Fair and Efficient Dynamic Bandwidth Allocation for Multi-Application Networks

Ronaldo M. Salles [1]

Department of Systems Engineering
Military Institute of Engineering
22290-270, Rio de Janeiro, Brazil
Email: salles@ieee.org

and

Javier A. Barria

Department of Electrical and Electronic Engineering
Imperial College London
London, SW7 2BT, United Kingdom.
E-mail: j.barria@imperial.ac.uk

Abstract

The large diversity of applications and requirements posed to current network environments make the resource allocation problem difficult to work out. This paper proposes a dynamic algorithm based on weighted fair queueing (WFQ) to promote *fairness* (in the Rawlsian sense) and *efficiency* (in the Paretian sense) in the allocation of bandwidth for multi-application networks. Utility functions are used to characterize application requirements and provide the informational basis from where the algorithm operates. Aggregation techniques are employed to ensure scalability in the network core. Simulation results confirm a significant improvement of our approach over traditional bandwidth allocation algorithms: gains over 59% (62%) on minimum utility in relation to *maxmin* (*proportional*) fairness without compromising system utility. The algorithm also provides low errors (below 10% when compared to the zero-delay centralized approach) whenever response time does not exceed 1000 times the timescale involving flow arrivals and departures.

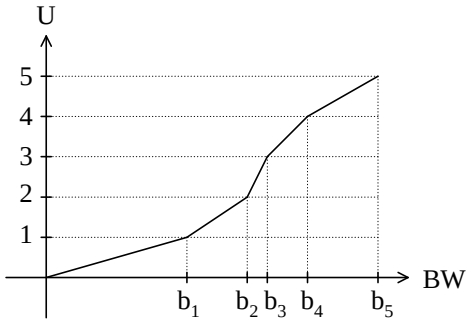
Key words: multi-application networks, fairness, utility functions, lexicographic criterion, bandwidth allocation

[1] This research was carried out during the author's stay at Imperial College London and was supported by CNPq Brazil, under Grant: 200049/99.

1 Introduction

Multi-application networks are increasingly predominant in today's telecommunication enterprises. The convergence trend towards IP technology has facilitated the deployment of environments where a wide variety of applications, ranging from highly adaptive to strict real-time, coexist and have their traffic transmitted over the same network infrastructure. In this case, as opposed to the homogeneous scenario, the amount of resources required by each type of application to perform well may differ substantially imposing an extra difficulty to the resource allocation problem. The concept of *utility function* [2] can be used to provide information about the amount of resources needed by each application and also to support the determination of an adequate solution for the bandwidth allocation problem.

In a multi-application network environment several different types of utility functions (concave, step, s-shaped, linear, etc.) are envisioned and have been qualitatively described in the literature [SHE95]. Quantitative studies on the determination of utility functions are also the subject of intensive research, methodologies to assess the quality of video and audio transmissions have long been addressed by the ITU [ITU97,ITU00a]. By employing subjective testing based on Mean-Opinion-Scores (MOS), alloted network resources can be directly associated with different levels of application performance (QoS). Fig.1 illustrates an example of piecewise linear utility function that is usually obtained from quantitative studies. In fact, quantitative utility functions



$$u(x) = k + \frac{x - b_k}{b_{k+1} - b_k}$$
$$b_k \leq x < b_{k+1}$$

Fig. 1. Piecewise Linear Utility Functions - MOS scale

can be determined for a large variety of applications: multimedia conferencing [ITU99], video streaming [LU03], VoIP [COL01], MPEG-4 transmissions [ZHA02], TCP flows [LIA00].

In this paper, we consider the QoS requirements imposed by the multi-application environment represented by general monotone utilities expressed as functions of the allotted bandwidth. Utility functions provide the informational basis

[2] In general terms, *utility functions* represent how each particular application perceives quality according to the amount of allocated resources.

from where our algorithm operates and decision are made. They also establish a common ground that allows the performance of different applications to be related and the optimal bandwidth allocation solution obtained. By focusing on utility (application performance) the algorithm improves the quality of service delivered to network users, which constitutes an important goal for service providers in a competitive market.

Besides the natural appeal to allocate resources in the most efficient way, it is equally important to avoid that an individual (or a group of individuals) is over-penalised when compared to others. Several works in the literature have brought the attention to this issue [CHE98,NAH98,MA99,FLO99,MIT99] and illustrate the lack of fairness in the allocation of network resources specially when multiple types of applications share the same network infrastructure. In [BON01] the authors proved that unfair allocations cause instability to the network and advocated that fairness should be considered as a design objective for the system.

The theory of social choice and welfare economics provides a broader view to the problem. Under this framework the bandwidth allocation problem in multi-application environments can be generally formulated as a *division problem* [SEN95]. From [DAS77,ROB80] a realm of different criteria are available, when utility functions are comparable and defined over the same scale, where the selection of a given criterion depends on the properties to be satisfied at the solution. To overcome the difficulties faced by previous allocation schemes we want the solution to be *fair* and *efficient*. In social choice theory, *fairness* is generally characterized by the egalitarian principle due to Rawls [RAW99]: "the system is no better-off than its worse-off individual", i.e. it implies the maximization of the benefit (utility) of the worse-off individual (max min$\{utility\}$). While *efficiency* in welfare economics is related to *Pareto Efficiency*: "an allocation is Pareto Efficient if no one can improve his benefit without decreasing the benefit of another" [SEN99]. Among several different criteria the *lexicographic criterion* emerged as the one that enjoys both *fairness* and *efficiency* as defined above. We use the *lexicographic criterion* to sort out the bandwidth allocation problem and propose an algorithm based on weighted fair queueing (WFQ) to enforce the solution.

In Section 2 we review some traditional work on bandwidth allocation and comment about the problems that might occur if such schemes are applied in multi-application environments. In Section 3 we formulate the bandwidth allocation problem under the lexicographic criterion. Utility aggregation techniques are presented in Section 4 in order to simplify the whole framework and allow the solution to scale according to the number of flows in the system. Section 5 proposes the dynamic bandwidth allocation algorithm based on weighted fair queueing updates. Simulation results are presented in Section 6 and finally in Section 7 the paper ends with the main conclusions.

2 Related Work

Traditionally, *maxmin fairness* [BER92] has been widely accepted as the fairness notion in computer networks, being used as a standard to perform bandwidth allocation among homogeneous connections [3]. However, it has been argued that it may generate sub-optimal allocations in several contexts depending on the actual *utility function* of the applications [CRO01].

Kelly in [KEL98] using microeconomics theory introduced the idea of *proportional fairness* where each application j is associated with a strictly concave utility function u_j (logarithmic) and allocations x are determined from the solution of $\max_x \sum_{j=1}^{N} u_j(x)$. It was further shown that the additive increase, multiplicative decrease congestion control in TCP tends to realize *proportional fairness* rather than *maxmin fairness*. Thus, *proportional fairness* fits well in homogeneous environments with *best-effort* applications.

However, as mentioned in the previous section, in a multi-application environment utility functions are not all concave and so *proportional fair* allocations are no more unique (several local solutions may exist) and may be completely unbalanced (some flows get zero utilities while others experience a very good performance). In fact, fairness embedded in *proportional fairness* depends on the concavity of the utility functions [Mo00]. The next example illustrates this idea and compares *maxmin* and *proportional fair* allocations with the *lexicographic* solution studied in this paper.

Example 1 *Consider the line network in Fig.2 where it is assumed that all links have same capacity c and each route has the same number of connections ($n_0 = n_1 = n_2 = n_3 = n$).For this network we have the following allocations,*

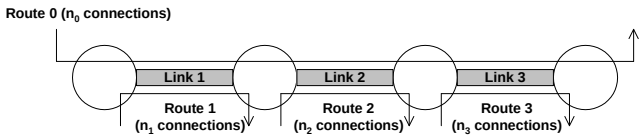
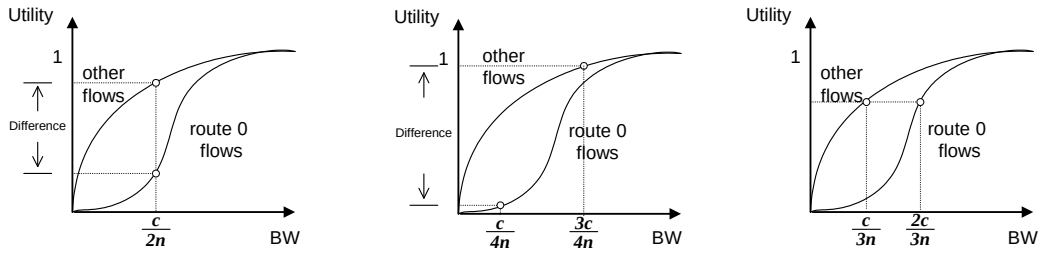


Fig. 2. Line Network

allocation of a route i connection:	x_i
max-min fair allocations:	$x_0 = x_1 = x_2 = x_3 = \frac{c}{2n}$
proportional fair allocations:	$x_0 = \frac{c}{4n} \quad x_1 = x_2 = x_3 = \frac{3c}{4n}$

Assume now connections on route 0 *described by S-shaped utility functions while all other connections described by strictly concave curves. The performance of* maxmin *and* proportional fair *allocations are indicated in Fig.3(a) and Fig.3(b). It can be observed from the figure a considerable difference on*

[3] ATM Forum - Traffic Management Specificaiton 4.0, April 1996



(a) *maxmin* (b) *proportional* (c) *lexicographic*

Fig. 3. Example of *maxmin*, *proportional fair*, and *lexicographic* allocations for the line network.

performance experienced by route 0 *flows when compared to the others. The figure shows that both schemes are quite unfair from an utilitarian perspective. The performance of route* 0 *connections under* proportional fairness *is indeed very poor, the applications are almost getting absolute no utility from the offered network service. At the same time, applications not using route* 0 *perform well and provide high satisfaction. If instead we apply the lexicographic criterion, from Fig.3(c) a much fairer allocation is obtained. This allocation conforms with the* fairness *and* efficiency *definitions adopted. Now all applications are satisfied and there is no discrimination according to the performance (utility) viewpoint.*

The example has shown that both *maxmin* and *proportional* fairness notions are restricted to the homogeneous environment and although they can be extended to the respective weighted versions, it is still necessary to determine the optimal weight settings for each network scenario. More discussion about this issue is left to Subsection 6.1 where the performance of different criteria are compared.

