User Fair Queing: Fair Allocation of Bandwidth for Users

Albert Banchs
NEC Europe Ltd., Network Laboratories Heidelberg
E-mail: Albert.Banchs@ccrle.nec.de

***Abstract*—*User Fairness* aims at fairly distributing the network resources among users, where a user is an entity that can possibly send different flows through different paths. In this paper we first propose a criterion for *user fairness* based on political economics fairness theory: the *User Maxmin Fairness* criterion. Then we propose an architecture, *User Fair Queuing* (UFQ), that provides *user maxmin fairness* without keeping per-user state in the core nodes. UFQ requires neither admission control nor signaling. We present simulations and analysis on the performance of the proposed architecture.**

I. Introduction

In a commercial Internet, the traffic behavior is determined by the contracts between the ISPs and the users, where a user can be a dial-up user, a corporate network or a group of individual customers or networks. Since the user is the entity with whom the contract is signed, it should also be the unit to which network resources are allocated. However, while much research in the past has been directed to fair resource allocations for flows (see e.g. maxmin fairness [1] and proportional fairness [2][1]), much less effort has been invested on fair allocation of resources for users. The work done in this paper tries to fill this gap: we study how to fairly share the network resources among users, when a user can possibly send several flows through different paths. Hereafter we call the concept of fairly distributing resources among users *user fairness*.

User fairness in the current best-effort Internet is based on *TCP fairness* [4], which distributes network resources among users on a flow basis: with TCP, each flows receives a throughput inversely proportional to its round-trip delay. However, not all flows in the Internet use the TCP protocol and it is relatively easy for non-adaptive sources to gain greater shares of network bandwidth and thereby starve other, well-behaved, TCP friendly sources. For that reason the TCP fairness solution relies on TCP friendly behavior of the applications in order to fairly share the network resources. In addition, the fact that TCP fairness assigns resources on a flow basis makes the amount of resources received by each user dependent on the number of flows sent, which may lead to an unfair overall distribution of the resources.

One approach for user fairness that overcomes the above mentioned problems of TCP fairness is the *pre-allocation of resources on a link-basis*. This is the fairness concept behind fair user-based queuing schemes, such as Weighted Fair Queuing (WFQ) [5] and Class-based Queuing (CBQ) [6]. These queuing algorithms distribute the link bandwidth among users in a fair way, independent of the number of flows that each user is sending through this link and the aggressiveness of the sources. One of the remaining drawbacks of these approaches is that since they work on a local basis, they cannot ensure a fair distribution of the overall network resources.

In the last few years, architectures for providing Quality of Service (QoS) in the Internet have been the focus of extensive research. These research efforts have identified two fundamentally different approaches for QoS: Integrated Services (IntServ) [7] and the Differentiated Services (DiffServ) [8]. The fairness concept behind these architectures is the *resource allocation on demand*. With IntServ and DiffServ, users request a certain amount of network resources; the request is processed by an admission control entity in order to determine whether the available resources can satisfy the request, and the user is notified of the acceptance or rejection of his request. One of the important challenges of these architectures is precisely how to perform this admission control. With IntServ this was done on a flow basis, which leads to unscalability, and with DiffServ admission control is still an open issue (in [9] and [10] we study this issue for one-to-one and one-to-any services, respectively).

In this paper we introduce a new concept for user fairness that overcomes the drawbacks of the above mentioned approaches. The proposed concept distributes resources equally among users taking into account the overall usage of the network resources, while admission control is avoided. The goal is to provide two users that pay a same price with the same amount of resources, independent of the number of flows and links used by each user and the aggressiveness of the sources. We focus on elastic traffic and a single domain. In [11] we extend the work presented here for real-time traffic, and in [12] we extend it with service differentiation and inter-domain communication.

The paper is structured as follows: in Section II we provide a review on fairness concepts in computer networks. These concepts are then applied in Section III to the problem of fairly allocating resources for users, resulting in the *user maxmin fairness* criterion. In the following section we present a core stateless mechanism, the *User Fair Queuing* architecture, that implements the proposed fairness criterion. In Section V we present our simulation results and Section VI concludes the paper with a summary. A longer version of this paper containing proofs of the theoretical results as well as additional simulation results, can be found at `http://www.ccrle.nec.de/reports/2002/ufq.pdf`

[1] In [3] we provide an overview on this research.

0-7803-7476-2/02/$17.00 (c) 2002 IEEE.

II. Review on Fairness Concepts in Computer Networks

The concept of *fairness* has been studied in various scientific areas. Most thorough and theory-based approaches arose from the field of political science and political economics. In this field, the concepts of *utility* [13] and *welfare* [14] functions were developed for the purpose of defining *fairness*. In this section we review this theory from the computer network's viewpoint. In [3] we provide a more extensive description of the concepts explained in this section.

A. Utility function

In order to express the user's satisfaction with the service delivered to him by the network, network performance must not be measured in terms of network-centric quantities like throughput, packet drops or delay, but should be rather evaluated in terms of the degree to which the network satisfies the service requirements of each user's applications. For instance, if a particular application cares more about throughput than delay, or vice-versa, the network service to that application should be evaluated accordingly.

Utility functions in computer networks [15] formalize the above notion of network performance. Let s_i describe the service delivered to the i'th application or user; s_i contains all the relevant measures (delay, throughput, packet drops, etc.) of the delivered service. Then, the utility function u_i maps the service delivered s_i into the performance of the application; increasing u_i reflects increasing application performance. The utility function, thus, describes how the performance of an application depends on the delivered service.

In the following, we elaborate on the utility function for elastic traffic, i.e. the traffic type on which we are concerned in this paper. Examples of elastic applications are file transfer, electronic mail and remote terminal. These applications are tolerant of delays and their satisfaction is basically measured in terms of bandwidth. Therefore, bandwidth is the only relevant measure for network resources in this paper and the only one that will be considered hereafter.

Elastic applications experience a marginal rate of performance enhancement as bandwidth is increased, so their utility function is strictly concave everywhere. Following [2], in this paper we use the logarithmic function to represent the utility of elastic traffic (see Figure 1). Thus, the utility of an elastic flow i will be

$$u_i(r_i) = log(r_i) \tag{1}$$

where r_i is the flow's throughput.

