-effort connections in a typical multiple access communication system supporting some bandwidth on demand processes. Because of stringent limitations on the signalling overhead and time available to transmit and process information, it is not possible to solve the allocation globally to obtain the optimally fair allocation vector. A two-level procedure is proposed where connections are aggregated within terminals, which send aggregated requests to the controller. The controller then computes the appropriate aggregated allocation per terminal and sends the corresponding allocation vector back to the terminals. Each terminal then computes the allocation vector for its connections. We want to study what aggregated information the terminals should send and what allocation problem should the controller and the terminals solve to produce a near-optimal (in terms of fairness) allocation vector. We propose a primal-based decomposition approach, examine in detail a number of approximations and show that by transmitting its total demand and number of connections, each terminal can achieve a near-optimal and near-fair allocation of capacity.

1 Introduction

The fair allocation of capacity among best-effort users is a subject that is gaining wide interest both in wire-line networks, such as ABR services in ATM [1, 2], non-QoS constrained services in IP networks [3] and in wireless and wireline access and satellite systems. In this paper, we focus on systems with multiple access links for which a process is needed to share the available capacity on a demand basis. All these systems are characterized by a set of users, each submitting a request for bandwidth, the sum of which usually exceeds the available total bandwidth. The required bandwidths are not absolutely needed and the users are willing to live with an allocation smaller than what they requested. The

* This work was performed while the two last authors were at Nortel Networks, Harlow, UK.

problem is then to allocate the total available bandwidth in an optimally fair manner. In all that follows, the measure of optimality always refer to the fairness of the allocation process unless otherwise noted.

Although the fair allocation problem is relatively simple to solve to optimality, in practice, there are a number of situations where the actual computation turns out to be difficult. Consider for instance the case of a large population of best-effort connections on a satellite system having typically many thousand terminals, each with potentially several connections. Because the bandwidth of the multiple access uplink is very limited, there is a need to limit the signalling overhead, i.e., the amount of information sent to request bandwidth. Also, the bandwidth allocation process is performed periodically every few tens of milliseconds for geo-stationary satellite systems and the allocation process should be able to run within this time limit. One of the main issues is that the problem of sharing fairly a given capacity among a large number of un-coordinated users competing for this capacity is very time consuming and there will be a scalability issue if we want to design a fast, low-overhead and optimally fair allocation process.

These systems, however, have a hierarchical structure where a terminal will manage a number of connections. These terminals will have some processing power and are only able both to request bandwidth from the controller and to reallocate this bandwidth among their connections. With a structure like this, each terminal receiving (or computing) the requests from its own connections could transmit to the controller some aggregated information based on the individual requests. In such a system, two questions naturally come to mind and form the subject of this paper: 1) How should the individual call requests be aggregated by the terminals, i.e., what aggregated information should each terminal send on behalf of its connections? and 2) how good are these approximations with respect to the optimal allocation?

The main findings of this work is that if each terminal sends only the sum of the requests of each of its connections, the controller cannot allocate the available capacity in a near optimal fashion. We can get a much better solution if each terminal sends both the sum of the requests of each of its connections and the number of connections. Based on these requests, the controller can solve an optimization problem of lower complexity (as opposed to the one it would have to solve if each connection had sent its own request) and thus is able to perform the computation within the allowed time. We show that this process yields results very close to the optimal solution.

The paper is structured as follows. First, we state the model and propose a primal decomposition of the optimal allocation. We solve the sub-problems and state the master problem. Then, we investigate two types of approximations. The first set uses only the sum of the individual requests for each terminal while in the second set, each terminal is allowed to send an aggregated request made up of two terms, one of which being the above sum. Numerical results then show that sending the number of connections as the second term yields a nearly optimal solution and that this is probably the best trade-off that can be done

in terms of efficiency and signalling overhead. Conclusions and future extensions to the work follow.