Several algorithms have been proposed to return *fair* solutions for bandwidth allocation problems. Regarding *maxmin fairness*, the main approach has been the development of asynchronous distributed procedures with explicit rate computations for individual sessions [CHA95,ARU96,ABR01]. Approximated solutions have also been proposed so that allocations could be returned in logarithmic time [AWE98]. Other approaches focus on distributed algorithms that can be implemented without the complexity of explicit rate calculations, it can be proved that *maxmin* allocations are obtained in the stationary regime if all links employ fair queueing policies [MAS02].

Regarding *proportional fairness*, distributed algorithms have been proposed using microeconomics theory and/or convex optimisation. The basic approach is to work with primal-dual decomposition so that only one price (Lagrange

multiplier) needs to be kept updated at each link for the solution of the primal problem [KEL98,ABR01,LOW99]. Such framework based on convex duality uses the fundamental assumption that utility functions are all concave. Again, in multi-application environments where utilities are of general shape the direct application of this approach is compromised.

In contrast, the approach proposed in this paper return fair allocations based on the lexicographic criterion (see Section 3). The proposed scheme does not follow the primal-dual approach since it does not restrict utility functions to be concave (but only strictly increasing), and also because the lexicographic criterion does not have a closed-form expression which is mild for common optimisation procedures.

Instead, the distributed algorithm proposed in this paper uses fair queueing policies. It differs from traditional *maxmin fairness* algorithms in several aspects. First, the algorithm considers utility information to perform the allocations, and thus it takes into account different QoS requirements of several applications. Second, the algorithm works at the aggregate level with piecewise linear utility functions in order to reduce complexity and allow the system to scale. Third, it is based on simple messages exchanges between nodes that while requiring some sort of synchronisation introduces very low overheads to the network.

3 The Lexicographic Criterion

First we introduce the notation to formalize the bandwidth allocation problem. The network is defined by a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{L}$ the set of links; each link $l \in \mathcal{L}$ has a capacity of $C_l > 0$; $\mathcal{N}$ represent the set of application flows sharing the network resources. Flows are routed through single paths according to a routing matrix A, if $a_{lj} = 1$ flow j crosses link l, $a_{lj} = 0$ otherwise. Note that any routing algorithm can be used as long as flows are not split over multiple paths. However, the approach can be also extended for this case whenever splitted flows are treated as a collection of individual sub-flows. The set of feasible allocations $\mathcal{X}$ (domain set) must satisfy all capacity constraints and non-negativity conditions,

$$\mathcal{X} = \Big\{ \sum_{j \in \mathcal{N}} a_{lj} x_j \leq C_l;\ x_j \geq 0;\ l \in \mathcal{L} \Big\} \tag{1}$$

According to [ROB80], the solution that satisfies the *lexicographic criterion*

(leximin) is given by:

$$\boldsymbol{x}^{lex} = \arg \max_{\boldsymbol{x} \in \mathcal{X}} \left\{ \lim_{\gamma \to \infty} \sum_{j \in \mathcal{N}} \frac{[u(\boldsymbol{x}, j)]^{1-\gamma}}{1-\gamma} \right\} \tag{2}$$

where $u(x_j, j)$ is the utility function of flow j. The limit in (2) makes the use of traditional optimization procedures not directly applicable to compute $\boldsymbol{x}^{lex}$. In fact, a better way to characterize this point would be in terms of the *objective set* (image of the domain set $\mathcal{X}$) and preference relations [MAS95].

Let $\mathcal{Z} \subseteq \Re^N$ defines the *objective set* if it contains all points $\boldsymbol{z} : (z_1, z_2, \ldots, z_N)^T$, where $z_j = u(\boldsymbol{x}, j)$, $\boldsymbol{x} \in \mathcal{X}$ and $j \in \mathcal{N}$. We want to characterize $\boldsymbol{z}^{lex} \in \mathcal{Z}$ so that its components are given by $u(\boldsymbol{x}^{lex}, j)$. Let π be a permutation for any $\boldsymbol{z} \in \mathcal{Z}$, such that $\pi(\boldsymbol{z}) = \boldsymbol{z}^p$ if,

$$z_j^p \geq z_{j-1}^p \tag{3}$$
$$z_j^p = z_k \tag{4}$$
$$j, k \in \mathcal{N} \tag{5}$$

Thus, $\pi(\boldsymbol{z})$ simply puts the components of $\boldsymbol{z}$ in a nondecreasing order.[4] We also define the preference operator $\succ$ over $\mathcal{Z}$, such that for any two feasible objective vectors $\boldsymbol{y}, \boldsymbol{z} \in \mathcal{Z}$ and permutations $\boldsymbol{y}^p = \pi(\boldsymbol{y})$, $\boldsymbol{z}^p = \pi(\boldsymbol{z})$, $\boldsymbol{y} \succ \boldsymbol{z}$ if $\exists m$, $1 \leq m \leq N$, satisfying:

$$y_j^p = z_j^p, \quad \text{for } \; j < m \tag{6}$$
$$y_m^p > z_m^p \tag{7}$$

Definition 1 *The lexicographic objective vector $\boldsymbol{z}^{lex}$ is the largest with respect to $\succ$, i.e. $\nexists \boldsymbol{z} \in \mathcal{Z}$ such that $\boldsymbol{z} \succ \boldsymbol{z}^{lex}$.*

From the domain set $\mathcal{X}$ and Definition 1 some important properties are readily available:

- (i) $\boldsymbol{x}^{lex}$ exists if $\mathcal{Z} \neq \varnothing$
- (ii) $\boldsymbol{z}^{lex}$ ($\boldsymbol{x}^{lex}$) is unique in the objective (domain) set
- (iii) $\boldsymbol{x}^{lex}$ is *fair* in the *Rawlsian* sense
- (iv) $\boldsymbol{x}^{lex}$ is *Pareto Efficient.*

In fact, the lexicographic criterion when applied to the bandwidth allocation problem can be viewed as a generalization over maxmin fairness to consider the utility viewpoint. The objective vector $\boldsymbol{z}^{lex}$ can be also characterized when

[4] For instance, if $\boldsymbol{z} = (0.7, 0.9, 0.3, 0.1)$ then $\pi(\boldsymbol{z}) = (0.1, 0.3, 0.7, 0.9)$

maxmin fairness is applied directly to the objective set $\mathcal{Z} \subseteq \Re^N$ representing an allocation of utilities. The lexicographic bandwidth allocation for each flow is then given by the inverse utility functions $x_j^{lex} = u^{-1}(z^{lex}, j)$. The next definition illustrates this generalisation.

Definition 2 *A feasible allocation $\boldsymbol{x} : (x_1, x_2, \ldots, x_J)$ is lexicographically optimal if and only if an increase of any utility within the domain of feasible allocations must be at the cost of a decrease of some already smaller utility. Formally, for any other feasible allocation $\boldsymbol{y}$, if $u(y, j) > u(x, j)$ then there must exist some k such that $u(x, k) \leq u(x, j)$ and $u(y, k) < u(x, k)$.*

Work along this direction can be found in [CAO99] where bottleneck theory [JAF81,BER92] was employed to define *utility bottleneck links* as follows.

Definition 3 *Given a feasible allocation vector $\boldsymbol{x} \in \mathcal{X}$, link l is the utility-bottleneck link of crossing flow $j \in \mathcal{N}$, if $\sum_{i \in \mathcal{N}} a_{li} x_i = C_l$ and $u(x_j, j) \geq u(x_i, i)$, $\forall i \in \mathcal{N}$ such that $a_{li} = 1$.*

Note that, if all utilities are given by the identity function $u(x, \cdot) = x$, Def. 2 reduces to the definition of *maxmin fairness* and Def. 3 to the definition of *botlleneck links*.

Two important properties follow directly from Def. 3 being stated below.

Property 1 *Any two flows i and j bottlenecked at link l achieve the same utility levels: $u(x_i, i) = u(x_j, j) = \bar{u}(l)$, or alternatively, link l provides utility $\bar{u}(l)$ for all flows bottlenecked locally.*

Property 2 *If a flow k crosses link l but is bottlenecked elsewhere in the network, then $u(x_k, k) < \bar{u}(l)$.*

The following theorem relates the lexicographic solution with *utility bottleneck links* and together with Definition 3, Property 1 and Property 2 provides all the necessary ingredients for the construction of an interative procedure to find the lexicographic solution.

Theorem 1 *A feasible allocation vector $\boldsymbol{x} \in \mathcal{X}$ leads to the lexicographic solution, if and only if each flow has an utility bottleneck link with respect to $\boldsymbol{x}$. **Proof** see the Appendix.*

In this way, it is possible to construct the lexicographic solution from a hierarchy of optimization problems similarly to what was done before for maxmin allocations [GAF84]. At each stage, an *utility bottleneck link* is determined along with the corresponding bottlenecked flows, then they are eliminated altogether from the network and a new reduced problem is solved. The procedure

is repeated iteratively until all flows get blocked, at this stage the lexicographic solution is achieved.

Let $\bar{u}(l)$ represents the utilities achieved by the flows crossing link l ($a_{lj} = 1$), from Definition 3, Property 1 and 2, the *global utility-bottleneck link* [5] can be obtained through the solution of,

$$\bar{u}(g) = \min_{l \in \mathcal{L}^*} \left\{ \bar{u}(l) \;\middle|\; \sum_j x_j = C_l;\ u(x_j, j) = \bar{u}(l);\ \forall j \in \mathcal{N},\ \text{s.t.}\ \ a_{lj} = 1 \right\} \quad (8)$$

where $\mathcal{L}^* \subseteq \mathcal{L}$ represents the used links only.

The solution of problem (8) determines the link g and also gives lexicographic allocations for the flows crossing g: $\forall j \in \mathcal{N}$ such that $a_{gj} = 1 \Rightarrow x_j = u^{-1}(\bar{u}(g), j)$. After that, if we eliminate link g from the network together with its flows, and by doing the corresponding updates on the sets $\mathcal{L}^*, \mathcal{N}$ and on the available capacities A_l of the remaining links, we generate a new independent and reduced problem. Equation (8) can be applied again to this new problem and the whole procedure repeated until all flows get blocked by their respective bottleneck links. At this point we have the lexicographic solution (Theorem 1), which has been obtained from the lowest to the highest utility components, and the allocation returned is unique for the initial set of flows $\mathcal{N}$.

The procedure just described is summarized below in Algorithm–1.