B. Welfare function

The basic problem of *welfare* is to determine which of the feasible resource allocations should be selected. For this purpose, a *Welfare function* $W(u_1, u_2, \ldots, u_n)$ that aggregates the individual utility functions u_i is defined. The resource allocation selected (called the *fair resource allocation*) is then the one that maximizes the welfare function:

$$max(W(u_1, u_2, \ldots, u_n)) \tag{2}$$

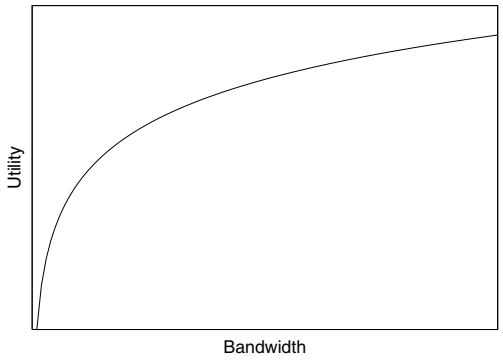

Fig. 1. Utility function of an elastic traffic flow.

For different purposes, different welfare functions exist, each corresponding to a different fairness criterion. A fairness criterion, thus, is defined by the welfare function that it maximizes. The most widely used fairness criteria in computer networks are *maxmin fairness* and *proportional fairness*.

1) Maxmin fairness: Maxmin fairness [1] is the most popular fairness concept in computer networks. This fairness criterion corresponds to the welfare function:

$$W(u_1, u_2, \ldots, u_n) = min(u_1, u_2, \ldots, u_n) \tag{3}$$

Maxmin fairness, thus, yields a solution $u = (u_1, u_2, \ldots, u_n)$ for $max(min(u_1, u_2, \ldots, u_n))$. A maxmin fair allocation has the property that for all i, u_i cannot be increased without simultaneously decreasing u_j for some j with $u_j \leq u_i$.

The idea behind maxmin fairness is to distribute resources as equally as possible among the competing entities. As a consequence, with this criterion the most poorly treated entities are given the greatest possible allocation.

2) Proportional fairness: The proportional fairness criterion [2] is becoming increasingly popular in the field of computer networks. A proportional fair allocation is the solution to the welfare maximization problem with the welfare function sum of utilities:

$$W(u_1, u_2, \ldots, u_n) = \sum_i u_i \tag{4}$$

and with the individual utility functions u_i of Equation 1.

Proportional fairness, thus, yields a solution u for $max(\sum_i u_i)$. A proportional-fair allocation has the property that for any other feasible allocation u^*, the aggregate of proportional changes is zero or negative, i.e. $\sum_i (u_i^* - u_i)/u_i \leq 0$.

The idea behind proportional fairness is to maximize the overall performance. With proportional fairness, a worse treated entity may see its utility decreased if this allows a large enough increase to an already better treated entity.

3) Weighted Fairness: Both maxmin and proportional fairness criteria can be generalized on introducing weights W_i associated with each entity as a means to express the relative value of this entity for the system [16]. With weighted fairness, the utility received by an entity in the fair allocation will increase with its associated weight W_i.

The introduction of weighting leads to the following welfare function for *weighted maxmin fairness*

$$W(u_1, u_2, \ldots, u_n) = min(u_i(r_i/W_i)) \tag{5}$$

and *weighted proportional fairness*

$$W(u_1, u_2, \ldots, u_n) = \sum_i W_i \cdot u_i \tag{6}$$

Weighted maxmin fairness aims at distributing resources among the competing entities proportionally to their weights. Weighted proportional fairness aims at maximizing the overall performance when some entities have a higher value than others.

III. User Fairness

User Fairness deals with the allocation of bandwidth among users, when a user may send one or more flows, possibly through different paths. Each flow i of user u experiences a certain utility u_i, which depends on its allocated bandwidth r_i as defined in Equation 1.

Following the fairness concepts explained in the previous section, a user fair allocation is the one that maximizes the welfare function W that aggregates the individual utilities u_i, $W(u_i)$. In this section we study which is the appropriate welfare function for the problem of user fairness.

A. Welfare function composition

According to the definition of welfare in Section II, the utility experienced by a user u, u_u, is the result of aggregating with a welfare function the utility of the flows of this user (intra-user aggregation). We call W_{intra} the welfare function that performs this intra-user aggregation:

$$u_u = \underset{i \in I_u}{W_{intra}}(u_i) \tag{7}$$

where I_u is the set of flows of user u.

Similarly, the total welfare experienced in the network is the result of aggregating the individual utilities of all the users in the network (inter-user aggregation). We call W_{inter} the welfare function that performs this inter-user aggregation:

$$W = \underset{\forall u}{W_{inter}}(u_u) = \underset{\forall u}{W_{inter}}(\underset{i \in I_u}{W_{intra}}(u_i)) \tag{8}$$

The bandwidth allocation for user fairness, thus, will be the one that maximizes the welfare function $W_{inter}(W_{intra}(\cdot))$, choosing the appropriate functions for W_{inter} and W_{intra}. We will choose the functions W_{inter} and W_{intra} according to the goal of providing a good level of inter and intra user fairness as described in the following.

B. Inter and Intra User fairness

We say that a bandwidth allocation is *inter-user fair* if the network bandwidth is fairly distributed among the different users. Similarly, we say that a bandwidth allocation is *intra-user fair* if the bandwidth assigned to a user is fairly distributed among his flows. Inter and intra user fairness are better illustrated in the example of Figure 2. In this example we have a network with three links (a,b,c) and three users (1,2,3). The first user (user 1) is sending three flows, one through each of the links (a,b,c), the second (user 2) is sending two flows, one through link a and the other through link b, and the third user (user 3) is sending only one flow through link c. All links have a capacity normalized to 1.

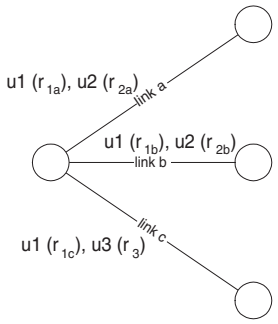

Fig. 2. Example of inter and intra user fairness.

An inter-user fair allocation for the above scenario would be the following:

$$\begin{aligned} r_{1a} = 1/2 \quad & r_{2a} = 1/2 \\ r_{1b} = 1/2 \quad & r_{2b} = 1/2 \\ r_{1c} = 0 \quad & r_3 = 1 \end{aligned}$$

Note that in the above allocation all users get the same total bandwidth (1 unit of capacity) and therefore the allocation is inter-user fair. However, if we look on how the bandwidth allocated to user 1 is distributed among his flows, we observe an extreme degree of intra-user unfairness. User 1 will most probably not be satisfied with the above allocation, since one of his flows is totally starved.

Another possible allocation that corrects the intra-user unfairness of the first one is the following:

$$\begin{aligned} r_{1a} = 1/2 \quad & r_{2a} = 1/2 \\ r_{1b} = 1/2 \quad & r_{2b} = 1/2 \\ r_{1c} = 1/2 \quad & r_3 = 1/2 \end{aligned}$$

The above distribution provides a perfect level of intra-user fairness, since for each user, all his flows experience the same throughput. However, the level of inter-user fairness is poor: in link c, users 1 and 3 are allocated the same bandwidth, even though user 1 is using more network resources than user 3 in total. User 3 will most probably not be satisfied with this allocation.

