Fair Bandwidth Allocation for Multicasting in Networks with Discrete Feasible Set

Saswati Sarkar, *Member*, *IEEE*, and Leandros Tassiulas

Abstract—We study fairness in allocating bandwidth for loss-tolerant real-time multicast applications. We assume that the traffic is encoded in several layers so that the network can adapt to the available bandwidth and receiver processing capabilities by varying the number of layers delivered. We consider the case where receivers cannot subscribe to fractional layers. Therefore, the network can allocate only a discrete set of bandwidth to a receiver, whereas a continuous set of rates can be allocated when receivers can subscribe to fractional layers. Fairness issues differ vastly in these two different cases. Computation of lexicographic optimal rate allocation becomes NP-hard in this case, while lexicographic optimal rate allocation is polynomial complexity computable when fractional layers can be allocated. Furthermore, maxmin fair rate vector may not exist in this case. We introduce a new notion of fairness, maximal fairness. Even though maximal fairness is a weaker notion of fairness, it has many intuitively appealing fairness properties. For example, it coincides with lexicographic optimality and maxmin fairness, when maxmin fair rate allocation exists. We propose a polynomial complexity algorithm for computation of maximally fair rates allocated to various source-destination pairs, which incidentally computes the maxmin fair rate allocation, when the latter exists.

Index Terms—Maximal fairness, multicast, discrete bandwidth allocation.

1 Introduction

Multicasting provides an efficient way of transmitting data from a sender to a group of receivers. Many real-time applications like teleconferencing, audio, and video broadcasting require communication within a group and, hence, multicast is the inherent mode of delivery in these applications. Congestion control is critically important for real-time applications because many of these applications consume significant bandwidth and tax the network resources severely. The users need to adapt to the available bandwidth of the network, which fluctuates with time. Therefore, real-time sources use a hierarchical or a layered coding scheme. In this approach, a signal is encoded into a number of layers that can be incrementally combined to provide progressive refinement. The receivers adapt to congestion by adding and dropping layers. As each layer is added, there is an improvement in the quality of the received signal and additional bandwidth is needed to transport the combined stream. When a layer is dropped, the associated reduction in bandwidth causes a graceful degradation in the quality of reception. Every receiver would like to receive as many layers as possible. Networks, however, have limited resources and delivering a large number of layers would cause acute congestion. Fairness of resource allocation becomes important in this situation.

Multicasting poses some specific fairness challenges. The fairness objective is that every receiver receives service at a rate commensurate with its capabilities and the capacity of the path leading to it from the source. In the absence of additional provisions, a single rate of transmission per session may either overwhelm the slow receivers or starve the fast ones. Multirate transmission should be used to counter network heterogeneity. Multirate transmission allows different receivers of the same session to subscribe to different number of layers. The source transmits at a rate matching the fastest of its receivers. At every link, the transmission rate of a session is equal to that of the fastest session receiver downstream of the link.

The bandwidth of each layer is often predetermined and cannot be changed according to the needs of the network. A receiver either fully receives a layer or does not receive the layer at all. It cannot partially subscribe to a layer, unlike the scenario in [15]. While a continuous set of rates can be allocated when receivers subscribe to fractional layers [15], the network can only allocate a discrete set of rates to the receivers when receivers cannot subscribe to fractional layers. We study fair allocation of service rates under this additional constraint. As it turns out, fairness in a discrete set is vastly different from that in a continuous set.

Maxmin fairness [2] is a well-accepted notion of fairness. A rate allocation is maxmin fair if no receiver can be allocated a higher rate without reducing the rate of another receiver having equal or lower rate. We demonstrate later in this paper that a maxmin fair rate allocation may not exist in a discrete set. But, a maxmin fair rate allocation always exists in a continuous set [15]. Lexicographic optimality is another notion of fairness. A lexicographically optimal rate vector is one that maximizes its minimum component in a feasible set, subject to this maximization, it maximizes the

- *S. Sarkar is with the Department of Electrical and Systems Engineering, Univesity of Pennsylvania, 200 S. 33rd St., Philadelphia, PA 19104. E-mail: swati@ee.upenn.edu.*
- *L. Tassiulas is with the Department of Electrical Engineering, University of Maryland, College Park, MD 20742-3285. E-mail: leandros@isr.umd.edu.*

Manuscript received 16 May 2002; revised 3 June 2003; accepted 31 Oct. 2003.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 116564.

second minimum, etc.[1] Lexicographically optimal rate allocation exists in a discrete set, but, as we prove later, its computation is an NP-hard problem. Lexicographically optimal rate allocation is, however, identical to the maxmin fair rate allocation and is thus computable in polynomial complexity in a continuous set [12], [15]. We can instead compute a maximally fair rate allocation in a discrete set. A rate allocation is maximally fair if no other rate allocation is "fairer." (We describe the concept of relative fairness between two rate allocations in Definition 2.) That is, if a rate allocation is maximally fair, then, to increase the rate of a receiver s, we must lower that of another receiver j to a value less than the new rate of s and, thus, must be "unfair" or "less fair" to receiver j. Maximal fairness is a weaker notion of fairness as compared to maxmin fairness and lexicographic optimality. But, maximal fairness has various desirable fairness properties, e.g., it coincides with maxmin fairness and lexicographic optimality when maxmin fair rate allocation exists. We discuss other desirable fairness properties of maximally fair allocations later. In a nutshell, maximal fairness is probably the best we can achieve in the discrete case in view of the nonexistence of maxmin fair rate allocation and the computational complexity of lexicographically optimal rate allocation. We will present a polynomial complexity algorithm for computing maximally fair rate allocation in this paper. This algorithm yields a maxmin fair rate vector, if it exists. Our algorithm for computation of maximally fair allocation does not assume any property specific to the Internet or ATM. So, it is applicable in a very general scenario. Keeping in mind ATM networks, we have incorporated minimum rate requirements and maximum rate constraints in our model.