ALGORITHM–1: Lexicographic Allocations
1. compute g and $\bar{u}(g)$ from problem (8)
2. for each $j \in \mathcal{N}$ s.t. $a_{gj} = 1$,
  $x_j \leftarrow u^{-1}(\bar{u}(g), j)$
  save x_j
  for each $l \in \mathcal{L}$, s.t. $a_{lj} = 1$: $A_l \leftarrow A_l - x_j$ **end**
  $\mathcal{N} \leftarrow \mathcal{N} \setminus \{j\}$
 end
 $\mathcal{L}^* \leftarrow \mathcal{L}^* \setminus \{a\}$
3. if $\mathcal{N} = \varnothing$: **STOP**
 else goto 1.

The complexity of Algorithm–1 depends on the solution of the nonlinear problem in (8) and on the size of the set $\mathcal{N}$. The aggregation techniques studied next have a direct effect over these two aspects.

[5] First bottleneck link obtained in the hierarchy of optimization problems. Link g constraints the flows so that it provides the minimum utility in the whole network.

4 Piecewise Linear Utility Aggregation

The techniques discussed in this section are based on the assumption that utility functions are given in the piecewise linear form (as illustrated in Fig.1) and on the adoption of the lexicographic criterion to solve the network resource allocation problem. If the functions are not in the piecewise linear form they can be converted to it by choosing the number of piecewise linear intervals.

Fairness embedded in the lexicographic criterion guarantees that every commodity flow that shares the same network resources achieves the same utility. Therefore, individual flows that use the same path (share same resources) can be aggregated into a single path flow so that in the solution the utility of the aggregate along that path will directly reflect the utility of each individual flow. This constitutes the main idea behind the aggregation procedures discussed in this section, which allows the lexicographic solution to be obtained using aggregated information only. Individual allocations are then computed using the specific inverse utility functions of the individual flows. The advantage of using piecewise linear utility functions as it will be shown next is that this aggregation process is facilitated even when commodity flows have different utility functions [BIA00].

For a flow $j \in \mathcal{N}$ its piecewise linear utility function is expressed as follows,

$$u(x_j, j) = k + \frac{x_j - b_k^j}{b_{k+1}^j - b_k^j} \qquad b_k^j \leq x_j \leq b_{k+1}^j \qquad k = \{0, \ldots, K\}. \tag{9}$$

In fact, such functions are completely defined by a sequence of K intervals given by the points $(b_0^j, b_1^j, \ldots, b_K^j)$. A different set of points defines a new function. Let $\mathcal{N}(p)$ be a subset of $\mathcal{N}$ containing all flows that uses path p, since they are sharing the same network resources along the path, in the lexicographic solution they will achieve the same utility u_p: $\forall j \in \mathcal{N}(p) \Rightarrow u(x_j, j) = u_p$,

$$u_p = k + \frac{x_j - b_k^j}{b_{k+1}^j - b_k^j}; \quad b_k^j \leq x_j \leq b_{k+1}^j \tag{10}$$

$$x_j = (u_p - k)(b_{k+1}^j - b_k^j) + b_k^j; \quad b_k^j \leq x_j \leq b_{k+1}^j \tag{11}$$

The overall allocation on path p will be,

$$X_p = \sum_{j \in \mathcal{N}(p)} x_j \tag{12}$$

$$X_p = \sum_{j \in \mathcal{N}(p)} (u_p - k)(b_{k+1}^j - b_k^j) + \sum_{j \in \mathcal{N}(p)} b_k^j; \quad \sum_{j \in \mathcal{N}(p)} b_k^j \leq X_p \leq \sum_{j \in \mathcal{N}(p)} b_{k+1}^j \tag{13}$$

Hence, u_p can be also expressed by

$$u_p = k + \frac{X_p - \sum_{j\in\mathcal{N}(p)} b_k^j}{\sum_{j\in\mathcal{N}(p)} b_{k+1}^j - \sum_{j\in\mathcal{N}(p)} b_k^j}; \quad \sum_{j\in\mathcal{N}(p)} b_k^j \le X_p \le \sum_{j\in\mathcal{N}(p)} b_{k+1}^j \tag{14}$$

by making $B_k^p = \sum_{j\in\mathcal{N}(p)} b_k^j$ in (14) we have,

$$u_p = u(X_p, p) = k + \frac{X_p - B_k^p}{B_{k+1}^p - B_k^p}; \quad B_k^p \le X_p \le B_{k+1}^p \tag{15}$$

Equation (15) indicates that no matter how many individual flows are using path p, they can be aggregated into a single flow and utility function. After obtaining u_p from the lexicographic solution, individual allocations can be computed using (11). In this way, the size of $\mathcal{N}$ is limited to the number of paths in the network, if there is just one path per origin-destination pair: $|\mathcal{N}| = |\mathcal{V}||\mathcal{V}-1|$.

The only problem with the above aggregation approach is the large amount of information that should be collected in order compute the parameter B_k^p. In fact, for each individual flow j the vector $(b_0^j, b_1^j, \ldots, b_K^j)$ should be known somehow previously to the aggregation takes place. To overcome this difficulty a class-based approach is used instead, where a limited number of classes are available according to publicly known utility functions. Users select the class they want to use depending on the characteristics of their traffic. Thus, there is no need to know each individual parameter b_k^j but only the number of individuals per class per path. If $\mathcal{Q}$ is the set of classes,

$$B_k^p = \sum_{c\in\mathcal{Q}} n(c,p) b_k^c \tag{16}$$

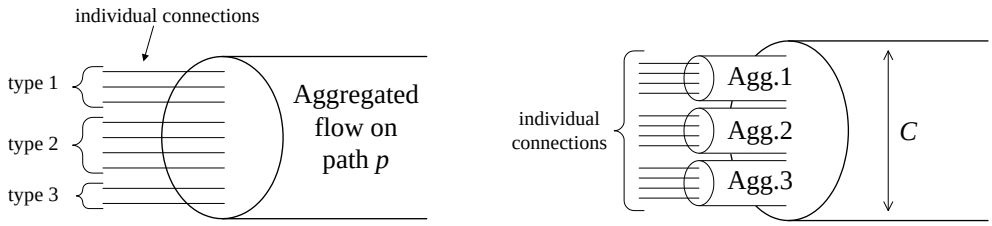
where $n(c,p)$ is the number of class-c flows on path p, and b_k^c's represent the parameters of the piecewise linear utility functions associated to class-c.

Fig.4(a) illustrates the aggregation procedure described so far. Assume there are nine individual flows using path p: three of class type 1, four of class type 2, and two of class type 3. Utility functions are assumed to be piecewise linear defined over 10 different segments ($K = 10$). In order to compute the lexicographic allocations without applying the aggregation technique, all the nine individual flows have to be considered in the bandwidth allocation problem as well as the 99 parameters [6] b_k^j defining their individual utility functions. However, using the aggregation technique it is only necessary to consider in the problem one flow over path p whose utility is described by the 11 parameters B_k^p below:

$$B_k^p = 3b_k^1 + 4b_k^2 + 2b_k^3 \qquad k = \{0, \ldots, 10\} \tag{17}$$

[6] b_k^j for $j \in \{1, \ldots, 9\}$ and $k \in \{0, \ldots, 10\}$

After the solution is determined the allocations for each one of the nine individual flows can be directly obtained from (11). With this approach the size of the problem is reduced since it was just considered one utility function per path.



(a) *path flow* (b) *link flow*

Fig. 4. Aggregated flows.

The same aggregation techniques can be used to further aggregate path flows into link flows as illustrated in Fig.4(b). For a given link l all path flows that are bottlenecked at l achieve the same utility. Assume the particular situation where all flows that cross link l happen to be bottlenecked locally at $\bar{u}(l)$. For this case, $\forall p$ such that $a_{lp} = 1 \Rightarrow u_p = \bar{u}(l)$. From (15),

$$\bar{u}(l) = u(X_p, p) = k + \frac{X_p - B_k^p}{B_{k+1}^p - B_k^p}; \quad B_k^p \leq X_p \leq B_{k+1}^p \tag{18}$$

and

$$X_p = (\bar{u}(l) - k)(B_{k+1}^p - B_k^p) + B_k^p; \quad B_k^p \leq X_p \leq B_{k+1}^p \tag{19}$$

All X_p sum up to the available bandwidth on link l, A_l ($= C_l$ for this case). Thus,

$$A_l = \sum_p a_{lp} X_p \tag{20}$$

$$A_l = \sum_p a_{lp}(\bar{u}(l) - k)(B_{k+1}^p - B_k^p) + \sum_p a_{lp} B_k^p \tag{21}$$

$$\sum_p a_{lp} B_k^p \leq A_l \leq \sum_p a_{lp} B_{k+1}^p \tag{22}$$

From (21) and (22):

$$\bar{u}(l) = k + \frac{A_l - \sum_p a_{lp} B_k^p}{\sum_p a_{lp} B_{k+1}^p - \sum_p a_{lp} B_k^p}; \quad \sum_p a_{lp} B_k^p \leq A_l \leq \sum_p a_{lp} B_{k+1}^p \tag{23}$$

Similarly as before, let $B_k^l = \sum_p a_{lp} B_k^p$, we have

$$\bar{u}(l) = k + \frac{A_l - B_k^l}{B_{k+1}^l - B_k^l}; \quad B_k^l \leq A_l \leq B_{k+1}^l \tag{24}$$

Note that equation (24) preserves the piecewise linear form of individual flow utilities as in (9). However, in this case individual flows have been aggregated into path flows and further into link flows, since B_k^l is the sum of parameters B_k^p for the paths that use l. Equation (24) may be viewed as the aggregated utility of flows bottleneck at link l, which was obtained in a two step approach: the first level of aggregation performed by source nodes to generate the path flows, while the second level performed at link l using path flow parameters B_k^p's.

It can be seen that the global utility bottleneck link can be now obtained from,

$$\bar{u}(g) = \min_{l \in \mathcal{L}^*} \left\{ k + \frac{A_l - B_k^l}{B_{k+1}^l - B_k^l};\ B_k^l \leq A_l \leq B_{k+1}^l \right\} \tag{25}$$

Given all the necessary information, B_k^l's and A_l, the complexity of problem (25) is $O(|\mathcal{L}^*| \log K)$ since it takes a maximum of $\log K$ operations to find the correct interval for each A_l (and so the value of k), and a maximum of $|\mathcal{L}^*|$ operations to compute the minimum $\bar{u}(l)$.