We conclude that a *user fair* allocation should provide a good level of both inter and intra user fairness. In the following we study which welfare functions W_{inter} and W_{intra} to choose in order to achieve this goal.

C. User Maxmin Fairness

The goal of inter-user fairness is to treat the different users as equally as possible. In Section II-B.1 we have argued that

the fairness criterion that best meets this goal is the *maxmin fairness* criterion. As a consequence, we have chosen to use the welfare function minimum for the aggregation of the utilities of the different users:

$$W_{inter}^{fairness} = min(u_u) \ \ \forall \text{ user } u \text{ in the network} \qquad (9)$$

The goal of intra-user fairness is to allocate the bandwidth received by a user among his flows as equally as possible to the user's desired distribution. In Section II-B.3 we have argued that the fairness criterion that best meets this goal is the *weighted maxmin fairness* criterion. This is the criterion we have chosen for intra-user aggregation, as expressed by the following welfare function:

$$W_{intra}^{fairness} = \min_{i \in I_u}(u_i(r_i/W_i)) \qquad (10)$$

with the constraint

$$\sum_{i \in I_u} W_i = 1 \qquad (11)$$

where I_u is the set of flows of user u and W_i are the normalized weights that express the relative value of flow i for its user.

The normalization of the sum of the weights of a user to 1 (Equation 11) comes from the necessity of being able to compare the $W_{intra}^{fairness}$ of different users. Note that with Equation 11, two users that get the same total bandwidth and have this bandwidth distributed proportionally to the weights W_i experience the same $W_{intra}^{fairness}$.

The combination of $W_{inter}^{fairness}$ and $W_{intra}^{fairness}$ leads to the following definition.

Definition 1—User Maxmin Fairness: [2] A bandwidth allocation $r = (r_1, r_2, \ldots, r_n)$ is *user maxmin fair* when it maximizes

$$\min_{\forall i}(r_i/W_i) \qquad (12)$$

where r_i is the throughput experienced by flow i and W_i is its normalized weight.

The proposed criterion for user fairness leads to the following allocation for the example of Figure 2:

$$\begin{aligned} r_{1a} &= 2/5 \quad r_{2a} = 3/5 \\ r_{1b} &= 2/5 \quad r_{2b} = 3/5 \\ r_{1c} &= 1/4 \quad r_3 = 3/4 \end{aligned}$$

which is a good tradeoff between the inter and intra user fair allocations given in Section III-B.

D. User utility

The level of satisfaction of a user depends on the overall performance of his flows, where some of his flows may have a higher relative value than others. In Section II-B.3 we have argued that the welfare function that best expresses this level of satisfaction of a user is the weighted sum function, corresponding to the *weigthed proportional fairness* criterion. In the following definition of user utility we have used this welfare function to perform intra-user aggregation. The definition of user utility will be used later in the paper to model the user behavior.

[2]Note that in the special case when all users are sending just one flow, the user maxmin fairness criterion coincides with the well accepted maxmin fairness criterion for flows.

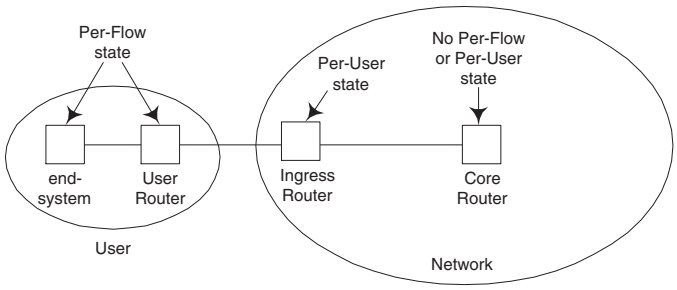

Fig. 3. UFQ architecture.

Definition 2—User Utility: The utility of user u, whose flows experience a throughput equal to r_i, is given by

$$u_u = W_{intra}^{utility}(\cdot) = \sum_{i \in I_u} W_i \cdot u_i(r_i) = \sum_{i \in I_u} W_i \cdot log(r_i) \qquad (13)$$

IV. User Fair Queuing

In this section, we propose a network architecture, *User Fair Queuing* (UFQ), that provides *user maxmin fairness* as defined in the previous section.

The proposed scheme avoids keeping per-flow or per-user state in the core and is inspired on previous work done in the context of core-stateless fair allocation of bandwidth among flows [17], [18], [19], [20]. While these proposals differ in details, they are all similar at the architectural level. Per-flow state at the core is avoided by having each packet header carry some additional state information, the *label*, which is initialized by the ingress node of the network. Then, core nodes use this information carried by the packet to decide whether in case of congestion an arriving packet should be enqueued or dropped.

UFQ is implemented in three steps: *user labeling*, *ingress label control* and *core dropping* (see Figure 3). In the first step (*user labeling*), the user assigns labels to his packets based on the sending rates of his flows and their weights (i.e. per-flow state is required). The second step (*ingress label control*) is performed at the ingress of the network. In this step, the labels assigned by the user are processed, and in case the user is labeling his packets with more resources than he should, packets are relabeled. The algorithm proposed for the ingress label control only requires keeping per-user state (i.e. it avoids per-flow state). Finally, the third step is performed at core nodes, where in case of congestion packets are dropped depending on their label. Since the *core dropping* is performed without keeping per-user or per-flow state, the proposed architecture scales with the number of users.

A. User labeling

At the user network, packet k of flow i is labeled with:

$$L_k = \frac{r_i^{send}}{W_i} \qquad (14)$$

where W_i is the weight of flow i as defined in Section III and r_i^{send} is the flow's sending rate.

For the estimation of the sending rate of flow i we use the same exponential averaging formula as in [17]. Using an exponential averaging gives more accurate estimation for bursty traffic, even when the packet inter-arrival time has significant variance.

Specifically, let t_k and l_k be the arrival time and length of the k^{th} packet of flow i. The estimated sending rate of flow i, r_i^{send}, is updated for every new packet k sent by flow i:

$$(r_i^{send})_k = (1 - e^{-(T_k/K)})\frac{l_k}{T_k} + e^{-(T_k/K)} \cdot (r_i^{send})_{k-1} \quad (15)$$

where $T_k = t_k - t_{k-1}$ and K is a constant. Following the rationale discussed in [17], in this paper we set $K = 100ms$.