2 Optimal Allocation Model

We want to compute the optimally fair allocation vector $\mathbf{x}$ by solving the problem

$$\max_{\mathbf{x}} \; Z = \prod_{i=1}^{n} x_i \tag{1}$$

$$\sum_{i=1}^{n} x_i = C \tag{2}$$

$$0 \leq x_i \leq d_i \tag{3}$$

where n is the total number of connections, C is the total available capacity, x_i is the allocation to connection i and d_i is the request made by connection i. Note that there may be more constraints on the terminals but that would not fundamentally impact the results of this study. The problem is interesting when $\sum_{i=1}^{n} d_i > C$ that is, when it is not possible to meet all the requests. In a satellite system, n will be of the order of several thousands and the problem has to be solved once every few tens of milli-seconds. At each period, the available capacity C will change due to the arrivals and releases of calls that are not best-effort (i.e., that are allocated some resource on a reservation or static basis) so that we do not expect the problems to be very similar from one period to the next. Hence proposals based on explicit knowledge of the demand for each connection in the system are not realistic in terms of signalling overhead as well as processing time and power.

3 A Primal Decomposition Method

The basic idea for reducing the complexity of the computation is to use a two-level allocation procedure. At the terminal level, the controller partitions the total capacity C among the terminals according to some rule, yielding an allocation C_i to terminal i. Once this allocation is made, each terminal allocates its C_i among its connections.

An important advantage of a primal decomposition method is that the solutions are always feasible with respect to the total capacity constraint (2). This is in sharp distinction with dual methods where this constraint is not met unless the multiplier has been exactly calculated. In the present case, because there is little time for iterations, it is expected that dual methods would not have time to converge and thus could yield poor solutions and this is why we concentrate on primal techniques.

3.1 The Primal Decomposition Model

We now give a precise definition of the primal decomposition model for problem (1–3). Suppose that we have allocated a capacity C_i to terminal i by some yet unspecified method. Once this allocation has been made, assume also that for each terminal i, we allocate C_i optimally among the n_i connections of this terminal. Let $d_{i,j}$ and $x_{i,j}(C_i)$ be the demand and the optimal capacity allocated to connection j of terminal i and define the value of the terminal i objective function as

$$P_i(C_i) = \prod_{j=1}^{n_i} x_{i,j}(C_i).$$

We write $P_i(C_i)$ and $x_{i,j}(C_i)$ because once the terminal allocation C_i is known, the allocation to the connections and the value of the objective function are completely defined. The optimal allocation problem can then be written as

$$\max_{C_i \geq 0} \; P = \prod_{i=1}^{m} P_i(C_i) \text{ subject to } \sum_{i=1}^{m} C_i \leq C \text{ and } C_i \leq D_i \tag{4}$$

where m is the total number of terminals and $D_i = \sum_j d_{i,j}$ is the sum of the requests for all the connection in terminal i. Each of the $P_i(C_i)$ is computed solving the following optimal allocation for terminal i

$$\max_{x_j} \; P_i(C_i) = \prod_{j=1}^{n_i} x_j \text{ subject to } \sum_{j=1}^{n_i} x_j \leq C_i \text{ and } 0 \leq x_j \leq d_j \tag{5}$$

where n_i is the number of connections of terminal i and vector x_j is the allocation vector for terminal i where we have dropped the terminal index i for simplicity. Note that problem (4) is entirely equivalent to problem (1–3) under the condition that the $P_i(C_i)$s are calculated by solving (5).

3.2 Solving the Sub-Problems

We examine the structure of the sub-problem (5) and propose a fast solution technique. Note that this problem has exactly the same structure as the original problem (1–3) but is of much smaller size since it deals only with the connections of a particular terminal i.

It is obvious that the objective function of (5) is monotone increasing in x_j. The maximum will most likely be on one of the vertices of the domain and the first order optimality conditions are not very useful for computing a solution. The hard part is rather to determine on *which* vertex lies the optimal solution.