While the previous problem in (8) was defined for all individual path variables x_j, the equivalent problem in (25) uses only link variables A_l and link parameters B_k^l. Moreover, problem (8) requires the use of non-linear optimization techniques to return a solution, while from (25) the solution can be returned straightway in a maximum of $O(|\mathcal{L}^*| \log K)$ operations. Therefore, the aggregation techniques have contributed to simplify computations since they are based on simple summation of utility function parameters and reduce the problem to a link-flow formulation. Algorithm–1 as well as the upcoming algorithms directly benefit from such techniques.

5 Distributed Bandwidth Allocation for the Lexicographic Solution

5.1 The Proposed Algorithm

The bandwidth allocation algorithm presented in Section 3 is based on a centralized approach where the complete information of all flows should be available at a single point in order to compute the optimal allocations. Clearly it is not possible to meet this requirement in large scale and dynamic network scenarios, thus it is convenient to investigate possible distributed implementations.

Our starting point is the work of Massoulie and Roberts [MAS02] which fo-

cuses on distributed algorithms that can be implemented without the complexity of explicit rate calculations. From their framework it can be proved that if all links employ fair queueing scheduling policies, *maxmin* fair bandwidth allocations are reached on the stationary regime for all contending network flows.

As commented before, if we were interested in a homogeneous environment with $u(x_j, j) = x_j$ the lexicographic criterion would be reduced to *maxmin* fairness and the above work suffices to cover our goal. Nevertheless, for any linear utility function on the form: $u(x_j, j) = k_j x_j$, the lexicographic solution can be achieved using weighted fair queueing – WFQ with individual weights $w_j = 1/k_j$. The question is now what should be the weight settings for a heterogeneous environment where utility functions are given in the piecewise linear form.

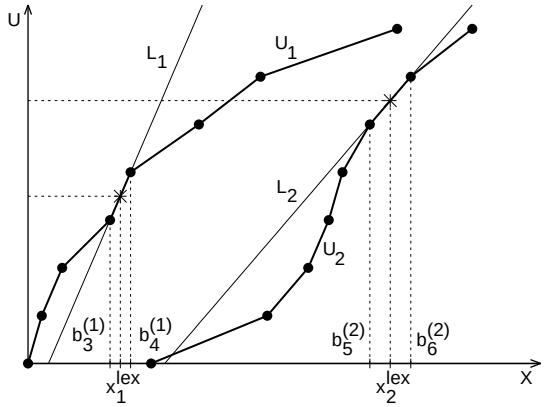


Fig. 5. Piecewise linear utilities and line equations

Fig.5 illustrates the approach for two piecewise linear utility functions U_1 and U_2. If we knew beforehand the line segments containing the lexicographic solution, which are represented in the figure by the forth (for U_1) and sixth (for U_2) segments, we could substitute the utility curves by the lines L_1 and L_2 and for this case, the lexicographic solution would be given by the same points x_1^{lex} and x_2^{lex}. Within those intervals,

$$\text{flow 1 allocation: } b_3^{(1)} \leq x_1 \leq b_4^{(1)} \qquad L_1 \text{ angular coefficient: } \frac{1}{b_4^{(1)} - b_3^{(1)}} \quad (26)$$

$$\text{flow 2 allocation: } b_5^{(2)} \leq x_2 \leq b_6^{(2)} \qquad L_2 \text{ angular coefficient: } \frac{1}{b_6^{(2)} - b_5^{(2)}} \quad (27)$$

and from Massoulie and Roberts's results, this solution can be obtained in the stationary regime if flows 1 and 2 are scheduled by WFQ policies along their routes. The WFQ weights should guarantee the lower bounds, $b_3^{(1)}$ and $b_5^{(2)}$, and that any spare capacity is shared according to the factors $b_4^{(1)} - b_3^{(1)}$ (allocation along line L_1) and $b_6^{(2)} - b_5^{(2)}$ (allocation along line L_2) in order to

achieve equal utility for the lexicographic solution.

The WFQ policy works at the packet level and guarantees to each flow i an overall bandwidth of [ZHA95]

$$\mathrm{BW}_i = \frac{w_i}{\sum_j w_j} C \tag{28}$$

where w_i is the weight associated to flow i and C the capacity of the link. Assuming $C > b_3^{(1)} + b_5^{(2)}$, from conditions (26) and (27) we must have:

$$\mathrm{BW}_1 = b_3^{(1)} + \mathrm{BW}_1^* \tag{29}$$

$$\mathrm{BW}_2 = b_5^{(2)} + \mathrm{BW}_2^* \tag{30}$$

$$\mathrm{BW}_1^* + \mathrm{BW}_2^* = C - (b_3^{(1)} + b_5^{(2)}) \tag{31}$$

$$\frac{\mathrm{BW}_1^*}{\mathrm{BW}_2^*} = \frac{b_4^{(1)} - b_3^{(1)}}{b_6^{(2)} - b_5^{(2)}} \tag{32}$$

Hence, the WFQ weights that satisfy (28)–(32) are given by:

$$w_1 = \frac{b_3^{(1)}}{C} + \frac{b_4^{(1)} - b_3^{(1)}}{b_4^{(1)} - b_3^{(1)} + b_6^{(2)} - b_5^{(2)}} \frac{(C - b_3^{(1)} - b_5^{(2)})}{C} \tag{33}$$

$$w_2 = \frac{b_5^{(2)}}{C} + \frac{b_6^{(2)} - b_5^{(2)}}{b_4^{(1)} - b_3^{(1)} + b_6^{(2)} - b_5^{(2)}} \frac{(C - b_3^{(1)} - b_5^{(2)})}{C} \tag{34}$$

In general, when there is $\mathcal{N}(l)$ flows sharing a link l, if $b_{k_i-1}^{(i)}$ and $b_{k_i}^{(i)}$ represent the limits of the piecewise linear segment containing the solution point for flow i, the necessary WFQ weights will be in the form

$$w_i = \frac{b_{k_i-1}^{(i)}}{C_l} + \frac{b_{k_i}^{(i)} - b_{k_i-1}^{(i)}}{\sum_{j\in\mathcal{N}(l)} b_{k_j}^{(j)} - \sum_{j\in\mathcal{N}(l)} b_{k_j-1}^{(j)}} \frac{C_l - \sum_{j\in\mathcal{N}(l)} b_{k_j-1}^{(j)}}{C_l} \tag{35}$$

where $\sum_{j\in\mathcal{N}(l)} w_j = 1$.

From (35) it can be seen that weights depend only on the determination of the piecewise intervals and on the capacity of the link. Therefore, if we are able to locate those intervals and inform the links about the corresponding utility parameters, they can set the correct WFQ weights to obtain the lexicographic solution. This idea will be explored next to build up a distributed algorithm for the bandwidth allocation problem. However, some important issues should be highlighted before presenting the algorithm

To avoid scalability problems due to the scheduling of a large number of individual flows in the core of the network, the algorithm works at the aggregate

level. Flows that use the same path are aggregated according to the procedures explained in Section 4 and illustrated in Fig.4(a). In this case the aggregate utility functions are given by (15) and the aggregate bandwidth parameters $B_{k_i}^{(i)}$ are used in (35) instead of the individual parameters. After final allocations are obtained for the aggregates, source nodes will be responsible to further share such allocations among individual flows inside the aggregates. In other words, the algorithm guarantees the solution for the aggregated flows, however once each aggregate receives its allocation it is up to source nodes to employ regulation procedures (scheduling, buffer management, etc.) to further share that allocation among the individual flows inside the aggregate [GUE99][DAS02]. These type of two level approach is a key issue in the solution of large-scale problems. For instance, such conceptual idea is the base of DiffServ networks [BLA98].

By working directly with WFQ weights we have a much more effective and stable procedure since network mechanisms can be set straight away to produce the optimal bandwidth allocation, and also there is no need to measure and constraint flows to secure the solution. Besides that, WFQ is a working conserving service discipline which means that there is no wasted resources and even when the weights are not yet set to their optimal values, all flows will continue to be served by the network. Furthermore, most of the routers and switches available in the market provide some form of WFQ implementation.

Another advantage of the approach is that the algorithm runs in the "control plane" without directly interfering or altering the flows in any way during operation. Thus, all network protocols are preserved and apart from control messages exchanged, no additional "data plane" procedure is necessary. When the algorithm terminates and the optimal weights for the lexicographic solution are obtained, each network node simply update the WFQ mechanisms associated with their outgoing links.

We split the algorithm into flow (Algorithm–2: Aggregate flow on path p) and link (Algorithm–2: Link-l) procedures in order to facilitate exposition despite both of them being executed by network nodes, the first procedure by the nodes which are source of the aggregate and the second by the nodes that control the transmission on the links. Fig.8 illustrates the messages exchanged by aggregate flows and links. The status field is responsible to provide the necessary order in which the lexicographic solution should be obtained. A value of zero indicates the flow is blocked by some saturated link (no more available bandwidth) along its path. Any other value indicates the piecewise linear utility segment in which the algorithm is operating – the current iteration.

Flows send to the links along their paths the corresponding parameters B_k^p obtained from (16). Note that B_k^p may be viewed as the bandwidth request of that flow along path p for interval k, and thus each link can allocate resources

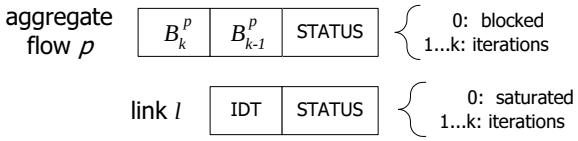
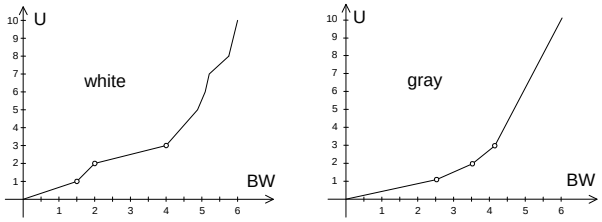


Fig. 6. Aggregate flow and link messages

taking into account this information. If links have enough resources to allocate to the flows the corresponding B_k^p, the solution will fall above B_k^p and the algorithm will continue to search in the upper intervals $k+1$, $k+2$ and so on. On the other hand, if a given link cannot provide the requested resources, it will act as a bottleneck for the crossing flows, whose lexicographic solution will fall inside the k-th interval.