B. Ingress Label Control

The probability of dropping a packet of flow i should decrease with the relative value of this flow for its user (W_i) and should increase with the flow's sending rate (r_i^{send}). As a consequence, the lower the value of the packet's label L_k, the better treatment this packet should be given. If there was no control on the labels assigned by a user, a user could exploit the system by assigning to his packets labels with lower values than Equation 14. The goal of the *ingress label control* is not to allow a user to benefit from labeling his packets with too low values.

Note that keeping per-flow information at the ingress (namely, W_i and r_i^{send}) label control could easily enforce Equation 14. However, this would introduce a considerable complexity at ingress nodes and would require the user to signal the values of W_i to the ingress. The algorithm we propose for label control avoids per-flow state and only requires per-user state.

In order to control labels with only per-user state, we define the following variable, which we call the state of user u at the ingress, S_u:

$$S_u = \frac{\underset{k \in K_u}{avg}\left(\frac{l_k}{L_k}\right)}{\underset{k \in K_u}{avg}\,(l_k)}\, r_u^{send} \quad (16)$$

where K_u is the set of packets of user u, r_u^{send} is his sending rate, l_k is the length of packet k and L_k is its label.

S_u is estimated using the following formula upon receiving the k^{th} packet of user u:

$$(S_u)_k = (1 - e^{-(\frac{l_k}{r_u^{send} \cdot K})})\frac{r_u^{send}}{L_k} + e^{-(\frac{l_k}{r_u^{send} \cdot K})} \cdot (S_u)_{k-1} \quad (17)$$

where r_u^{send} is estimated using Equation 15. The reason for using this estimation for S_u is that it allows us to bound the excess service that a user can receive, as discussed in Section IV-D.

The ingress label control algorithm is based on the observation that the following equality holds for a user who is labeling his packets as in (14):

$$S_u|_{\text{labeling of (14)}} = \sum_{i \in I_u} \left(\frac{r_i^{send}}{r_u^{send}}\right) \frac{1}{L_i}\, r_u^{send} = \sum_{i \in I_u} W_i = 1 \quad (18)$$

where I_u is the set of flows of user u, (r_i^{send}/r_u^{send}) is the portion of the data sent by user u that belongs to flow i and L_i is the label of the packets of flow i according to Equation 14.

Since S_u is a strictly decreasing function of L_k, if a user labels his packets with too low labels, this will lead to his state at the ingress, S_u, being larger than 1. Therefore, in order to avoid too low labels (which is the goal that we stated above for the ingress label control), we enforce that the state S_u of a user can never be larger than 1:

$$S_u \leq 1 \quad (19)$$

The above equation is enforced in the following way: if the arriving packet of a user has a label L_k that would lead to $(S_u)_k > 1$, then the *ingress label control* relabels the packet with a new label $L_k^{new} > L_k$ such that $(S_u)_k = 1$. Thus,

$$L_k^{new} = max\left(L_k, \frac{(1 - e^{-(\frac{l_k}{r_u^{send} \cdot K})})r_u^{send}}{1 - e^{-(\frac{l_k}{r_u^{send} \cdot K})} \cdot (S_u)_{k-1}}\right) \quad (20)$$

Note that a user who is labeling his packets as in (14) will not have his packets relabeled, since according to Equation 18, the labels L_k of this user will never lead to $(S_u)_k$ greater than 1.

Note that the above algorithm for the ingress label control only requires to keep per-user state at the ingress (namely, two values have to be stored for each user: S_u and r_u^{send}). The effectiveness of the proposed scheme will be discussed in Sections IV-D and IV-E.

C. Core dropping

In a congested link in which there is not enough bandwidth to serve all incoming packets, some packets must be dropped. Our goal is to drop packets in such a way that the resulting bandwidth distribution is *user maxmin fair*.

In UFQ, packets in a congested link l are dropped depending on their label with the following probability:

$$d_k = \begin{cases} 0 & L_k \leq L_{fair} \\ 1 - \frac{L_{fair}}{L_k} & L_k > L_{fair}, L_k^{new} = L_{fair} \end{cases} \quad (21)$$

where d_k is the probability of dropping packet k, L_k is its label and L_{fair} is the *fair label* of the congested link. L_{fair} is computed such that, if packets are dropped according to (21), the accepted rate in the congested link equals the link's capacity. Note that the non-dropped packets of a flow that experiences losses in the link are relabeled with a new label equal to the link's *fair label* L_{fair}.

The following theorem binds the algorithms proposed for user labeling and core dropping with the *user maxmin fairness* criterion of Section III.

Theorem 1: The bandwidth allocation resulting from the user labeling of (14) and the core behavior of (21) is *user maxmin fair*.

One remaining challenge is the estimation of the *fair label* L_{fair}. For scalability reasons, L_{fair} should be estimated without storing any per-user or per-flow information at the core nodes. Different solutions to the problem of estimating L_{fair}

without core state have been proposed in [17], [18], [20], [21]. The algorithm that we have used in this paper is the one proposed in [17].

The UFQ architecture resulting from the *user labeling*, *ingress label control* and *core dropping* algorithms is described in pseudocode in Algorithm 1 and illustrated in Figure 4.

Algorithm 1 UFQ pseudocode

User labeling:
on receiving packet k
  $r_i = (1 - e^{-(T_k/K)})\frac{l_k}{T_k} + e^{-(T_k/K)} r_i$
  $L_k = \frac{r_i}{W_i}$
  write_label(L_k)

Ingress Label Control:
on receiving packet k
  read_label(L_k)
  $r_u = (1 - e^{-(T_k/K)})\frac{l_k}{T_k} + e^{-(T_k/K)} r_u$
  $L_k = max\left(L_k, \frac{(1-e^{-(\frac{l_k}{r_u \cdot K})})r_u}{1-e^{-(\frac{l_k}{r_u \cdot K})} \cdot S_u}\right)$
  $S_u = (1 - e^{-(\frac{l_k}{r_u \cdot K})})\frac{r_u}{L_k} + e^{-(\frac{l_k}{r_u \cdot K})} \cdot S_u$
  write_label(L_k)

Core dropping:
on receiving packet k
  read_label(L_k)
  estimate L_{fair}
  $prob = max(0, 1 - \frac{L_{fair}}{L_k})$
  if $prob >$ unif_rand$(0,1)$ **then**
    drop(packet k)
  else
    enqueue(packet k)
  end if
  if $prob > 0$ **then**
    write_label(L_{fair})
  end if

D. Ingress Label Control and Excess Service

The service data received by a user who is labeling his packets as in (14) during a time interval T is:

$$F = T \sum_{i \in I_u} r_i = T \sum_{i \in I_u} W_i \cdot L^i_{fair} \tag{22}$$

where L^i_{fair} is the fair label of flow i's bottleneck and

$$\sum_{i \in I_u} W_i = 1 \tag{23}$$

F is the service to which a user is entitled. We call any amount above F the *excess service*.