Assume that for a given value of C_i, we have chosen a set $J_i(C_i)$ of constraints of the form $x_j \leq d_j$ to be saturated and let $k_i(C_i)$ be the number of such constraints. The optimal solution of the sub-problem for the given value C_i then becomes

$$x_j \begin{cases} = d_j & \text{if } j \in J_i(C_i) \\ < d_j & \text{otherwise.} \end{cases}$$

The choice of $J_i(C_i)$ is of course subject to the condition $\sum_j x_j \leq C_i$. If this condition is not met, we must choose another set of saturated constraints. We must then compute the values of the $x_j, j \notin J_i(C_i)$. We rewrite the problem (5) for this particular choice of saturated constraints

$$P(J_i) = \max \prod_{j \notin J_i(C_i)} x_j \text{ subject to } \sum_{j \notin J_i(C_i)} x_j = \overline{C}_i$$

where $\overline{C}_i = C_i - \sum_{j \in J_i(C_i)} d_j$ is the residual capacity not allocated to the saturated constraints and we write $P(J_i)$ to indicate that the value of the objective depends on the choice of the saturated constraints. Note also that since all the saturated constraints are in J, there are no bounds on the remaining variables. We can then use the first order optimality conditions to compute these variables and we find the optimal solution

$$x_j = \begin{cases} d_j & \text{if } j \in J_i(C_i) \\ \dfrac{\overline{C}_i}{n_i - k_i(C_i)} & \text{otherwise,} \end{cases}$$

in other words, the residual capacity is allocated equally among the unsaturated connections. An implicit condition is that this allocation does not exceed the bounds $d_j, j \notin J_i(C_i)$. If this is not the case, then the choice of $J_i(C_i)$ has to be modified. The optimal value of the objective function of the sub-problem can then be written

$$P(J_i) = \prod_{j \in J_i(C_i)} d_j \left[\frac{C_i - \sum_{j \in J_i(C_i)} d_j}{n_i - k_i(C_i)} \right]^{n_i - k_i(C_i)} . \tag{6}$$

The problem then reduces to choosing the set J_i that will give the largest value for $P(J_i)$. We can prove the following theorem (the proof is omitted because of lack of place):

Theorem 1. *The optimal solution of the sub-problem, for a given value of C_i, is obtained by saturating the largest possible number of constraints with the smallest possible values for the d_js.*

Once we have made this choice, the residual capacity is spread equally among the unsaturated connections.

We can propose a very simple algorithm for this particular allocation. Imagine that C is increasing from 0. As long as the smallest value of the d_js is not reached, we allocate C equally to all connections. When the smallest bound is reached, the corresponding value of x_j is set at this bound and we keep allocating the remaining capacity among the still unsaturated connections. This goes on until C has reached its real value.

3.3 Solving the Terminal Allocation Problem

Given the structure of the sub-problems and their optimal solutions, we now turn to the solution of the terminal allocation problem (4). From now on, we assume that the d_js are numbered by increasing value. The controller can calculate an optimal solution if it knows the value of the *functions* $P_i(C_i)$ for *all* terminals and *any* value C_i, which means that the computation and transmission times will be large. We now describe the form of these functions.

Using the logarithmic form of the problem, we write eq. (6) as

$$\log P_i(C_i) = \sum_{j \in J_i(C_i)} \log d_j + [n_i - k_i(C_i)] \left\{ \log \overline{C}_i - \log[n_i - k_i(C_i)] \right\}.$$

We know that $k_i(C_i)$ is a monotone increasing function of C_i. Assume that $C_i \approx 0$. In that case, no constraint can be saturated and we have $J_i(C_i) = \emptyset$, $k_i(C_i) = 0$ and

$$P_i(C_i) = \left(\frac{C_i}{n_i} \right)^{n_i}. \tag{7}$$

In other words, near 0, $\log P_i(C_i)$ is linear in $\log C_i$ with a positive slope n_i. This situation remains as long as C_i is not large enough to allocate a capacity of d_1 to all connections, that is, as long as $C_i \leq n_i d_1$. At that point, the first constraint $x_1 \leq d_1$ becomes saturated. If we keep increasing C_i, we get $\log P_i = \log d_1 + (n_i - 1)\left[\log(C_i - d_1) - \log(n_i - 1)\right]$ and this as long as $n_i d_1 < C_i \leq d_1 + (n_i - 1)d_2$. In that range, $\log P_i$ is a linear function of $\log(C_i - d_1)$ with a positive slope of $n_i - 1$.When $C_i \to \infty$, we can allocate the full demands to all connections and we have