Figs 7 and 8 illustrate the messages exchanged by the algorithm. Assume two flows, gray and white, share link L of capacity $c_L = 8$. In the first iteration both flows send their initial bandwidth requests (first piecewise interval) to link L. Since link L has enough capacity it sends back a status message for the next iteration. The flows continue to ask for more bandwidth following their respectives utility functions until iteration 3 (e), where the aggregate request is over C_L. At this time, link L is saturated and sends back a status message '0'. The flows once receiving this message know they are blocked and cannot ask for more bandwidth, link L can now set the corresponding WFQ weights to secure the allocation.



(a) *white flow* (b) *gray flow*

Fig. 7. Utility functions of white and gray flows.

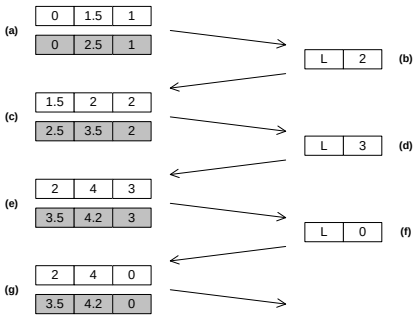


Fig. 8. Example of messages exchanged between flows and link

The algorithm continue until all flows get blocked and all intervals are determined, at this point the WFQ weights are set for each link according to (35). All computations are simple and the algorithm guarantees that information is exchanged in the proper order.

ALGORITHM–2: Agg. flow on path p

1. $k \leftarrow 1$
2. send $\{B^p_{k-1}, B^p_k, k\}$ to $\forall l \in p$
3. collect all link $l \in p$ broadcast
wait until all report next phase $(= k+1)$, or at least one report saturation ($phase\text{–}link = 0$)
4. if $\exists l \in p$ such that $status\text{–}link = 0$
send $\{B^p_{k-1}, B^p_k, 0\}$ to all $l \in p$
STOP
else $k \leftarrow k+1$
if $k > K$,
send $\{B^p_{K-1}, B^p_K, 0\}$ to all $l \in p$
STOP
end
goto 2.

ALGORITHM–2: Link-l

1. $k \leftarrow 1$, $\mathcal{B} \leftarrow \varnothing$
2. collect all msg. from $j \in \mathcal{N}(l) \setminus \mathcal{B}$
wait all report $status\text{–}flow$ 0 or k
3. for each $j \in \mathcal{N}(l) \setminus \mathcal{B}$
if $status\text{–}flow = 0$
save B^p_k and B^p_{k-1}
$R_b \leftarrow R_b + B^p_k$
$\mathcal{B} \leftarrow \mathcal{B} \cup \{j\}$
else $R_u \leftarrow R_u + B^p_k$
end
if $\mathcal{B} = \mathcal{N}(l)$
set WFQ as in Eq. (35)
STOP
4. if $R_b + R_u < C_l$,
$k \leftarrow k+1$
send $\{\text{IDT}, k\}$
$R_u \leftarrow 0$
else send $\{\text{IDT}, 0\}$
goto 2.

Algorithm–2 (Link-l) has to handle the additional complexity of flows that are already blocked (maybe by some other links) and flows that may be still not blocked at each iteration. The sets $\mathcal{N}(l)$ (set of flows using link-l) and $\mathcal{B}$ (set of blocked flows) as well as the variables R_b (bandwidth requests of blocked flows) and R_u (bandwidth requests of unblocked flows) are used for this purpose.

In Step **(3)** blocked flows have the bandwidth parameters corresponding to the last piecewise interval saved for future computation of the WFQ weights. Variable R_b is updated to account for the reserved link capacity for the already blocked flows, and the set $\mathcal{B}$ is also updated to include such flows. Note that when flows are blocked by an *utility bottleneck link*, the link computes $\bar{u}(l)$ from (18) and transmits this parameter to the crossing flows.

For the flows not already blocked, a temporary variable R_u is updated instead of R_b since their bandwidth limits are not yet fixed and may be changed in the next iteration. However, if all flows are blocked the algorithm set the WFQ weights and terminates. In case there are still unblocked flows, Step **(4)** verifies if link capacity is above the maximum bandwidth requirements.

The assumptions are that all exchanging messages are correctly received within

a finite time interval and that algorithms are restarted from time to time back to step **(1)** to track aggregate flow dynamics. Each node may start the algorithm using private clocks on common agreed periods, there are several works in the literature and practical implementations that can be applied to support this procedure [AZE94] [PAT94] [MIL96]. Note that such approach is only necessary to start the algorithms and do not need to have high precision since after the algorithm is started the waiting steps enforce the synchronization required to obtain the lexicographic solution.

5.2 Algorithm Analysis

The first important observation about the algorithm is its minimal overhead incurred in computations. The whole flow procedure (Algorithm–2 Agg. flow on path p) is basically composed by communication steps, while the link procedure just requires few summations and variable updates. The messages also pose very little overhead to network traffic since they consist of a maximum of three numbers: B_k^p (real), B_{k-1}^p (real), and status (integer). Since control messages are very small and should be transmitted with priority through the network, it is expected that the greatest time components incurred in control message transmission are given by propagation delays on links.

Proposition 1 *The algorithm converges in a maximum of K iterations if all control messages are exchanged in finite time, where K is the number of piecewise intervals defining the utility functions.*
Proof *see the Appendix.*

Proposition 2 *The estimated complexity of the algorithm is $O(Km)$.*
Proof *see the Appendix*

It is important to note that Algorithm–2 computational complexity does not depend on the size of the network but only on K and m. For a fixed K, complexity increases linearly with the number of aggregates. Therefore, the main observation is that in terms of computational burden Algorithm–2 scales with both network topology (nodes and links, $\boldsymbol{v}$ and $\boldsymbol{l}$) and network traffic (aggregates, $\boldsymbol{m}$).

6 Simulation Results

In this section we carry out two different experiments. Firstly in Subsection 6.1, we compare the solution returned by the lexicographic criterion with the results from the two most well known bandwidth allocation algorithms:

maxmin [BER92] and *proportional fairness* [KEL98]. We are interested in assessing the gains that may be achieved by using our approach when compared to those traditional bandwidth allocation schemes.

Then in Subsection 6.2 we set up an experiment to study the behaviour of the proposed bandwidth allocation algorithm (Algorithm–2) and assess its response under dynamic network environments. Given system dynamics and delays incurred in information exchange, no distributed procedure is able to return exact optimal solutions, but only approximations. We compare the results retuned by our distributed algorithm (Algorithm–2) with optimal allocations computed offline by its centralized counterpart (Algorithm–1).

6.1 Comparison with Maxmin and Proportional Fairness

In this subsection, the bandwidth allocation solution under the lexicographic (lex) criterion is computed and compared here with *maxmin fairness* (mmf) *proportional fairness* (ppf) solutions. The objectives of this experiment are twofold: (i) to determine whether the lexicographic criterion is able to improve fairness in the system, and (ii) whether the possible improvement is achieved at the expenses of other important system performance parameter – average utility.

The topology used is based on the high speed links (above Gbps) of the Multi-Gigabit pan-European Network Backbone Topology [7]. We assume that a capacity of 100Mbps was reserved for the experiment in all network links and should be adequately allocated to the different applications in the system. Applications may generate traffic from any network node (origin) to any other node (destination) and have an associated utility function on the form of the logarithmic expressions below:

$$u(x,j) = a_j.ln(b_j x + c_j), \quad u(I,j) = 1, \quad u(M,j) = 0 \tag{36}$$

$$a_j = \frac{1}{k_j - 10}, \quad b_j = \frac{e^{\frac{1}{a}} - 1}{I - M}, \quad c_j = \frac{I - Me^{\frac{1}{a}}}{I - M} \tag{37}$$

where I represents the ideal allocation and M the minimum requirement. By changing the parameter k different functions can be obtained: strictly concave for $k > 10$, linear for $k \to 10$, and convex for $k < 10$.

Figure 9 shows a plot of such expressions for $M = 10$Kpbs (minimum requirement, utility = 0), $I = 100$Kbps (ideal, utility = 1), and $k = \{5, 6, \ldots, 14, 15\}$ (concavity parameter).

[7] www.dante.net/geant/about-geant.html

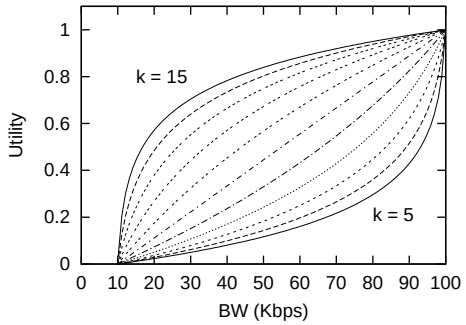


Fig. 9. Logarithm-based utility functions for $k = \{5, \ldots, 15\}$ (eleven curves).

We use the minimum utility in the system and the average utility as the performance measures to assess the quality of each solution returned,

$$min_u = \min_{j \in \mathcal{N}} \{u(\boldsymbol{x}, j)\} \qquad avg_u = \frac{1}{|\mathcal{N}|} \sum_{j \in \mathcal{N}} u(\boldsymbol{x}, j) \tag{38}$$

Thus, for each of the three schemes we obtained the respective solutions and computed: $\{min_u^{mmf}, avg_u^{mmf}\}$ for maxmin fairness, $\{min_u^{ppf}, avg_u^{ppf}\}$ for proportional fairness, and $\{min_u^{lex}, avg_u^{lex}\}$ for the lexicographic solution. The solution of higher minimum utility is preferred under a *Rawlsian* perspective, while a higher average utility is more desirable from an *utilitarian* viewpoint. The gains on using the *lexicographic* criterion can be evaluated from the ratios below:

$$\text{gain in } min_u = \frac{min_u^{lex}}{min_u^{ppf}} \quad \text{and} \quad \frac{min_u^{lex}}{min_u^{mmf}}$$

$$\text{gain in } avg_u = \frac{avg_u^{lex}}{avg_u^{ppf}} \quad \text{and} \quad \frac{avg_u^{lex}}{avg_u^{mmf}}$$

where gains above one indicates a better performance since in this case the minimum and average utility in the system under the lexicographic criterion are greater than the others.

The system was subjected to 10,000 individual applications whose utility function were described by the logarithmic curves in Figure 9. A total of 20 scenarios were considered in the experiment, where for each scenario a different and random assignment of flows to origin-destination pairs and utility functions took place.

Table–1 presents a summary of all the results obtained in terms of minimum utility gains (min_u) and average utility gains (avg_u). The results are presented using two decimal digit precision. According to the table, lexicographic allocations provided a sound improvement on the performance of the system since the gains in minimum utility were expressly high. The lowest improvement in

relation to minimum utility (min_u) was about 62% over proportional fairness, and 59% over maxmin fairness. One important aspect is that the improvement on minimum utility was achieved without compromising the average utility. In fact, for most of the cases the average utility was slightly higher under lexicographic allocations.