The ingress label control presented in Section IV-B has been designed with the goal of avoiding that a user can obtain more service than he is entitled to. In this section we study how well the scheme we have proposed meets this goal.

We cannot study the above issue with full generality, but we can analyze a simplified situation where the *fair label* L_{fair} of all links is held fixed. In addition, we assume that when a packet arrives a fraction of that packet equal to the flow's forwarding probability is transmitted.

Theorem 2 gives an upper bound to the excess service received by a user in this idealized setting. This bound is independent of the arrival process, the incoming labels and the time interval. The bound does depend crucially on the maximal rate R at which user's packets can arrive at the ingress (limited, for example, by the speed of the user's access link); the smaller this rate R the tighter the bound.

By bounding the excess service, we show that in the idealized setting the asymptotic throughput received by a user cannot exceed the throughput he is entitled to. Thus, users can only exploit the system over short time scales; the ingress label control limits their throughput effectively over long time scales.

Theorem 2: Consider a user sending n flows through n bottleneck links, all links with a constant *fair label* L^i_{fair}. Then, the excess service F_{excess} received by this user, that sends at a rate no larger than R, is bounded above by

$$F_{excess} < \left(\sum_{i \in I_u} L^i_{fair} \cdot W_i\right) \cdot \left(\frac{l_{max}}{L^{min}_{fair}} + K\left(2 + ln\frac{R}{L^{min}_{fair}}\right)\right) \tag{24}$$

where l_{max} represents the maximum length of a packet, I_u is the set of flows of the user, $L^{min}_{fair} = \min_{i \in I_u}(L^i_{fair})$ and

$$\sum_{i \in I_u} W_i = 1 \tag{25}$$

E. User Labeling and User Utility

The UFQ architecture is built around the assumption that users label their packets with $L_k = r_i^{send}/W_i$, where W_i is the weight of flow i in the user's utility function (*user labeling*, Equation 14). However, a user is allowed to label his packets with any label L_k with the only restriction of the ingress label control, which is much less restrictive on account of avoiding per-flow state at the ingress. A natural concern is whether a user can possibly benefit from labeling his packets with L_k different than r_i^{send}/W_i.

We cannot answer the above question with full generality, but we can analyze the same simplified situation as for Theorem 2 with the additional assumption that the sending rate of all flows is constant. Theorem 3 states that, in these conditions, a user sending n flows, all of them suffering from congestion, maximizes his utility when labeling his packets with $L_k = r_i^{send}/W_i$.

We conclude that, considering that all flows are susceptible to suffer from congestion, it is reasonable to assume that users label their packets with $L_k = r_i^{send}/W_i$, using an accurate estimation of flow i's sending rate r_i^{send} such as the one in Equation 15.

Theorem 3: Consider a user sending n flows at a constant bit rate r_i through n bottleneck links, all links with a constant *fair label*. Then, the user maximizes his utility when labeling the packets of flow i with $L_k = r_i^{send}/W_i$, where W_i is flow i's weight in the user's utility function.

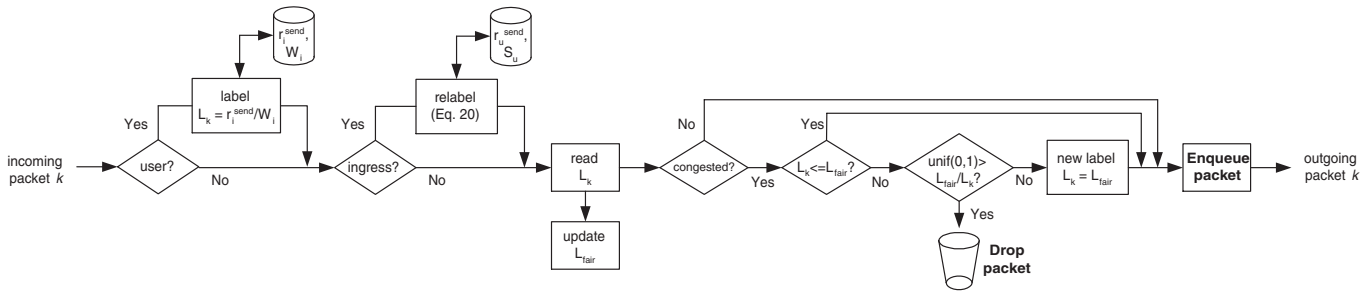

Fig. 4. UFQ algorithm.

V. Simulations

In this section we evaluate our algorithm by simulation. To provide some context, we compare UFQ's performance to three additional mechanisms for sharing resources: *FQ per-user*, *FQ per-flow* and TCP.

Fair Queuing (FQ) [5] is a queuing algorithm that aims at equally distributing the bandwidth of a link among traffic aggregates. In the *FQ per-user* approach, FQ is configured such that each traffic aggregate corresponds to the traffic generated by one user, in such a way that the link's bandwidth is divided equally among the users sending through this link. The *FQ per-user* approach is the basis of the *User Share Differentiation* (USD) architecture [22]. Note that USD, in contrast to UFQ, stores information for each user at core nodes, which results in a higher complexity.

In the *FQ per-flow* approach, FQ is configured such that each traffic aggregate corresponds to one flow, in such a way that the link's bandwidth is divided equally among the flows sending through the link. [17], [18], [19], [20] provide *FQ per-flow* without the need of storing per-flow state in core nodes.

The mechanism used for bandwidth sharing in the current Internet is the TCP protocol, which relies on the responsive behavior of the end-hosts to congestion. Active queue management schemes such as RED (Random Early Discarding) [23] aim at smoothening the behavior of TCP by providing early notification of congestion. Unless stated otherwise, simulation results for TCP will be provided using RED in the routers.

We have examined the behavior of UFQ under a variety of conditions, comparing its bandwidth allocations with the theoretical *user maxmin fair* (UMMF) distributions. Simulations V-A to V-E study the features of *user maxmin fairness* for bandwidth sharing and compares them with the other mechanisms. These simulations have been performed with constant bit rate UDP sources. Simulations V-F and V-G study some features of the UFQ mechanism. Finally, the support of different traffic models (TCP and ON-OFF sources) is analyzed in simulation V-H.

All simulations have been performed in ns-2 [24]. Unless otherwise specified, we use the following parameters for the simulations. All the flows of a user have the same weight. Each output link has a capacity of 10 Mbps, a latency of 1 ms and a buffer of 64 KB. In the RED case, the first threshold is set to 16 KB and the second to 32 KB. The fair queuing (FQ) discipline is implemented with the weighted round-robin (WRR) scheduler. The packet size is set to 1000 bytes.