Legout et al. [7] have proposed several bandwidth allocation policies for multicast sessions. The authors assume continuous feasible sets and specify how to distribute the bandwidth in a single link. Now, definition with respect to a single link leads to inefficient utilization of overall bandwidth in the network case. We consider notions of fairness for networks. We review related work for discrete bandwidth layer allocation briefly. Several congestion control protocols have been proposed for multilayer multicast, e.g., RLM (Receiver-driven Layered Multicast) [10], LVMR (Layered Video Multicast with Retransmissions) [8], [9], RLC (Receiver-driven Layered Congestion control) [16], PLM (Packet-pair receiver-driven cumulative Layered Multicast) [6]. These protocols dynamically adapt the number of layers allocated to receivers based on the congestion in the network. The performances of these protocols have been evaluated through simulation. Our research establishes a theoretical framework for congestion control in multirate multicast networks and is, thus, complementary to the existing approaches. We assume that a receiver cannot decode a higher layer if it has not received all the lower layers ("cumulative layering"). This assumption has also been made in several other references [6], [8], [9], [10], [16]. Byers et al. [3] proposed a scheme for choosing the layers so as to track any desired bandwidth allocation closely when the cumulative layering constraint is relaxed. The disadvantage of a noncumulative layering scheme is that it leads to inefficient utilization of link bandwidth. Rubenstein et al. also point out that a maxmin fair rate allocation may not exist for discrete bandwidth layers and suggest a remedial policy of coordinated random add and drop of the highest layer for various receivers [12]. This attains long-term rates close to the maxmin fair allocation. This oscillation is, however, likely to produce perceptually annoying distortion. The resulting perceptual quality may even be worse than not subscribing to the highest layer at all. Besides, this random add and drop of the highest layer generally leads to underutilization of link capacity. We propose the use of maximally fair rate allocation as an alternative.

This paper is organized as follows: We describe the network model in Section 2. We develop notions of fairness in the discrete case in Section 3. We present fairness properties that apply to any discrete feasible set in Section 3. We present fairness properties specific to the layer allocation problem in Section 4.1. We present an algorithm for computation of the maximally fair rates in Section 4.2. We identify some directions for future research in the concluding section, Section 5. Unless otherwise stated, the proofs can be found in the Appendix (which can be found on the Computer Society Digital Library at http://computer.org/tc/archives.htm).

2 Network Model

Consider a network with an arbitrary topology and N multicast sessions. A multicast session is identified by the pair (v, U), where v is the source of the session and U is the group of destinations. The traffic from node v is transported across a predefined multicast tree to nodes in U. The tree can be established either during the connection establishment phase for connection oriented networks or can be established using well-known multicast routing protocols, like DVMRP [4], CBT [1], [11], etc., for connectionless networks like the internet. The symbols introduced in this section have been summarized in Table 1.

Every source destination pair of a session is called a virtual session. For example, if a session n has source v and destination set $\mathcal{U}$, where $\mathcal{U} = \{u_1, \dots, u_t\}$, then n has t virtual sessions, $(v, u_1), \dots, (v, u_t)$. Refer to Fig. 1 for an example of sessions and virtual sessions. To ensure fairness in a multirate network, we need to consider fair layer allocation for the virtual sessions separately, instead of considering only the layer allocations for the overall session. Every virtual session (source-destination pair) has a minimum and a maximum layer requirement.[2] We assume that every layer consumes the same amount of bandwidth, b units, independent of the session. This assumption simplifies the mathematical framework without altering the nature of the results.

A layer allocation for the virtual sessions is said to be feasible if the number of layers for every virtual session is between the minimum and the maximum possible number of layers for the virtual session. Besides, if session n

1. Maxmin fairness and lexicographic optimality are defined formally in Definitions 3a, 3b, and 5, respectively.

2. The absence of these requirements can be incorporated by choosing the minimum layer requirement as 0 and the maximum layer requirement as ∞.