$$\log P_i(C_i \approx \infty) = \sum_j \log d_j$$

and this is independent of C_i so that we call this the *uniform* approximation. We can write the general form for $\log P_i(C_i)$ if we first consider the points 0, $n_i d_1$, $d_1 + (n_i - 1)d_2$, $\ldots$, $\sum_{l=1}^{k} d_l + (n_i - k)d_k \ldots \sum_{l=1}^{n_i} d_l$ on the C_i axis. These points define a set of intervals $\mathcal{I}_k, k = 0 \ldots n_i - 1$ where the function takes a different form in each interval. More precisely, we have, for $k = 0 \ldots n_i - 1$, when $C_i \in \mathcal{I}_k$,

$$\log P_i(C_i) = \sum_{j=1}^{k} \log d_j + (n_i - k) \left[\log \left(C_i - \sum_{j=1}^{k} d_j \right) - \log(n_i - k) \right] \tag{8}$$

where a sum with its upper limit lower than its lower one is defined as 0. The function $\log P_i(C_i)$ is made up of logarithmic segments and is continuous and monotone increasing. We can then solve problem (4) with (8) as the objective function.

There are two difficulties with this exact model. First, given the form of the functions (8), the solution of the main allocation problem (4) is likely to be difficult since the objective function is nonlinear and not differentiable. The second

and more serious difficulty is that the exact calculation of the functions (8) requires the knowledge of *all* the requests for *each* connection of all the terminals (i.e., all the $d_{i,j}$). This is not really surprising since we insist on calculating a truly optimal solution and we should expect that we would need as much information as if we were solving directly problem (1–3). We know that this is not feasible in practice and we want to investigate methods that would require less information and also perhaps make for an easier allocation problem.

4 Approximations

We examine here a number of approximations to the exact allocation problem in order to limit the information to be sent by the terminals to the controller. We proceed as follows. We determine what information is available and we describe a simple allocation procedure that has already been suggested elsewhere or that seems natural. We then translate this procedure into an approximation of the $P_i(C_i)$ functions. In this way, we can immediately see whether an approximation has a chance of being reasonably accurate or not. In all cases, we assume that the value of D_i is available for all terminals. We first examine a simple allocation procedure based only on the knowledge of the D_is and then we extend the analysis to cases where another set of numbers is available, either the number of connections per terminal (the n_is) or the product of the requests $\prod_j d_{i,j}$.

4.1 Approximation with D_i only

The simplest scalable technique is a two-level algorithm based on summing the demands in each terminal and sending them to the controller that allocates fairly and optimally the available capacity among the terminals based on these aggregated demands. Once the terminal allocation is done and received by the terminals, each terminal allocates fairly its quota among its connections. The C_is are calculated by solving the optimal terminal allocation problem

$$\max_{C_i \geq 0} \; P = \prod_{i=1}^{m} C_i \text{ subject to } \; \sum_{i=1}^{m} C_i = C \text{ and } \; C_i \leq D_i. \tag{9}$$

This amounts to replacing the exact $P_i(C_i)$ function of eq. (4) by the linear approximation $P_i(C_i) = C_i$. Going back to eq. (7), we see that this is equivalent to assuming $n_i = 1$. This is hardly surprising since we have a single value to characterize terminal i.

Once this solution is obtained, each terminal i allocates its C_is by solving the sub-problem (5). We can also see on Fig. 1 the plot of approximation (9) with the exact solution and the upper bound. We have also tried another method in which each terminal computes its own estimate of a best value for the number of connections and then uses this to do the allocation. In that case, the terminal allocation problem becomes degenerate and numerical results have shown that this yields to a very poor approximation of $\log P_i(C_i)$. For these reasons, this

approach has not been pursued. It should be quite clear that the more or less obvious approximations involving only the values of D_i are not going to give very good results and that more information is needed.