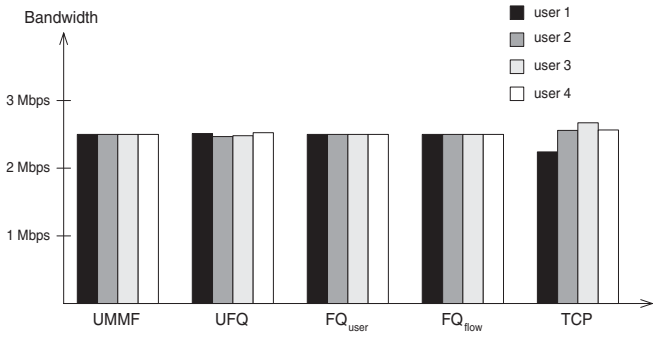

Fig. 5. Single Flow - One Link.

A. Single Flow - One Link

Figure 5 shows the resulting bandwidth distribution with the various mechanisms when a 10 Mbps link is congested by four users sending a single flow each. It can be seen that the *user maxmin fairness* criterion (UMMF) distributes the link's bandwidth among the four users equally. The results provided by the UFQ mechanism are very close to the ideal results (UMMF). Note that in this simple scenario the other three approaches (*FQ per-user*, *FQ per-flow* and TCP) distribute the link's bandwidth similarly to UFQ.

B. Single Flow - Several Links

Figure 6 shows the bandwidth distribution when four users are sending one flow through a different number of equally congested links (see Figure 7). Also in this case, UFQ distributes the bandwidth such that the four users receive the same throughput.

FQ per-user and *FQ per-flow* distribute the bandwidth in the same way as UFQ. In contrast, TCP gives a better treatment to those flows with a lower number of hops.

C. Several Flows - One Link

Figure 8 shows the bandwidth distribution when one link is congested by four users transmitting each a different number of flows (user i transmits i flows). With UFQ all users receive the same throughput.

With *FQ per-user* the throughput distribution is the same as with UFQ. In contrast, *FQ per-flow* and TCP favor those users who are sending more flows, giving to each user a throughput proportional to his number of flows. This is because *FQ per-flow* and TCP distribute the bandwidth on a per-flow basis.

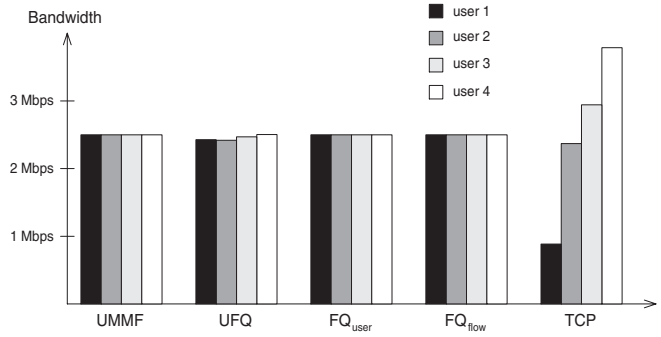

Fig. 6. Single Flow - Serveral Link.

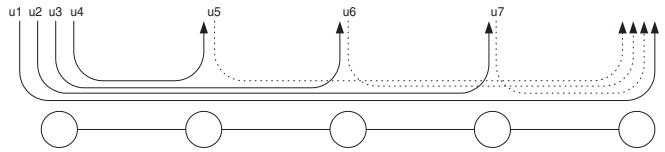

Fig. 7. Simulation scenario, Single Flow - Several Links.

D. Several Paths - Uniform Level of congestion

Figure 9 shows the bandwidth distribution for the case of users sending through different paths, when the level of congestion of all the links is the same (see Figure 10). UFQ provides all users with the same throughput.

In contrast to UFQ, the other approaches (*FQ per-user*, *FQ per-flow* and TCP) favor those users who are sending through more paths, giving them a throughput proportional to their number of paths. This is because these approaches distribute the bandwidth locally (either on a per-link basis – *FQ per-user*, on a per-flow basis – TCP, or both – *FQ per-flow*). The fact that UFQ works with overall network resources instead of locally is one of its key aspects, as compared to other existing approaches.

E. Several Paths - Heterogeneous Level of congestion

Figure 11 shows the bandwidth distribution for the case of users sending through different paths, when the level of congestion of the links is variable (see Figure 12).

In the results of Figure 11 it can be seen that, with UFQ, user i receives a larger throughput than user $i + 1$. This is because user i is sending his flows through less congested links (in average) than user $i + 1$.

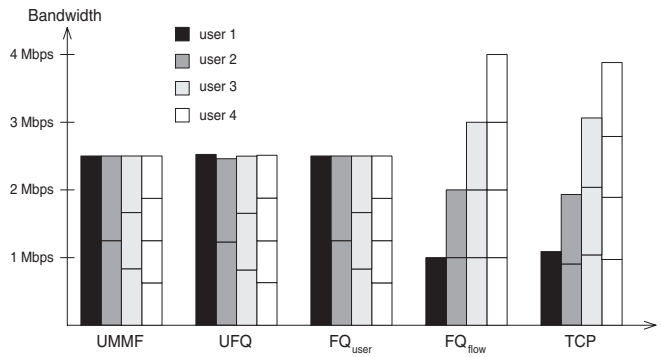

Fig. 8. Several Flows - One Link.

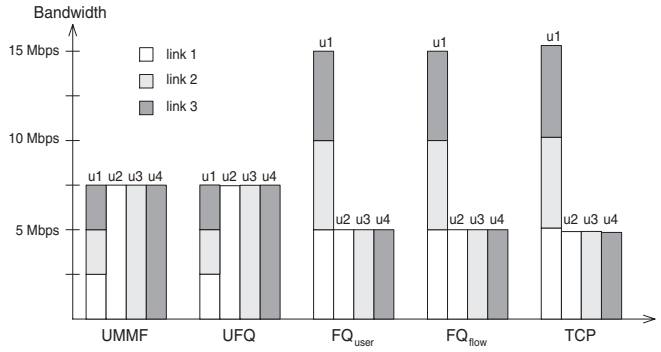

Fig. 9. Several Paths - Uniform Level of congestion.

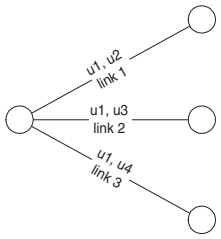

Fig. 10. Simulation scenario, Several Paths - Uniform Level of congestion.

With the other approaches (*FQ per-user*, *FQ per-flow* and TCP) user i also receives a larger throughput than user $i + 1$. However, with these approaches the difference between the throughputs is larger. The reason is the same as for the previous simulation: while these three approaches distribute the bandwidth on a local basis, UFQ takes into account the overall network resources. For example, with *FQ per-user*, *FQ per-flow* and TCP, bandwidth in link 1 is distributed such that all users receive 2.5 Mbps. Instead, UFQ gives four times more bandwidth in this link to user 4 (4.8 Mbps) than to user 1 (1.2 Mbps), on account of the fact that user 4 is sending only through this link, while user 1 is sending to three additional links.