TABLE 1
Summary of Symbols Used throughout the Paper

Symbol	Meaning
N	Number of sessions
M	Number of Virtual Sessions
$\chi(j)$	Session of virtual session j
$n(l)$	Set of sessions passing through link l
$m(k,l)$	Set of virtual sessions of session k passing through link l
C_l	Capacity of link l
γ_j	Number of layers allocated to virtual session j
$\vec{\gamma}$	Layer allocation vector, $\vec{\gamma} = (\gamma_1, \gamma_2, \ldots, \gamma_M)$
Γ_{il}	Number of layers allocated to session i in link l, (Maximum of the number of layers allocated to virtual sessions of session i traversing link l), $\Gamma_{il} = \max_{j\in m(i,l)} \gamma_j$
b	Bandwidth of a layer
r_j	Bandwidth allocated to virtual session j, $r_j = b\gamma_j$
λ_{il}	Bandwidth allocated to session i in link l, (Maximum of the bandwidth allocated to virtual sessions of session i traversing link l), $\lambda_{il} = \max_{j\in m(i,l)} r_j$
ι_i	Minimum layer requirement of virtual session i
ι_{il}	Minimum layer required by session i in link l, $\iota_{il} = \max_{j\in m(i,l)} \iota_j$
μ_i	Minimum bandwidth requirement of virtual session i, $\mu_i = b\iota_i$
μ_{il}	Minimum bandwidth required by session i in link l, $\mu_{il} = \max_{j\in m(i,l)} \mu_j$

corresponds to virtual sessions $m_1, \ldots, m_t$ in link l, then the bandwidth consumed by session n in link l is the maximum of the bandwidth allocated to the virtual sessions $m_1, \ldots, m_t$. The total bandwidth consumed by all sessions traversing through link l cannot exceed the capacity of link l. We next define this formally. Let p_j denote the maximum number of layers of virtual session j.

Definition 1 (Feasability Condition). *A layer allocation vector $\vec{\gamma}$ is feasible if*

1. *γ_j is an integer for all virtual sessions j,*
2. *$\iota_j \leq \gamma_j \leq p_j$, $\forall$ j, $p_j \geq \iota_j \geq 0$,*
3. *total bandwidth consumed by sessions traversing through link l does not exceed the capacity of link l, i.e., $b\sum_{i\in n(l)} \Gamma_{il} \leq C_l$ (Capacity condition).*

A rate allocation vector $\vec{r}$ is feasible if the corresponding layer allocation vector $\vec{\gamma}^r = (\gamma_1^r, \ldots, \gamma_M^r)$, defined as $\gamma_j = r_j/b$ is feasible. The feasibility conditions for rate and layer allocation vectors are equivalent and there is a one-to-one correspondence between the set of the feasible rate and layer allocation vectors.

Henceforth, we shall ignore the maximum layer constraints. This does not cause any loss in generality because the maximum layer constraints can be incorporated by adding artificial links between the receivers with the maximum layer constraints and the rest of the network. The capacity of such an artificial link is equal to the bandwidth consumed by the maximum number of layers of the respective receiver. As an example, consider the network shown in Fig. 1. Receiver u_1 can subscribe to 10 layers at most. If there were an additonal link l of bandwidth 5 units between e_4 and receiver u_1,, then receiver u_1 cannot receive more than 10 layers because of the constraint in link l. Thus, the augmentation of the network has the same effect as considering maximum bandwidth constraints separately. The size of the augmented network is comparable to that of the given network. So, the complexity of any algorithm for computation of fair layer allocation in a network should remain the same, even if we use the augmented network.

Finally, we would like to mention that end-users are concerned with the quality of reception. It is difficult to characterize the quality of reception as a mathematical function of the bandwidth, but the quality of reception improves with an increase in the bandwidth allocated to an end-user. We would focus on the allocation of bandwidth to the users. This involves: 1) deciding the number of layers to be allocated to each receiver and 2) delivering the desired number of layers. We consider the first part in this paper. We describe several notions of fairness in the next section and discuss how to compute the layer allocations as per these notions. The source can transmit each layer on a separate multicast group. Once the fair layer allocation is computed, the receivers subscribe to the appropriate multicast groups.

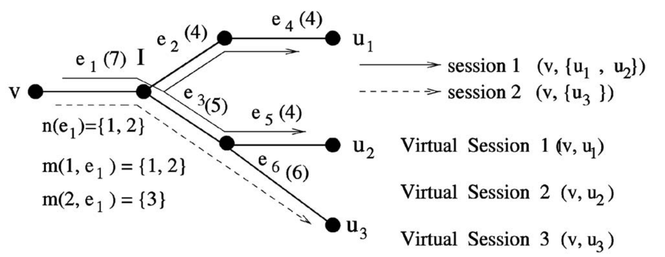

Fig. 1. This figure illustrates the concept of sessions and virtual sessions and shows some sample feasibility constraints, $n(e_i)$ and $m(i, e_j)$s. Session 1 includes virtual sessions, $1, 2$. Session 2 includes virtual session 3. The numbers in brackets, (), denote the capacities of the respective links. Every layer consumes 0.5 units of bandwidth. The capacity constraint for link e_1 is $0.5(\Gamma_{1e_1} + \Gamma_{2e_1}) \leq 7$ and that for link e_3 is $0.5(\Gamma_{1e_3} + \Gamma_{2e_3}) \leq 5$. Here, $\Gamma_{1e_1} = \max(\gamma_1, \gamma_2)$, $\Gamma_{2e_1} = \Gamma_{2e_3} = \gamma_3$, $\Gamma_{1e_3} = \gamma_2$. Equivalently, the capacity constraint for link e_1 is $\lambda_{1e_1} + \lambda_{2e_1} \leq 7$ and that for link e_3 is $\lambda_{1e_3} + \lambda_{2e_3} \leq 5$. Here, $\lambda_{1e_1} = \max(r_1, r_2)$, $\lambda_{2e_1} = \lambda_{2e_3} = r_3$, $\lambda_{1e_3} = r_2$. The minimum and maximum layer constraints are $8 \leq \gamma_1 \leq 10$, $2 \leq \gamma_2 \leq \infty$, and $0 \leq \gamma_3 \leq 10$. Equivalently, $4 \leq r_1 \leq 5$, $1 \leq r_2 \leq \infty$, and $0 \leq r_3 \leq 5$.