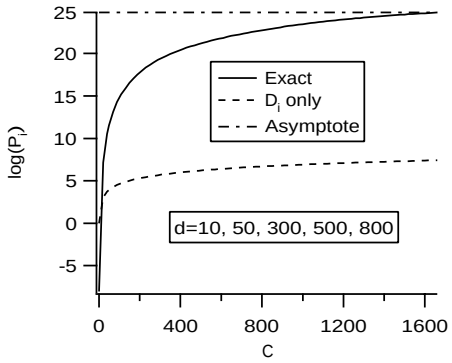

Fig. 1. Comparison of uniform approximation with the exact solution

4.2 Approximations with D_i and n_i

Given that the solutions computed from the models with only the D_is available can be relatively poor, we suggest that more information about the terminals should be used in the solution of the terminal allocation problem. In order to minimize the signalling overhead, we use only one additional value per terminal and the one that readily comes to mind is the n_is, the number of connections in terminal i. Hence each terminal i sends an aggregated request made up of two terms: D_i and n_i.

We consider first the simplest allocation technique that makes use of both the D_is and the n_is. Let $n = \sum_i n_i$ be the total number of connections. A straightforward method would be to allocate C/n to each connection, which assumes in effect that all connections have the same request. For this reason, we call this the *uniform* approximation. Each terminal that has a total request $D_i \leq C_i = n_i(C/n)$ gets its allocation D_i. This is subtracted from C, leaving a residual capacity C'. This in turn is allocated among the remaining terminals with the same rule, and so on until all the capacity has been allocated. Once the allocation C_i has been computed, each terminal solves its own allocation problem (5) as before.

The question is then what terminal allocation problem is being solved by this algorithm. Let $\mathcal{T}_i]$ be the set of connections that belong to terminal i. Rewrite problem (1–3) as:

$$\max Z = \prod_i \prod_j x_{i,j} \text{ subject to } \; x_{i,j} \leq d_{i,j} \text{ and } \; \sum_{i,j} x_{i,j} = C. \qquad (10)$$

The uniform allocation can be rewritten as

$$x_{i,j} = y_i \ \forall \ j \in \mathcal{T}_i \text{ and } \ d_{i,j} = \frac{C}{n} \ \forall \ (i,j)$$

$$\max Z = \prod_i (y_i)^{n_i} \text{ subject to } \ y_i \leq \frac{C}{n} \text{ and } \ \sum_i n_i y_i = C$$

and if we define $X_i = n_i y_i$, we get the equivalent form

$$\max Z = \prod_i \left(\frac{X_i}{n_i}\right)^{n_i} \text{ subject to } \ X_i \leq n_i \left(\frac{C}{n}\right) \text{ and } \ \sum_i X_i = C \tag{11}$$

and we see that in that case, we get the power allocation

$$P_i(C_i) = \left(\frac{C_i}{n_i}\right)^{n_i}. \tag{12}$$

This is the first segment of the real function as can be seen from Eq. (7).

4.3 Approximation with D_i and A_i

Although the n_i are an obvious choice to transmit in addition to the D_is, this is not the only possibility. Another quantity that is readily available is the product of the demands $A_i = \prod_{j=1}^{n_i} d_{i,j}$ which could be used to obtain a better allocation of the terminal capacities.

One way to use the value of A_i would be to assume that there are two connections for each terminal and to try to determine their requests $\overline{d}_1$ and $\overline{d}_2$ such that $\overline{d}_1 + \overline{d}_2 = D$ and $\log(\overline{d}_1 \overline{d}_2) = \log(A)$. Unfortunately, it is not difficult to find examples of request vectors $d_j, j = 1 \ldots n$ such that there do not exist two real values for $\overline{d}_1$ and $\overline{d}_2$.

Although approximation (12) seems to be reasonably accurate, we can see that it overestimates the exact solution at the boundary $C_i = D_i$. Letting $\tilde{P}_i(C_i)$ denote the value of approximation (12), we get at the boundary

$$\tilde{P}_i(D_i) = \left(\frac{D_i}{n_i}\right)^{n_i} \text{ and } P_i(D_i) = \prod_j d_{i,j}$$

which are different. We could hope for a better fit if we could use an approximation of the form (12) but with the requirement that it should go through the exact value of $P_i(D_i)$ at $C_i = D_i$. In order to do this, we need the value of $\log A_i = \sum_j \log d_j$. We want the terminal to determine and send an n^* such that

$$\log A = n^*(\log D - \log n^*). \tag{13}$$

This function is concave and can have 0, 1 or 2 solutions depending on whether D/e is larger (2 solutions), equal (1 solution) or smaller (no solution)