F. Intra-user Differentiation

In UFQ, a user expresses the relative value of his flows with the use of weights. In order to study this feature we repeated the experiment of simulation V-C assigning different weights to the two flows of user 2: $W_1 = 0.33$ and $W_2 = 0.66$ (note that $W_1 + W_2 = 1$).

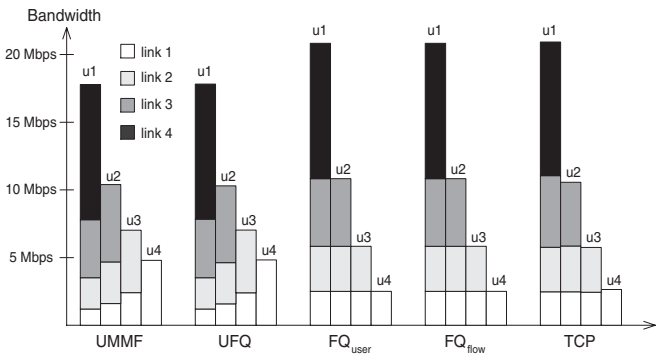

Fig. 11. Several Paths - Heterogeneous Level of congestion.

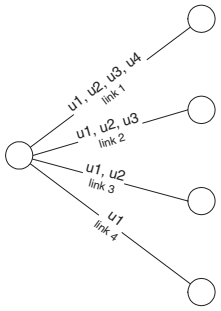

Fig. 12. Simulation scenario, Several Paths - Heterogeneous Level of congestion.

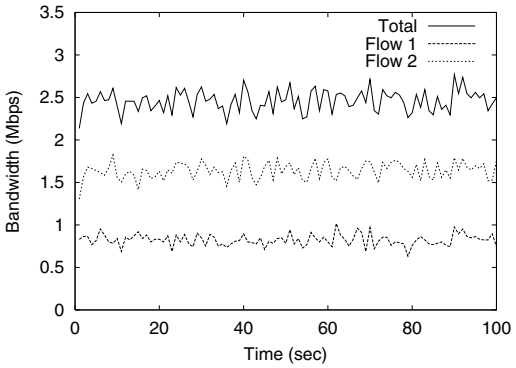

Fig. 13. Intra-user differentiation.

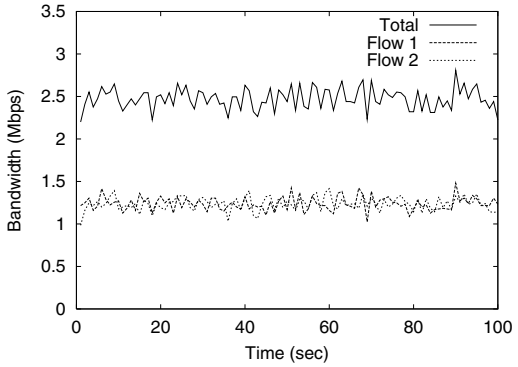

Fig. 14. Ingress label control.

	TCP		Bursty UDP	CBR UDP
	sources	RTT	avg. rate	avg. rate
user 1	-	-	-	10 Mbps
user 2	-	-	-	15 Mbps
user 3	-	-	15 Mbps	-
user 4	5	20 ms	10 Mbps	-
user 5	5	20 ms	-	10 Mbps
user 6	5	20 ms	-	-
user 7	10	50 ms	-	-
user 8	10	20 ms	-	-

TABLE I
SINGLE FLOW - ONE LINK

Figure 13 plots the receiving rates averaged over 1 second interval for flow 1, flow 2 and the total of user 2. It can be observed that the throughput received by flow 2 is twice as much as the received by flow 1, which matches the user's preferences.

G. Ingress label control

In order to assess the effectiveness of the ingress label control algorithm described in Section IV-B, we repeated the previous experiment but with the weights $W_1 = 3.33$ and $W_2 = 6.66$. Note that in this case $W_1 + W_2 = 10$, i.e. user 2 misbehaves and assigns lower labels than he should.

The results obtained from the above configuration are plotted in Figure 14. We can observe that the ingress control is effective since user 2 does not gain any excess service by misbehaving. In addition, the intra-user differentiation feature is lost.

H. Different Traffic Models

So far we have only considered constant bit rate UDP traffic. We now study the behavior of UFQ under different traffic models. We consider a 40 Mbps link congested by 8 users sending a mixture of constant bit rate UDP, bursty UDP and endless TCP traffic. The bursty UDP traffic consists of an aggregate of ON/OFF sources, with ON periods following a Pareto distribution (average 50 ms), OFF periods exponentially distributed (average 50 ms) and a sending rate in the ON period of 2 Mbps. Table I shows the characteristics of the traffic sent by each user. For those users sending both TCP and UDP traffic, weights are set to equally distribute the user's bandwidth between TCP and UDP.

Figure 15 shows the resulting bandwidth distribution according to the *user maxmin fairness* criterion (UMMF), UFQ scheduling, a standard FIFO router and a RED router.

From these results we conclude that the bandwidth received by a user with the UFQ architecture depends on the level of responsiveness of his traffic. All users sending non-responsive UDP traffic receive the same throughput, independent of the sending rate and level of burstiness. In contrast, the throughput received by TCP traffic depends on the level of responsiveness of this traffic, determined by the number of TCP sources and their RTT. However, from the results obtained it can be observed that TCP behaves reasonably well; in all cases TCP receives a throughput between 60% and 90% of its fair share rate, both when competing with UDP traffic of the same or a different user. Note that with FIFO and RED TCP traffic is starved.

The FBA-TCP architecture [25] has been proposed in the context of core stateless fair queuing for flows (specifically, within [17]) to improve the level of fairness between TCP and UDP. This is achieved by setting the maximum congestion window of TCP to the product of the RTT and L_{fair}. This approach could also be used within UFQ simply by adding the flow's weight W_i to this product.