3 Fairness in a Discrete Feasible Set

To the best of our knowledge, fairness in a discrete feasible set has not been studied before. First, we describe various existing notions of fairness and show that they are inadequate for a discrete feasible set. We subsequently motivate a new notion and present several appealing properties of this new notion in discrete feasible sets. All definitions and properties mentioned in this section hold for any arbitrary discrete feasible set in R^K, for any positive integer K, and are not specific to the feasible set of layer or rate allocation vectors. For simplicity, however, we mention layer allocation vectors explicitly and present examples for layer allocation vectors only.

We use the concept of *relative fairness* introduced in [13] to define maxmin fairness. A layer allocation vector $\vec{A}$ is *fairer* than another layer allocation vector $\vec{B}$, if, for every virtual session i that has a higher number of layers under $\vec{B}$ than under $\vec{A}$, there is some other virtual session j whose number of layers was already no more than that of i under $\vec{A}$ and has been decreased further by $\vec{B}$. A more formal definition of relative fairness follows.

Definition 2 (Relative Fairness). *A layer allocation vector $\vec{A}$ is* fairer *than another layer allocation vector $\vec{B}$ if*

- *$\vec{A} \neq \vec{B}$ and*
- *the existence of an i such that $A_i < B_i$ implies that there exists a j such that $A_j \leq A_i$ and $B_j < A_j$.*

In Fig. 1, layer allocation $(8, 5, 5)$ is fairer than $(8, 6, 4)$.

We first present Proposition 1, which we will use later in this section.

Proposition 1. *Consider two M-dimensional vectors $\vec{A}$ and $\vec{B}$. Vector $\vec{A}$ is fairer than $\vec{B}$ if and only if there exists a component s such that $A_s = \min_{j \in \tau} A_j$ and $A_s > B_s$, where $\tau = \{j : A_j \neq B_j\}$, A_j and B_j are the jth components of $\vec{A}$ and $\vec{B}$, respectively.*

Proposition 1 has been proven in [13]. Proposition 1 gives a necessary and sufficient condition for $\vec{A}$ to be fairer than $\vec{B}$. Ignore any component i such that $A_i = B_i$. Consider the components that have the minimum value among the rest of the components in $\vec{A}$ (note that there can be several such components). Then, the necessary and sufficient condition is that $A_s > B_s$ for one such minimum component s. For example, compare $\vec{A} = (8, 5, 5)$ and $\vec{B} = (8, 6, 4)$ in Fig. 1. We ignore component 1. Components 2 and 3 both have minimum value, 5, among the rest and component 3 satisfies the condition.

Definition 3a (Maxmin fairness). *A feasible layer allocation vector $\vec{A}$ is maxmin fair if it is fairer than all other feasible layer allocation vectors.*

Definitions 2 and 3a can be combined to obtain the following equivalent definition of a maxmin fair layer allocation vector, which happens to be the classical definition of maxmin fairness [2]:

Definition 3b (Maxmin fairness). *A feasible layer allocation vector $\vec{A}$ is maxmin fair if it satisfies the following property with respect to any other feasible layer allocation vector $\vec{B}$: If there exists i such that $B_i > A_i$, then there exists j such that $A_j \leq A_i$ and $B_j < A_j$.*

In the network of Fig. 1, the maxmin fair layer allocation vector is $(8, 5, 5)$. It is easy to check that this layer allocation vector is feasible. Now, we explain why this layer allocation vector is maxmin fair. It is not possible to increase γ_1 above 8 because of the capacity constraint of link e_2. Any increase in γ_2 (γ_3) will cause a decrease in γ_3 (γ_2) because of the capacity constraint of link e_3, and leads to $(8, 5 + x, 5 - y)$ $((8, 5 - x, 5 + y))$, where $x > 0, y > 0$. Clearly, $(8, 5, 5)$ is fairer than $(8, 5 + x, 5 - y)$ or $(8, 5 - x, 5 + y)$, if $x > 0, y > 0$.

Definition 4 (Lexicographic Comparison). *Given an M-dimensional vector $\vec{V}$, we define its lexicographically ordered version $\hat{V}$ as follows: $\forall j \in \{1, \ldots, M\}$, $\exists$ k such that $\hat{V}_j = V_k$ and $\hat{V}_1 \leq \hat{V}_2 \leq \ldots \hat{V}_M$. In other words, components of $\hat{V}$ are an ordered version of those of $\vec{V}$. A layer allocation vector $\vec{A}$ is lexicographically greater than a layer allocation vector $\vec{B}$ if there exists i such that $\hat{A}_i > \hat{B}_i$ and $\hat{A}_j = \hat{B}_j$ if $j < i$. Vectors $\vec{A}$ and $\vec{B}$ are lexicographically equal if $\hat{A} = \hat{B}$, i.e., if $\hat{A}_i = \hat{B}_i$, $\forall i \in \{1, \ldots, M\}$. Vector $\vec{A}$ is lexicographically less than $\vec{B}$ if $\vec{B}$ is lexicographically greater than $\vec{A}$.*