The experiment presents further evidence that under a multi-application environment, where different types of utility functions coexist, the lexicographic criterion provides better allocations in terms of the utility viewpoint (higher minimum without compromising the average). Since utility functions are associated with the performance of each application, lexicographic allocations improve overall system performance in this environment.

Scenario	Gain in min_u		Gain in avg_u	
	lex/ppf	*lex/mmf*	*lex/ppf*	*lex/mmf*
1	3.72	3.72	0.99	1.01
2	2.62	3.39	1.06	1.00
3	2.76	1.62	0.90	1.01
4	4.88	3.44	1.01	1.01
5	6.34	4.12	1.03	1.11
6	3.70	2.51	0.95	0.97
7	2.55	1.62	1.07	1.08
8	2.70	3.28	1.00	0.98
9	2.57	2.83	1.02	1.04
10	3.17	1.84	0.97	1.02
11	3.49	3.98	1.04	1.03
12	2.66	2.72	1.06	1.09
13	3.93	3.68	1.02	1.01
14	2.83	1.59	0.97	0.99
15	1.62	1.73	1.00	0.97
16	3.96	1.68	0.96	0.98
17	3.76	2.02	1.00	1.05
18	3.90	3.09	1.00	1.01
19	3.44	2.98	1.00	1.01
20	1.84	1.94	1.03	1.01
Lowest	**1.62**	**1.59**	**0.90**	**0.97**
Mean	**3.32**	**2.69**	**1.00**	**1.02**
Highest	**6.34**	**4.12**	**1.07**	**1.11**

Table–1: Gains obtained on each of the 20 scenarios studied

6.2 Evaluation of Proposed Algorithm

In this experiment we use the same network topology as before but this time the system is subjected to two different types of traffic: regular *best-effort* (BE) flows and G.711 flows. For the BE flows we adopt the utility function

derived in [LIA00], using $b_{\min} = 22\text{Kbps}$ [8] we have: $u_{BE} \approx 1 - \left(\frac{22}{x}\right)^2$. Including all packet headers and packetization interval, G.711 encoded voice flows may need up to 96Kbits/s of bandwidth for a high quality transmission [YAM01]. The results presented in [JIA02] show an almost straight line relating the performance of G.711 applications to packet losses, we use those performance results to build up a linear utility function for G.711 flows: $u_{G711} = \frac{x}{39} - 1.46$.

We consider a system where a fixed number of aggregated flows are routed through pre-established paths from origin to destination pairs. Each aggregate was composed by individual flows whose utility functions are the piecewise linear forms of BE and G.711 just described. The number of individual flows inside each aggregate was varied to resemble a dynamic environment. To achieve that we simply model the number of active connections n inside each aggregate by a M/M/1 system with arrival rate λ and departure rate μ. The *exit rate* from any state is given by $\lambda + \mu$ and the mean time in which the process leaves the state is: $\Delta t = 1/(\lambda + \mu)$ [KAR75]. This equation gives a simple temporal measure about the dynamics of the system, i.e. how fast the process changes states. We evaluate the performance of the algorithm when subjected to this simplified aggregate traffic model.

We assume that control messages used in Algorithm–2 have priority over regular data packets so that they are likely to be subjected just to the propagation delays on the links. According to the distances between the European capitals and speed of light in the fiber cables, the diameter [9] of the chosen network topology is around 15 ms, and so the minimum round-trip-time is approximately 30 ms. Given the priority queueing of control packets, the high reliability of the links and low dropping probability, we also assume that a control message is very likely to be acknowledged before 300 ms, which is used to bound the maximum round-trip-time.

Algorithm–2 takes a maximum of K iterations (number of piecewise linear segments defining the utility functions) to find the solution. In our experiment the utility functions were defined over 10 line segments ($K = 10$), that is a maximum of 10 iterations will take place on each run of the algorithm. Given the low complexity of the algorithm, we assume that iteration time takes no more than the bound considered (300 ms). Hence, one algorithm run takes less than 3 s, which gives also the lower bound between two consecutive runs of the algorithm. We varied the parameters ΔT (time between two consecutive runs of the algorithm $\geq$ 3 s) and Δt (aggregated flow dynamics), and computed the errors incurred in allocation and utility when compared to the optimal

[8] From [MAT97] this value correspond to $MSS = 1460$ bytes, $RTT = 500$ms and $\kappa = 1$

[9] maximum shortest path among all end-to-end nodes using the propagation delay as metrics

bandwidth allocation given by a centralized approach (Algorithm–1) for the most updated global network state. The errors were measured according to the percentage deviation from such optimal values.

For each different scenario we ran a 30000 s simulation experiment and computed the errors just before a new algorithm update. It is expected that errors will increase with the difference between ΔT and Δt, since high aggregate dynamics makes algorithm solutions be outdated faster. The results are presented next in the form of scattered and histogram plots.

6.3 Scenario 1: $\Delta T = 3\,s$

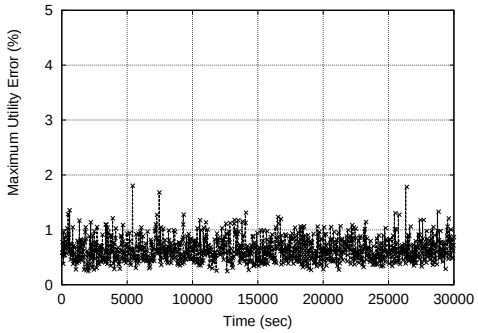
For this case, Algorithm–2 run at each 3 s period requiring good synchronization among network links to start the procedures on time. It is not expected that such timed approach will be used in practice, but as long as it provides a lower bound for ΔT as explained before, we present the results as reference.

Figs. 10(a)–10(f) presents the maximum utility errors [10] experienced by the flows at each 3 s interval. The results in the graphs reflect the behavior of the algorithm for three different aggregate dynamics: $\Delta t = 300$ ms (low), $\Delta t = 3$ ms (medium), and $\Delta t = 30\,\mu$s (high). Left column graphs show scattered plots, and right column graphs the corresponding histogram computed for 10 equally spaced intervals (bins).

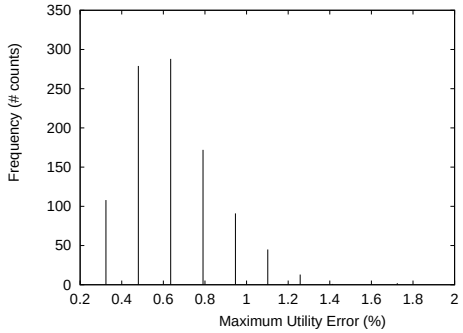
For the case of low aggregated flow dynamics, Figs. 10(a) and 10(b) show that all errors fell below 2% and so the distributed algorithm was able to track the state of the system with high accuracy. The results confirmed a very good performance of the distributed algorithm when network delays are comparable to aggregate dynamics.

When aggregated flow dynamics is increased by a factor of 100 (Δt is reduced from 300 ms to 3 ms) maximum utility errors are also increased: the mean of the histogram in Fig.10(c) grows to more than 10 times the previous mean in Fig.10(a). In this faster environment the delays involving algorithm updates have a stronger effect on the optimality of the allocations returned, however most of the errors fell below 10% still characterizing a good algorithm performance.

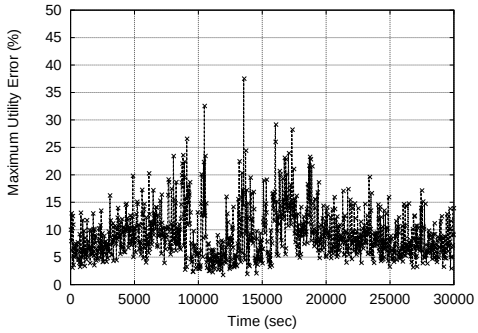
[10] maximum error in utility among all flows in the system, computed from the percentual deviations of instantaneous optimal values



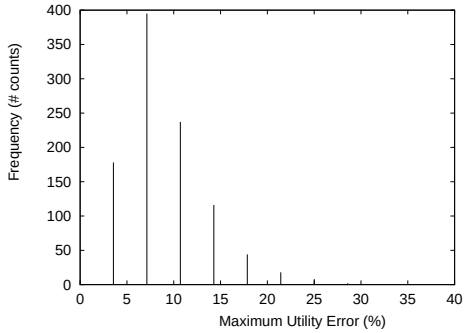
(a) scattered plot, $\Delta t = 300$ms



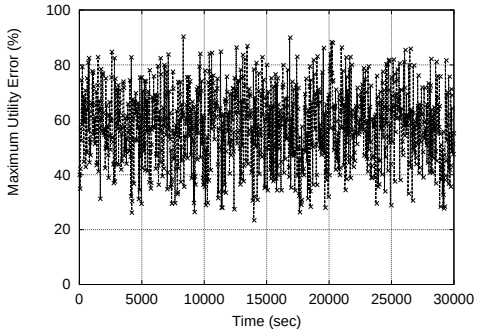
(b) histogram, $\Delta t = 300$ms



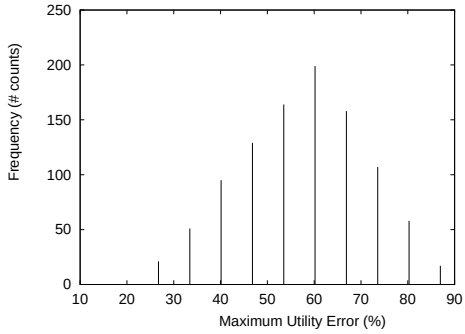
(c) scattered plot, $\Delta t = 3$ms



(d) histogram, $\Delta t = 3$ms



(e) scattered plot, $\Delta t = 30$us



(f) histogram, $\Delta t = 30$us

Fig. 10. Maximum utility errors.

If we further increase aggregate dynamics up to $\Delta t = 30\,\mu$s as shown in Fig.10(f), the effect of network dynamics is exacerbated and the algorithm is not able to cope with such fast environment. In the case of utility, errors were as high as 90% making the use of the algorithm in this extremely dynamic situation not applicable. In fact, according to our model and the results presented, the algorithm is able to accurately track network dynamics up to a timescale of 3 ms, three orders of magnitude faster than algorithm response for the studied topology.