VI. SUMMARY AND CONCLUSIONS

User Fairness aims at a fair distribution of network resources among users. The need for user fairness is motivated by the fact that the user is commonly the entity to which pricing schemes

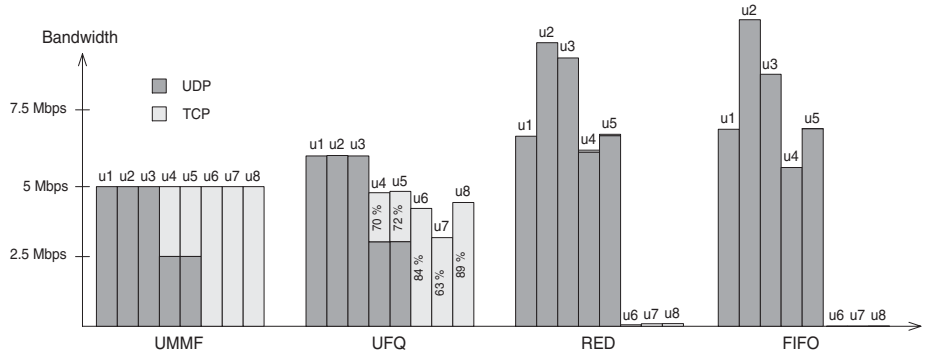

Fig. 15. Different Traffic Models - One Link.

apply; as a consequence, the user should also be the unit to which network resources are assigned.

However, while much effort has been invested in the past for the definition of fairness among flows, much less effort has been spent to address fairness among users. Our definition of *user maxmin fairness* in this paper tries to fill this gap.

Along with the *user maxmin fairness* criterion we have also proposed the *User Fair Queuing* (UFQ) architecture.

In UFQ, a user is allowed to assign any label values to his packets to indicate their relative priority. At the ingress, we have proposed an algorithm to control these labels assigned by the user. We have shown that the proposed label control does not allow the asymptotic throughput of a user to exceed its fair rate.

The fact that neither admission control nor signaling are required strongly contributes to the simplicity of UFQ. The fact that no per-user state is kept at core nodes makes the proposed architecture scalable.

The bandwidth distribution resulting from UFQ depends on the way users label their packets. In the paper we show that, if users label their packets in order to maximize their level of satisfaction or utility, then the resulting bandwidth allocation is *user maxmin fair*.

The performance of the proposed architecture has been validated via simulation. Simulation results show that the amount of network resources received by a user does not depend on the number of flows or their paths, and show a fairly good independence on the sources' responsiveness.

REFERENCES

[1] D. Bertsekas and R. Gallager, *Data Networks*, chapter 6, pp. 524–529, Prentice-Hall, 1987.

[2] F. P. Kelly, "Charging and rate control for elastic traffic," *European Transactions on Telecommunications*, vol. 8, no. 1, pp. 33–37, January 1997.

[3] R. Denda, A. Banchs, and W. Effelsberg, "The Fairness Challenge in Computer Networks," in *Proceedings of the 1st International Workshop on Quality of future Internet Services (QofIS 2000)*, Berlin, Germany, September 2000.

[4] V. Jacobson, "Congestion Avoidance and Control," *Computer Communication Review*, vol. 18, no. 4, pp. 314–329, August 1988.

[5] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queuing Algorithm," *Internetworking Research and Experience*, pp. 3–26, October 1990.

[6] S. Floyd and V. Jacobson, "Link Sharing and Resource Management Models for Packet Networks," *IEEE/ACM Transactions on Networking*, vol. 3, no. 4, pp. 365–386, August 1995.

[7] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: an Overview," RFC 1633, June 1994.

[8] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," RFC 2475, December 1998.

[9] S. Sato, K. Kobayashi, H. Pan, S. Tartarelli, and A. Banchs, "Configuration Rule and Performance Evaluation of Diffserv Parameters," in *Proceedings of the Seventeenth International Teletraffic Congress (ITC17)*, Salvador da Bahia, Brazil, September 2001.

[10] M. Brunner, A. Banchs, S. Tartarelli, and H. Pan, "A one-to-any Probabilistic Assured Rate Per-Domain Behavior for Differentiated Services," Internet draft, February 2001.

[11] A. Banchs and R. Denda, "A Scalable Share Differentiation Architecture for Elastic and Real-Time Traffic," in *Proceedings of the Eight IEEE/IFIP International Workshop on Quality of Service (IWQoS 2000)*, Pittsburg, PA, June 2000.

[12] A. Banchs, O. Leon, and S. Sallent, "The Olympic Service Model: Issues and Architecture," in *Proceedings of the 2nd International Workshop on Quality of future Internet Services (QofIS 2001)*, Coimbra, Portugal, September 2001.

[13] H. R. Varian, *Intermediate Microeconomics - A Modern Approach*, W. W. North & Company, New York/London, fifth edition, 1999.

[14] H. R. Varian, "Distributive Justice, Welfare Economics, and the Theory of Fairness," *Philosophy & Public Affairs*, vol. 4, no. 3, pp. 223–247, 1975.

[15] S. Shenker, "Fundamental Design Issues for the Future Internet," *IEEE Journal on Selected Areas in Communication*, vol. 13, no. 7, pp. 1176–1188, September 1995.

[16] L. Massoulie and J. Roberts, "Bandwidth Sharing: Objectives and Algorithms," in *Proceedings of IEEE INFOCOM '99*, New York, NY, March 1999.

[17] I. Stoica, S. Shenker, and H. Zhang, "Core-Stateless Fair Queueing: Achieving Approximately Fair Bandwidth Allocations in High Speed Networks," in *Proceedings of ACM SIGCOMM '98*, Vancouver, Canada, August 1998, pp. 118–130.

[18] Z. Cao, Z. Wang, and E. Zegura, "Rainbow Fair Queueing: Fair Banwdith Sharing Without Per-Flow State," in *Proceedings of IEEE INFOCOM 2000*, Tel-Aviv, Israel, March 2000.

[19] A. Clerget and W. Dabbous, "*T*UF: Tag-based Unified Fairness," in *Proceedings of IEEE INFOCOM 2001*, Anchorage, Alaska, April 2001.

[20] H. Zhu, A. Sang, and S. Li, "Weighted Fair Bandwidth Sharing Using SCALE Technique," *Computer Communications Journal, Special Issue in QoS*, vol. 24, no. 1, January 2001.

[21] M. Nabeshima, T. Shimizu, and I. Yamasaki, "Fair Queuing with In/Out Bit in Core Stateless Networks," in *Proceedings of the Eight IEEE/IFIP International Workshop on Quality of Service (IWQoS 2000)*, Pittsburg, PA, June 2000.

[22] Z. Wang, "A Case for Proportional Fair Sharing," in *Proceedings of the Sixth IEEE/IFIP International Workshop on Quality of Service (IWQoS '98)*, Napa, CA, May 1998.

[23] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 1, pp. 397–413, August 1993.

[24] UCB/LBNL/VINT, "Network Simulator (ns), version 2," `http://www.isi.edu/nsnam/ns/`.

[25] R. Kapoor, C. Cassetti, and M. Gerla, "Core-Stateless Fair Bandwidth Allocation for TCP flows," in *Proceedings of IEEE ICC 2001*, Helsinki, Finland, June 2001.