Definition 5 (Lexicographic Optimality). *A feasible layer allocation is lexicographically optimal if it is lexicographically greater than or equal to every feasible layer allocation.*

In the network of Fig. 2, a lexicographically optimal layer allocation vector is $(1, 0, 3, 3)$. Clearly, this allocation is feasible. The lexicographically ordered version of this vector is $(0, 1, 3, 3)$. The minimum component of every feasible layer allocation vector must be 0 since two sessions traverse link e_1, which has bandwidth 1. The second minimum component must be 1 or less for the same reason. Since two sessions traverse link e_3, which has bandwidth 6, the third minimum must be 3 or less. If the third minimum is at the largest possible value, which is 3, the fourth minimum is also 3 or less. In this simple example, the lexicographically optimal allocation can be computed using

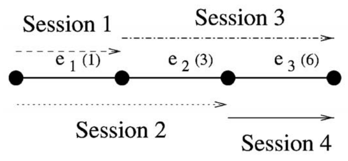

Fig. 2. This figure shows an example network which is used in the text to illustrate the difference between the three notions of fairness: maxmin fairness, lexicographic optimality, and maximal fairness. The sources emit unit bandwidth layers.

the argument described above. The computation is NP-hard in general, as we discuss later.

We discuss why maxmin fairness and lexicographic optimality are good notions of fairness in general. A simple fairness objective can be to allocate equal bandwidth to all end-users. This, however, leads to underutilization of bandwidth in the links. If equal distribution is not an issue, then the spare bandwidth can be used to increase the bandwidth allocated to a user without decreasing that of any other user. Consider the network shown in Fig. 2 for an example. Assume that every layer consumes 0.5 units of bandwidth. An equal distribution leads to the layer allocation $(1, 1, 1, 1)$, while $(1, 1, 5, 7)$ is a feasible layer allocation as well. This motivates the notion of maxmin fairness. Note that a feasible layer allocation vector is maxmin fair if it is not possible to maintain feasibility and increase the number of layers of a virtual session without decreasing the number of layers of any other virtual session that has equal or fewer number of layers. Layer allocation $(1, 1, 5, 7)$, rather than $(1, 1, 1, 1)$ is maxmin fair in the previous example. 1) If the paths for two end-users are identical, maxmin fairness mandates that the users be allocated equal number of layers. 2) A maxmin fair vector is pareto optimal by definition (i.e., a component can be increased and feasibility be maintained only by decreasing some other component). It is not always possible to satisfy both 1) and 2) in discrete feasible sets. For example, consider a network with one link with capacity 1 unit and two users traversing the link. Let every layer consume 1 unit of bandwidth. The only feasible allocation that gives an equal number of layers to both these end-users is $(0, 0)$, which is not pareto-optimal. Lexicographic optimality maintains 2) and attains 1) within a discrete approximation. The discrete approximation is in the following sense: A layer allocation is lexicographically optimal if its smallest component is the largest among the smallest components of all feasible layer allocations, subject to it having the largest second smallest component, and so on. In the single-link example, the allocations $(0, 1)$ and $(1, 0)$ are both lexicographically optimal.

Lemma 1. *If $\vec{A}$ is fairer than $\vec{B}$, then $\vec{A}$ is lexicographically greater than $\vec{B}$. The converse is not true.*

We prove the first part in the Appendix (which can be found on the Computer Society Digital Library at http://computer.org/tc/archives.htm). A counterexample shows that the converse is not true. In Fig. 2, applying the definitions of relative fairness and lexicographical ordering, neither of the layer allocations $(1, 0, 3, 3)$ and $(0, 1, 2, 4)$ is fairer than the other, but the first is lexicographically greater than the second.

We distinguish between the notions of maxmin fairness and lexicographic optimality. Maxmin fairness is optimal with respect to relative fairness, while lexicographic optimality optimizes with respect to lexicographic comparison. Lemma 1 shows that relative fairness is stronger than lexicographic comparison. Thus, maxmin fairness is stronger than lexicographic optimality. This is further enforced by the following observations: In general, if we consider finite dimensional vectors, with the feasible set closed and bounded, a lexicographically optimal vector always exists, but a maxmin fair vector may not exist. But, as Lemma 2 proves, if a maxmin fair vector exists, it is lexicographically optimal.

Lemma 2. *If a vector $\vec{A}$ is maxmin fair in a feasible set, then it is lexicographically optimal in the same feasible set.*

The notions of maxmin fairness and lexicographic optimality are inadequate for a discrete feasible set. This is because a maxmin fair layer allocation may not exist in general. Refer to Fig. 2 for more insight. There are only a finite number of feasible layer allocations. For any of these, it is possible to maintain feasibility and increase the number of layers of one session without decreasing the number of layers of another session that has an equal or lower number of layers. For example, consider the allocation $(1, 0, 3, 3)$. It is possible to increase the number of layers of session 2, by reducing that of session 1. Session 1, however, has a higher number of layers than session 2 in this allocation. Thus, there is no maxmin fair layer allocation. Lexicographically optimal allocation exists, e.g., $(1, 0, 3, 3)$ is the lexicographically optimal allocation in Fig. 2, but its computation is an NP-hard problem in many feasible sets. We will prove this result in the Appendix (which can be found on the Computer Society Digital Library at http://computer.org/tc/archives.htm) for the layer allocation problem. Incidentally, if the feasible set were continuous, the maxmin fair rate allocation vector would always exist and would, hence, be lexicographically optimal (Lemma 2). The definitions of maxmin fairness and lexicographic optimality can be used interchangeably in this case. In the discrete case, however, we need to distinguish between the two, but neither is adequate for reasons discussed before. We introduce a different notion of fairness, *maximal fairness*.