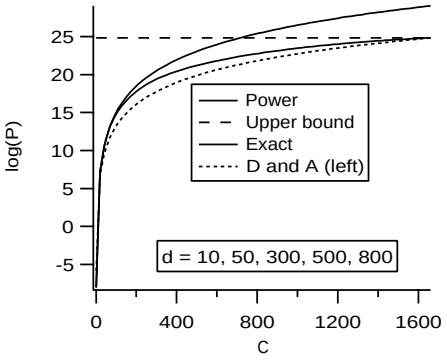

Fig. 2. Comparison of three approximations

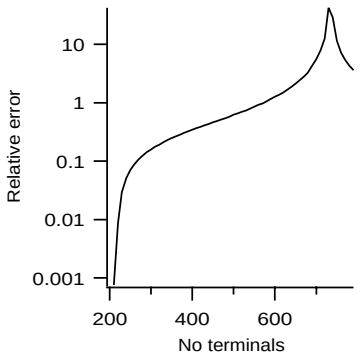

Fig. 3. Relative error with the uniform approximation and the D_is only

than $\log A$. In practice, it seems that there are generally two solutions and that the smaller one is much better that the larger one. A comparison of the four approximations is shown on Figure 2. The case corresponding to the larger solution of eq. (13) is not shown because it is not very good. We can see that the uniform approximation is an overestimate of the true function while the use of the product of the demands yields a lower bound.

5 Accuracy of the Approximations

The accuracy of the approximations can be checked by comparing with the optimal solution of the total problem. The values used to plot the figures are computed with 1000 random cases for each given number of terminals. Each case is obtained by generating an independent random value of n_i (number of connections per terminal i) and $d_{i,j}$ (the request of each connection j of terminal i) with $P(n_i = k) = 0.09$ for $1 \leq k \leq 10$ and equally distributed for the remaining values up to a value of 32 and $d_{i,j}$ equally distributed between 1 and 9. These connections are competing for an overall capacity equal to 5120 units corresponding to 32 time slots and 160 carriers in a (MF-TDMA) Multi Frequency Time Division Multiple Access uplink.

A first criterion that we use is the relative error Δ defined as $\Delta = |Z - Z^*|/|Z^*|$ where Z is the value of the objective function obtained with the approximation and Z^* is the optimal value of the objective function.

We can already see that the approximations of the $P_i(C_i)$ functions with D_i only have a very poor accuracy and this can be checked in Fig 3 where we show on a logarithmic scale the relative error as a function of the number of terminals. We can see that this relative error varies widely, in some cases being more than a factor 10. In other words, for some configurations, the fairness of the approximate allocation is more than 10 times poorer than the optimal allocation. The really interesting cases are the ones with an aggregate request based on two

terms, the D_is and either the n_is or the A_is. The first case is compared in Fig 4 and the second in Fig. 5 where we have plotted the relative error as a function of the number of terminals. In both cases, when there is a small number of terminals m, all demands can be met and all methods give identical results, which explains why the curves go to zero near the origin. Similarly, when m is large, it is impossible to satisfy any request and in that case also all the methods are equivalent. In the middle region, the peak of the error curve occurs when the approximate values coincide with Z^* and in this region, the approximation where we send the values of the n_is is much better than the one where we use the A_is.