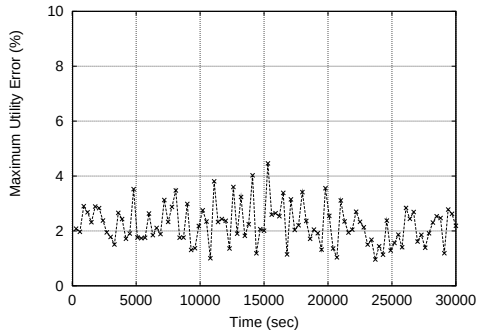
6.4 Scenario 2: $\Delta T = 300\,s$

This scenario provides light network overhead since the algorithm is executed in a lower frequency, at 5 min intervals. The expected drawback is the lack of capacity to track high network dynamics, however the behavior observed in the previous scenarios recurs here and the results obtained indicate good performance for Δt up to 300 ms.

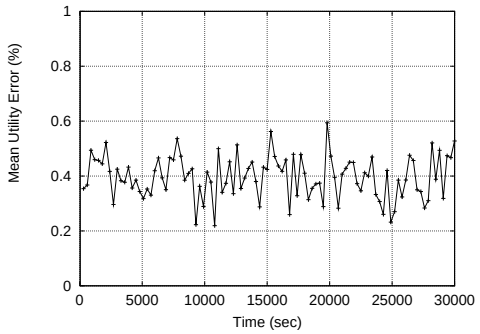
Figs. 11(a) and 11(b) give results for $\Delta t = 3$ s, where maximum errors were around 2% and mean errors around 0.4%. For the faster environment in Figs. 11(c) and 11(d) almost all errors fell below 10%, which ensure a good response of the algorithm up to this timescale. The last two graphs, Figs. 11(e) and 11(f), show results for even faster dynamics and similarly to the previous scenario errors are high characterizing poor performance.

Both scenarios have shown a very similar pattern in terms of performance. According to the results presented, the algorithm was able to track aggregate dynamics with low errors up to a ratio $\Delta T/\Delta t \approx 1000$. The choice for the algorithm update intervals will depend on two main issues: the timescales of aggregate dynamics and the network delays incurred in messages exchange.

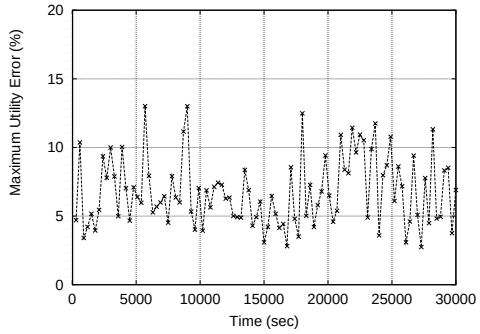
From the network topology studied, we have estimated beforehand that the algorithm will need at least 3 s to successfully complete the whole cycle and return a solution. This provides a physical bound to algorithm performance as seen in Scenario 1 where for $\Delta t < 3$ ms errors were exacerbated. For the extreme case where network delays are large and aggregate dynamics are excessively high, most of the distributed approaches will be ineffective since there will be no feasible way to track system dynamics. In this case, the simplest solution may be the use of fixed allocations for the aggregates instead of employing dynamic bandwidth allocation algorithms.



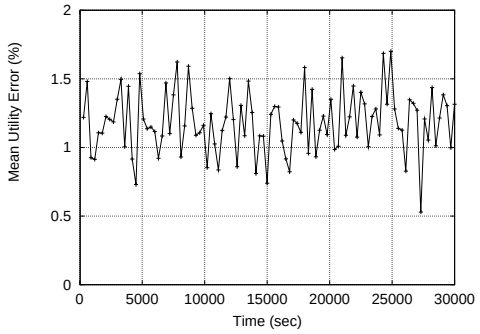
(a) Max. utility errors for $\Delta t = 3$s



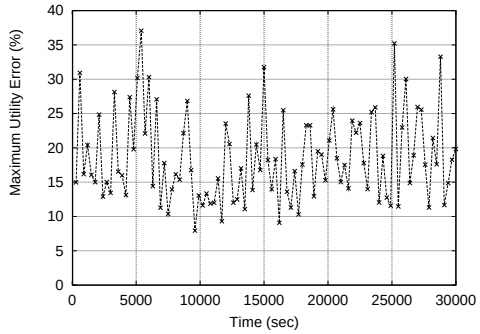
(b) Mean utility errors for $\Delta t = 3$s



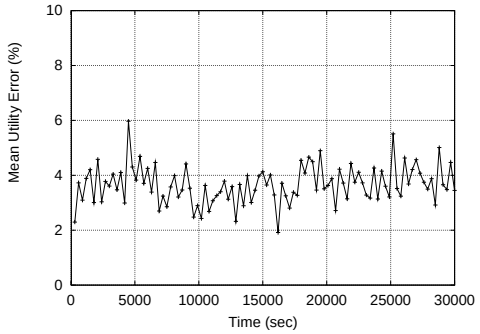
(c) Max. utility errors for $\Delta t = 300$ms



(d) Mean utility errors for $\Delta t = 300$ms



(e) Max. utility errors for $\Delta t = 30$ms



(f) Mean utility errors for $\Delta t = 30$ms

Fig. 11. Maximum and Mean utility errors.

7 Conclusion

We studied the problem of bandwidth allocation of network resources in an environment where multiple types of applications coexist. The framework of Welfare Economics was employed to characterize *fair* and *efficient* solutions. Among several different allocation criteria, the *lexicographic* criterion was selected given that it satisfies simultaneously those two fundamental properties. From simulation results it could be seen that such criterion produces desirable allocation solutions. The underlying conclusion is that the *lexicographic* criterion emerges as a good method to tackle network resource allocation problems and should be more widely employed in the networking area. It is particularly advantageous in the case of multi-class / multi-service networks where it can be used to determine dynamic interclass partitions.

The use of piecewise linear utility functions while simplifies the algorithm, did not impose any constraint on the concavity of the functions making the approach suitable for a wide range of applications. Furthermore, the number of intervals defining the piecewise functions can be arbitrarily increased tightening deviations from any given utility function closed-form expression.

One initial centralized procedure (Algorithm–1) was studied and further a distributed bandwidth allocation algorithm (Algorithm–2) was proposed based on WFQ weight settings to enforce *lexicographic* solutions. Simulation results showed that our proposed distributed algorithm was able to track network dynamics with low errors from the zero-delay centralized solution. Although it requires some sort of synchronization among network nodes, its simplicity and possibility to handle flow aggregates constitute necessary ingredients for the application in large scale IP networks.

Acknowledgements

The authors would like to thank the referees for their most helpful and constructive comments. The first author would also like to express his sincere gratitude to the Military Institute of Engineering and to the CNPq - Brazilian Government for all the support received during this research.

References

[ABR01] S. P. Abraham and A. Kumar, "A New Approach for Asynchronous Distributed Rate Control of Elastic Sessions in Integrated Packet Networks," *IEEE/ACM Transactions on Networking*, 9(1):15–30, February 2001.

[ARU96] A. Arulambalam, X. Chen, and N. Ansari, "Allocationg Fair Rates for Available Bit Rate Service in ATM Networks," *IEEE Communications Magazine*, 34(11):92–100, November 1996.

[AWE98] B. Awerbusch and Y. Shavitt, "Converging to Approximated Max-Min Flow Fairness in Logarithmic Time," *IEEE INFOCOM'98*, 3:1350–1357, April 1998.

[AZE94] M.M. de Azevedo and D.M. Blough, "Fault-tolerant Clock Synchronization for Distributed Systems with High Message Delay Variation", *Proceedings of the IEEE Workshop on Fault-tolerant Parallel and Distributed Systems*, 1994, pp. 268–277.

[BER92] D. Bertsekas and R. Gallager, *Data Networks*, Prentice Hall, 2nd Edition, 1992.

[BIA00] G. Bianchi and A.T. Campbell, "A Programmable MAC Framework for Utility-Based Adaptive Quality of Service Support," *IEEE Journal on Selected Areas in Communications*, 18(2):244–255, Feb. 2000.

[BLA98] S. Blake, D. Black, M. Carlson, E. Davies, and W. Weiss, "An Architecture for Differentiated Services," *IETF DiffServ-WG*, RFC 2475, December, 1998.

[BON01] T. Bonald and L. Massoulie, "Impact of Fairness on Internet Performance," *ACM SIGMETRICS Performance Evaluation Review*, 29:82–91, June 2001.

[CAO99] Z. Cao and E. W. Zegura, "Utility Max-Min: An Application-Oriented Bandwidth Allocation Scheme," *Proceedings of the IEEE INFOCOM'99*, 2:793–801, March 1999.

[CHA95] A. Charny, D. Clark, and R. Jain, "Congestion Control with Explicit Rate Indication," *IEEE ICC'95*, 3:1954–1963, June 1995.

[CHE98] S. Chen and K. Nahrstedt, "An Overview of Quality of Service routing for the Next Generation High-Speed Networks: Problems and Solutions," *IEEE Network Magazine*, 12(6):64–79, Dec. 1998.

[COL01] R. G. Cole and J. Rosenbluth, "Voice over IP Performance Monitoring," *ACM Computer Communication Review*, 4(3), 2001.

[CRO01] P. Gevros, J. Crowcroft, P. Kirstein, and S. Bhatti, "Congestion Control Mechanisms and the Best Effort Service Model," *IEEE Network*, 15(3):16–26, May/June 2001.

[DAS02] A. Das, D. Dutta, and A. Helmy, "Fair Stateless Aggregate Traffic Marking Using Active Queue Management Techniques," *LNCS 2496 - 5th IFIP/IEEE International Conference on Management of Multimedia Networks and Services*, 211–223, Oct. 2002.

[DAS77] C. D'Aspremont and L. Gevers, "Equity and the Informational Basis of Collective Choice," *The Review of Economic Studies*, 44(2):199–209, June 1977.

[FLO99] S. Floyd and K. Fall, "Promoting the Use of End-to-End Congestion Control in the Internet," *IEEE/ACM Transactions on Networking*, 7(4):458–472, Aug. 1999.

[GAF84] E.M. Gafni and D.P. Bertsekas, "Dynamic Control of Session Input Rates in Communication Networks," *IEEE Transactions on Automatic Control*, AC-29(11):1009–1016, Nov. 1984.

[GUE99] R. Guerin and V. Peris, "Quality-of-Service in Packet Networks: Basic Mechanisms and Directions," *Computer Networks*, 31(3):169–189, Feb. 1999.