Definition 6 (Maximal Fairness). *A feasible layer allocation $\vec{A}$ is maximally fair if no other feasible layer allocation is fairer than $\vec{A}$.*

In Fig. 2, the layer allocation vectors $(1, 0, 3, 3)$, $(0, 1, 2, 4)$ are maximally fair. If any of the components is increased in either vector, then another component must be decreased to a value less than the new value of the component that has been increased. This precludes the existence of a layer allocation that is fairer than either of these and, thus, these layer allocations are maximally fair. We present an algorithm for computing maximally fair allocations in arbitrary networks in Section 4.2.

Lemma 3. *If there exists a maxmin fair vector, then it is the only maximally fair vector.*

Lemma 4. *If a layer allocation vector is lexicographically optimal, then it is maximally fair. The converse is not true.*

We prove the forward part in the Appendix (which can be found on the Computer Society Digital Library at http://computer.org/tc/archives.htm). We show the converse using a counter example. In Fig. 2, both $(1, 0, 3, 3)$ and $(0, 1, 2, 4)$ are maximally fair. The second layer allocation is, howeve,r not lexicographically optimal as it is lexicographically less than the first.

We now distinguish maximal fairness from maxmin fairness and lexicographic optimality. Lemmas 2 and 3 show that the three notions are one and the same if maxmin fair layer allocation exists. A maxmin fair layer allocation may not exist in a discrete feasible set because, owing to the discreteness constraint, bandwidth cannot be distributed equally among sessions traversing paths of the same capacity and congestion level. This precludes the existence of any single layer allocation that is fairer than the others. In this case, lexicographic optimality distributes the bandwidth so as to maximize the minimum component, and subject to this maximization maximizes the second minimum component, etc. Maximal fairness distributes the bandwidth to ensure a weaker condition: If bandwidth of any component i is increased, then the bandwidth of some other component must be decreased to a value less than that of the new value of component i. Unlike a maxmin fair allocation, a lexicographically optimal allocation $\vec{A}$ need not be fairer than all other allocations, but, at the same time, none of the other allocations is fairer than $\vec{A}$. The same observation holds for maximally fair layer allocations. Unlike lexicographic optimality, maximal fairness does not provide any guarantee of the maximality of the minimum, second minimum, etc. in the overall feasible set. There is, however, a guarantee for a certain pairwise ordering between the minimum components of a maximally fair allocation and any other layer allocation. We explain this in the next paragraph (property 2)). Lemma 4 further demonstrates the difference between lexicographic optimality and maxmimal fairness. Fig. 2 can be used to illustrate the difference between the three notions. Summarizing the observations made before, this network does not have a maxmin fair layer allocation, has two maximally fair layer allocations, $(1, 0, 3, 3)$ and $(0, 1, 2, 4)$, and one lexicographically optimal allocation, $(1, 0, 3, 3)$.

We argue that maximal fairness is a good notion of fairness. Maximal fairness has many intuitively appealing properties. Some of these properties hold for any arbitrary feasible set, whereas some others are particular to the resource allocation problem we are studying. We present the general properties in this section and thereby show that maximal fairness is a good notion of fairness.

1. A maximally fair vector is pareto optimal by definition. Thus, bandwidth of an end-user can be increased only by decreasing that of another end-user.
2. Consider two layer allocations, $\vec{A}$ and $\vec{B}$, where $\vec{A}$ is maximally fair. Ignore the end users who have equal number of layers under both. Let τ be the set of end users who have the minimum number of layers among the rest of the end-users under $\vec{B}$. Then, the number of layers allocated to one of the end users in τ, say s, by $\vec{A}$ is greater than that allocated by $\vec{B}$. This follows from the necessary and sufficient condition in Proposition 1 and the fact that $\vec{B}$ is not fairer than $\vec{A}$. We refer to this property as the min-order property. The min-order property guarantees that, between a maximally fair layer allocation and any other layer allocation (which may or may not be maximally fair), there exists a certain ordering for the number of layers allocated to an end-user who has the least bandwidth.
3. As Lemma 4 shows, a lexicographically optimal layer allocation belongs to the set of maximally fair layer allocations.
4. From Lemma 3, any algorithm for computation of a maximally fair layer allocation will yield a maxmin fair layer allocation, if one exists. This is interesting, in view of the observation that, even if a maxmin fair rate allocation exists in a discrete feasible set, it may be different from the maxmin fair rate allocation in the continuous feasible set, i.e., the feasible set defined by the capacity and the minimum rate constraints only. Refer to Fig. 3 for an example. This difference is because of the difference in the feasible set of rate vectors in the two cases. The feasible set of rate vectors for discrete bandwidth layers is a proper subset of the corresponding continuous feasible set. Hence, the maxmin fair rate vectors are different in some cases, even when the maxmin fair rate vector exists for discrete bandwidth layers. This means that an algorithm for computation of maxmin fair rates in the continuous feasible set may not compute the maxmin fair rate vector for the discrete bandwidth case, even when the latter exists. Thus, we have a strong incentive to compute the maximally fair rate allocation. We will present several other appealing fairness properties of a maximally fair vector that hold specifically for the layer allocation case.