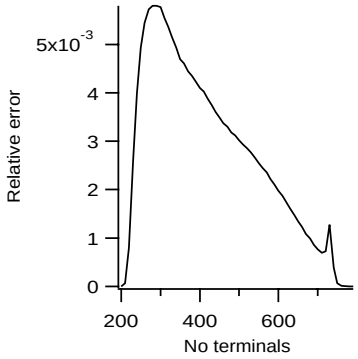

Fig. 4. Relative error of the approximation with the D_is and n_is

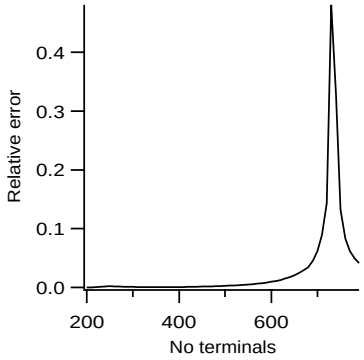

Fig. 5. Relative error of the approximation with the D_is and A_is

Another criterion to quantify numerically the unfairness of a scheme is the fairness index introduced by K.Jain in [4]. Suppose that an approximate solution allocates $\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n$ instead of the optimal allocation $x_1^*, x_2^*, \cdots, x_n^*$. Defining for each connection j the normalized allocations $x_j = \overline{x}_j / x_j^*$, the fairness index γ is then defined as $\gamma = \left(\sum_j x_j\right)^2 / \left(n \sum_j x_j^2\right)$

If the allocation given by the approximate solution is the same as the optimal one, then $\gamma = 1$, and the system is 100% fair. As the disparity increases, fairness decreases and a scheme which favors only a selected few connections has a fairness index near 0. The results shown on Fig. 6 indicate that for the approximation with D_i only, the scheme can be only 55% fair. In other words, this scheme favors 55% of the connections and discriminates the others. However, the scheme using D and n is almost 99% fair.

A third criterion has been proposed by L.Massoulié in [5] for file transfers. In that case, a legitimate bandwidth sharing objective would be to minimize the time needed to complete the transfers. This is done via a potential delay equal to the reciprocal of the rate allocation $1/x_j$ and by finding the allocations that minimizes the total potential delay $\sum_j 1/x_j$. In fact, we can show that the

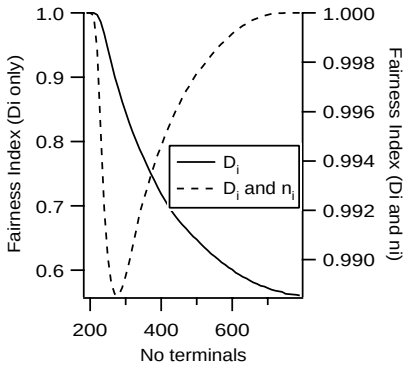

Fig. 6. Fairness measure of the approximations with D_i only and D_i and n_i

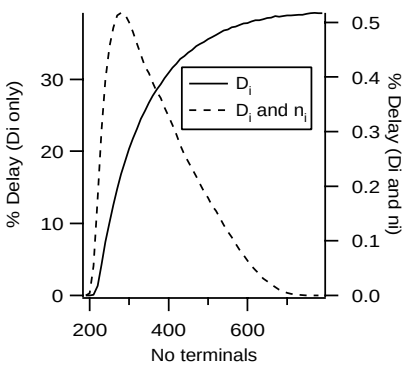

Fig. 7. Relative increase in potential delay

algorithm for the main problem (1–3) also minimizes the total potential delay. We can see on Fig. 7 that the approximation using D_i increases the optimal potential delay by 40% whereas its increase is less than 0.5% for the approximation with D and n.

The conclusion is quite clear that the best accuracy is obtained when each terminal transmits both D_i and n_i. Given the high accuracy of the approximation, we feel that it is not necessary to look for better approximations and that a two-level algorithm with these parameters is sufficient.

References

1. H. Yaiche, R. Mazumdar, and C. Rosenberg, "A game theoretic framework for rate allocation and charging of available bit rate (ABR) connections in ATM networks," in *Proc. Broadband'98*, P. Kühn, Ed., 1998.
2. A. Arulambalam and X.Q. Chen, "Allocating fair rates for available bit rate sevice in ATM networks," *IEEE Communications Magazine*, pp. 92–100, 1992.
3. A. Charny, D. Clark, and R. Jain, "Congestion control with explicit rate indication," in *Proceedings of the IEEE International Conference on Communications*. June 1995, IEEE.
4. R. Jain, D. Chiu, and W. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared systems," Tech. Rep. DEC TR-301, Digital Equipment Corp., Sept. 1984.
5. L. Massoulié and J. Roberts, "Bandwidth sharing and admission control for elastic traffic," in *Proc. INFOCOM'99*, Mar. 1999, pp. 1395–1403.