[ITU00a] Inter. Telecomm. Union, "Methodology for the Subjective Assessment of the Quality of Television Picture," *Radiocommunication Sector of ITU*, Recommendation ITU-R BT.500-10, March 2000.

[ITU97] Inter. Telecomm. Union, "Methods for the Subjective Assessment of Sound Quality – General Requirements," *Radiocommunication Sector of ITU*, Recommendation ITU-R BS.1284, Oct. 1997.

[ITU99] Inter. Telecomm. Union, "Subjective Video Quality Assessment Methods for Multimedia Applications," *Telecommunication Standardization Sector of ITU*, Recommendation ITU-T P.910, Sep. 1999.

[JAF81] J.M. Jaffe, "Bottleneck Flow Control," *IEEE Transactions on Communications*, Vol. 29, No. 7, July, 1981, pp. 954–962.

[JIA02] W. Jiang and H. Schulzrinne, "Comparison and Optimization of Packet Loss Repair Methods on VoIP Perceived Quality under Bursty Loss," *Proc. of ACM Inter. Work. on Net. and Oper. Sys. for Dig. Audio and Video*, 73–81, May 2002.

[KAR75] S. Karlin, *A First Course in Stochastic Processes*, Academic Press, London, 1975.

[KEL98] F. P. Kelly, A.K. Maulloo, and D.K.H. Tan, "Rate Control for Communication Networks: Shadow Prices, Proportional Fairness and Stability," *Journal of the Operational Research Society*, 49:237–252, 1998.

[LIA00] R. Liao and A. T. Campbell, "Dynamic Edge Provisioning for Core IP Networks," *Proceedings of the IEEE International Workshop on Quality of Service*, 148–157, June 2000.

[LOW99] S. Low and D. Lapsey, "Optimization Flow Control-I: Basic Algorithm and Convergence," *IEEE/ACM Transactions on Networking*, 7(6):861–874, December 1999.

[LU03] X. Lu, S. Tao, M. E. Zarki, and R. Guerin, "Quality-based Adaptive Video Over the Internet," *Communication Networks and Distributed Systems Modeling and Simulation Conference*, Jan. 2003.

[MA99] Q. Ma and P. Steenkiste, "Supporting Dynamic Inter-Class Resource Sharing: A Multi-Class QoS Routing Algorithm," *IEEE INFOCOM'99*, 2:649–660, March 1999.

[MAS02] L. Massoulie and J. Roberts, "Bandwidth Sharing: Objectives and Algorithms," *IEEE/ACM Transactions on Networking*, 10(3):320–328, June 2002.

[MAS95] A. Mas-Colell, M.D. Whinston, and J.R. Green, *Microeconomic Theory*, Oxford University Press, 1995.

[MAT97] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm," *ACM Computer Communication Review*, 27(3):1–15, July 1997.

[MIL96] D. Mills, "Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI," *IETF Network-WG*, RFC 2030, October, 1996.

[MIT99] D. Mitra and K. G. Ramakrishnan, "A Case Study of Multiservice, Multipriority Traffic Engineering Design for Data Networks," *Proceedings of the IEEE Globecom'99*, 1B:1077–1083, Dec. 1999.

[Mo00] J. Mo and J. Walrand, "Fair End-to-End Window-Based Congestion Control," *IEEE/ACM Transactions on Networking*, 8(5):556–567, October 2000.

[NAH98] K. Nahrstedt and S. Chen, "Coexistence of QoS and Best-Effort Flows: Routing and Scheduling," *10th International Workshop on Digital Communications: Multimedia Communications*, Italy, September, 1998.

[PAT94] B. Patt, *A Theory of Clock Synchronization*, PhD Thesis, Massachusetts Institute of Technology (MIT), United States, 1994.

[RAW99] J. Rawls, *A Theory of Justice*, Oxford University Press, 1999.

[ROB80] K. W. S. Roberts, "Interpersonal Comparability and Social Choice Theory," *Review of Economic Studies*, 47(2):421–439, Jan. 1980.

[SEN95] A. K. Sen, *Collective Choice and Social Welfare*, 4th ed., North-Holland Publishing, 1995.

[SEN99] A. Sen, "The Possibility of Social Choice," *The American Economic Review*, 89(3):349–378, June 1999.

[SHE95] S. Shenker, "Fundamental Design Issues for the Future Internet," *IEEE Journal on Selected Areas in Communication*, 13(7):1176–1188, Sep. 1995.

[VAR02] H. Varian, *Intermediate Microeconomics: A Modern Approach*, W. W. Norton, July 2002.

[YAM01] H. Yamada and N. Higuchi, "Voice Quality Evaluation of IP-based Voice Stream Multiplexing Schemes," *Proc. of the IEEE Conference on Local Computer Networks*, 356–364, 2001.

[ZHA95] H. Zhang, "Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks," *Proceedings of the IEEE*, 83(10):1374–1396, Oct. 1995.

[ZHA02] L. Zhao, J. Kim, and C. C. J. Kuo, "Constant Quality Rate Control for Streaming MPEG-4 FGS Video," *Proceedings of the IEEE International Symposium on Circuits and Systems*, May 2002.

Appendix

Proof (Theorem 1) ($\Rightarrow$) Let $\boldsymbol{x}$ be the lexicographic solution and assume that flow j (with allocation x_j) does not have an associated utility-bottleneck link. Let a be a link used by flow j and δ_a the available capacity on link a.
Case-(1): If $\delta_a > 0$ for all links a along the path p_j of flow j, then flow's j allocation can be increased to $x'_j = x_j + \delta$, where $\delta = \min\{\delta_a \,|\, a \in p_j\}$. In this case we have $u(x'_j, j) > u(x_j, j)$, which contradicts the optimality of $\boldsymbol{x}$.

Case-(2): If there is a link a along p_j where $\delta_a = 0$, then flows bottlenecked locally have utilities greater than $u(x_j, j)$. Let $\mathcal{S}$ be the set of flows bottlenecked at some link along p_j. For any $i \in \mathcal{S}$, we have $\Delta u = u(x_i, i) - u(x_j, j) > 0$, and since utility functions are monotone:

$$\forall i \in \mathcal{S},\ \exists\, x'_i < x_i \quad s.t. \quad u(x_i, i) - \frac{\triangle u}{2} = u(x'_i, i) > u(x_j, j)$$

If we decrease the allocations of all flows $i \in \mathcal{S}$ from x_i down to x'_i there will be at least an available capacity of $\triangle x = \min\{x_{_i} - x'_i \mid i \in \mathcal{S}\}$ on these links. As before, let $\delta > 0$ be the minimum available capacity among the non-bottleneck links, therefore we can increase flow's j utility to:

$$u(x'_j, j) > u(x_j, j),\ \text{where } x'_j = x_j + \min\{\triangle x, \delta\}$$

The resulting vector $\boldsymbol{x}'$ is feasible and yields a corresponding utility vector that is preferred in terms of the preference operator, $\boldsymbol{x}' \succ \boldsymbol{x} \Rightarrow \boldsymbol{x}$ is not the lexicographic solution.

($\Leftarrow$) Conversely, assume each flow has an associated utility-bottleneck link. For a given flow $i \in \mathcal{N}$ let link l be its bottleneck. For any other flow j, such that $a_l(j) = 1 \Rightarrow u(x_i, i) \geq u(x_j, j)$ and the only way to increase flow's i utility is decreasing the utility of another flow j of lower or equal utility. Since i can be any flow in $\mathcal{N}$, there is no way to get a larger utility vector with respect to the operator $\succ$ ■

Proof (Proposition 1) To facilitate explanations we simply refer to Algorithm–2 (Agg. flow on path p) as "agg-p", and Algorithm-2 (Link-l) as "link-l". First observe that agg-p send messages either in step **(2)** or step **(4)**, which are collected by link-l at step **(2)**. On the other hand, link-l only sends message in step **(4)** which are collected by agg-p in step **(3)**. The **wait** condition during collection steps (step **(3)** in agg-p and step **(2)** in link-l) guarantees that the procedure only progresses, following the necessary order, once those messages are exchanged sucessfuly. Assuming that all messages eventually go through, agg-p passes step **(3)** and goes to step **(4)**, where there are only two possibilities: i) if flow is blocked it transmits a status '0' message and stops, or ii) k is incremented for the next piecewise interval request. In case of (ii), agg-p may go to step **(2)** again and the whole procedure is repeated. However, if k has reached K, agg-p transmits a status '0' message and termintes. Thus, in a maximum of K iterations agg-p sends a status '0' message to link-l and terminates. Upon receiving a status '0' message from agg-p, link-l considers the flow blocked and updates the set $\mathcal{B}$ to include this flow. Therefore, in a maximum of K iterations link-l will have $\mathcal{B} = \mathcal{N}(l)$ and also terminates, which completes the proof. ■

Proof (Proposition 2) We now estimate the computational complexity of Algorithm–2 iterations according to the number of operations a given network node running the algorithm has to perform. Let $\boldsymbol{l}$ represents the number of links in the topology, $\boldsymbol{v}$ the number of nodes, and $\boldsymbol{m}$ the number of aggregates. First, assume that we have an even distribution of aggregates in the network so that each node is the source of $\boldsymbol{mv}^{-1}$ aggregated flows. Algorithm–2 (Agg. flow on path p) has to run for each aggregate and according to its steps the only computational burden is to collect messages from the links along path p, check the *status* field, and send back the response. Since there is a maximum of $\boldsymbol{v - 1}$ links per path the computational complexity will be $O(\boldsymbol{v})$. Each node has to run a copy of Algorithm–2 (Agg. flow on path p) for the aggregates in which it is source ($\boldsymbol{mv}^{-1}$), thus the computational complexity for the node is $O(\boldsymbol{m})$. Regarding the other part of the algorithm (Algorithm–2: Link-l) each node has also to compute the WFQ weights for their links. This procedure is of linear complexity on the number of aggregates as can be seen in steps **(2)** and **(3)**. For the worst case scenario we may assume that all aggregates in the network are crossing the node, hence the complexity of Algorithm–2 (Link-l) is $O(\boldsymbol{m})$, which makes the complexity of both procedures taken together (Algorithm–2: Agg. flow on path p and Algorithm–2: Link-l) be also $O(\boldsymbol{m})$. Since the algorithm takes a maximum of K iterations to converge, hence the complexity of the whole algorithm for each network node will be $O(K\boldsymbol{m})$. ■