We now summarize the differences in the fairness issues for continuous and discrete feasible sets as discussed in this section. Among the three notions of fairness presented in this paper, maxmin fairness is the strongest for both cases. If a maxmin fair vector exists, then it is lexicographically optimal and maximally fair. While the existence of a maxmin fair vector is guaranteed in a continuous feasible set, the same cannot be said of discrete feasible sets. In addition, the computation of a lexicographically optimum vector is NP-hard in a discrete feasible set, while this is computable with polynomial complexity in a continuous feasible set [15]. This motivates investigation of a weaker notion of fairness, maximal fairness, which has several intuitively appealing fairness properties. We explore the fairness issues further in the context of the layer allocation problem in the next section.

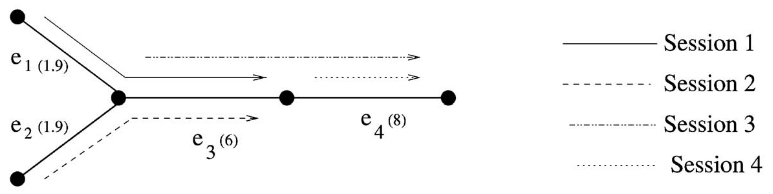

Fig. 3. This figure shows that the maxmin fair rate allocation in a discrete feasible set may be different from that in the corresponding continuous feasible set (continuous feasible set obtained by removing the constraint that the layer allocations must be integers), even if the discrete feasible set contains a maxmin fair allocation. The maxmin fair rate allocation is $(1.9, 1.9, 2.2, 5.8)$, assuming that the feasible set is continuous. Let every layer consume 1 unit of bandwidth. Now, $(1, 1, 4, 4)$ is the maxmin fair allocation in the discrete set. This figure also furnishes an example illustrating that it is not possible to obtain a maximally fair allocation in the discrete feasible set by rounding up the components of the maxmin fair allocation in the corresponding continuous feasible set. The intuitive layer allocation algorithm suggested in Section 4.2 would either yield a layer allocation vector $(1, 1, 3, 5)$ or $(1, 1, 2, 6)$ depending on the order in which the sessions are chosen for allocation of additional layers. But, $(1, 1, 4, 4)$ is the maxmin fair allocation and, hence, maximally fair as well. It is also fairer than both $(1, 1, 3, 5)$ and $(1, 1, 2, 6)$, neither of which is thus maximally fair. Also, the maximally fair layer allocation for component 3 is greater than $\lceil r_3^c \rceil$ and component 4 is less than $\lfloor r_4^c \rfloor$.

4 Fair Allocation of Discrete Bandwidth Layers

We have studied fairness properties which apply to any discrete feasible set in the previous section. These fairness properties apply to the layer allocation problem as well. In this section, we study the fairness problem specifically for the layer allocation case. We present fairness properties specific to this resource allocation scenario in the first subsection. We show that the computation of a lexicographically optimal layer allocation is NP-hard. Next, we present an intuitively appealing fairness property of a maximally fair layer allocation. In the next subsection, we present a polynomial complexity algorithm for computing the maximally fair layer allocation.

4.1 Resource Allocation Specific Fairness Properties

Theorem 1 (NP-hardness). *Computation of the lexicographically optimal layer allocation vector is NP-hard.*

Next, we present an intuitively appealing property of maximally fair allocation, which holds in the layer allocation problem we are studying. We first introduce the concept of pseudobottleneck links. This is analogous to the concept of bottleneck links for a continuous feasible set [15].

Definition 7 (Pseudobottleneck Link). *A link l is said to be* pseudobottlenecked *with respect to a virtual session k traversing l, for layer allocation vector $\vec{\gamma}$ and corresponding rate allocation $\vec{r}$ if the following conditions hold:*

1. $b\sum_{i\in n(l)}\Gamma_{il} > C_l - b$ *(equivalently,*
$$\sum_{i\in n(l)}\lambda_{il} > C_l - b),$$
2. $\gamma_k = \Gamma_{\chi(k)l}$ *(equivalently, $r_k = \lambda_{\chi(k)l}$),*
3. *If $\gamma_j > \iota_{\chi(j)l}$, where $\iota_{il} = \max_{j\in m(i,l)} \iota_j$, then $\gamma_j \leq \gamma_k + 1$ (equivalently, if $r_j > \mu_{\chi(j)l}$, then $r_j \leq r_k + b$).*

We now explain the different conditions. Condition 1 states that the capacity of link l is *almost* fully utilized, i.e., the difference between the capacity of the link and the sum of the rates allocated to the sessions traveling across the link must be less than the layer bandwidth b. Let virtual session k belong to session i. Condition 2 states that k has the maximum number of layers among all virtual sessions of session i traversing link l. Condition 3 states the following: Let the number of layers assigned to any other virtual session j traversing through link l be higher than that of a virtual session k by two or more layers. Let j belong to session p. Then, j's number of layers is less than or equal to the minimum number of layers required by session p in link l. Thus, if there is no minimum layer requirement, then the number of layers assigned to any other virtual session j traversing through link l would not exceed that of k by more than one.

Example 4.1. Consider the network in Fig. 1 for an example illustrating the concept of pseudobottleneck links. Let every layer consume 1 unit of bandwidth each, i.e., $b = 1$. Minimum layer constraints are $4, 2, 0$ for virtual sessions 1, 2, and 3, respectively. There are no maximum rate constraints. Consider the layer allocation $(4, 2, 3)$. Link e_1 is pseudobottlenecked w.r.t. virtual sessions $1, 3$ and link e_3 is pseudobottlenecked w.r.t. virtual session 2. Now, consider the layer allocation vector $(4, 4, 1)$. Virtual session 3 does not have a pseudobottleneck link. This is because it traverses through links e_1, e_3, and e_6. Virtual session 2 traversing through link e_3 has three more layers than virtual session 3 and does not have a minimum number of layers requirement. This violates pseudobottleneck condition 3. Total bandwidth consumed by the sessions traversing through link e_1 is 5 units, but e_1's capacity is 7 units, which is 2 units more than the capacity utilized and $b = 1$. Similarly, link e_6's utilized capacity is 1 unit, but its actual capacity is 6 units. This violates the pseudobottleneck condition 1. Link e_2 (e_3) is, however, pseudobottlenecked w.r.t. virtual session 1 (2).

Lemma 5 (pseudobottleneck lemma). *A feasible layer allocation vector is maximally fair iff every virtual session has a pseudobottleneck link.*

The pseudobottleneck lemma serves as a test for maximal fairness of a feasible layer allocation vector. There exists a similar result for maxmin fairness in a continuous feasible set which says that a rate allocation is maxmin fair if and only if every virtual session has a bottleneck link [15]. The definitions of bottleneck links are similar in both cases. The pseudobottleneck lemma strengthens our contention that maximal fairness is a good notion of fairness. Note that,

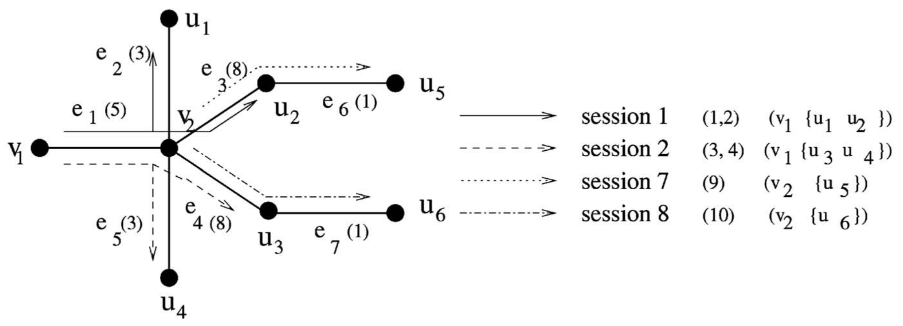

Fig. 4. This figure indicates that the intuitively appealing algorithm presented in Section 4.2 for generating the maximally fair allocation fails. The numbers next to the sessions refer to the virtual sessions belonging to the sessions, e.g., session 1 consists of virtual sessions $1, 2$ with receivers u_1, u_2, respectively. Sessions $3, 4, 5, 6$ span one link each, e_3, e_4, e_6, e_7, respectively, and consist of 1 virtual session each, virtual sessions $5, 6, 7, 8$ respectively. These sessions have not been shown in the figure. Here, $b = 1$ and $C_l = b\lfloor \frac{C_l}{b} \rfloor$ for every link l. The maxmin fair rate allocation assuming the feasible set to be continuous is given by $\vec{r}$, where $r_i = 2.5$, $i \in \{1, \dots, 4\}$, 5, $i \in \{5, 6\}$, , and 0.5, otherwise. Going by this algorithm, we initially allocate γ_i layers to virtual session i, where $\gamma_i = 2$, $i \in \{1, \dots, 4\}$, 5, $i \in \{5, 6\}$, 0, otherwise. Virtual sessions $7, \dots, 10$ are allocated the minimum number of layers. We can increment the layers of either virtual sessions $7, 8$ or $7, 10$, or $8, 9$, or $9, 10$, but not those of any bigger combination among $7, \dots, 10$ because of feasibility. Let us select virtual sessions $9, 10$ arbitrarily. Now, $(3, 2, 2, 2, 5, 5, 0, 0, 1, 1)$ or $(2, 2, 2, 3, 5, 5, 0, 0, 1, 1)$ are the possible output layer allocations depending on whether we select virtual session 1 or 4 for further incrementation at the next stage. Neither is maximally fair. A layer allocation $(3, 3, 2, 2, 4, 5, 0, 0, 1, 1)$ is feasible and fairer than $(3, 2, 2, 2, 5, 5, 0, 0, 1, 1)$. A layer allocation $(2, 2, 3, 3, 5, 4, 0, 0, 1, 1)$ is feasible and fairer than $(2, 2, 2, 3, 5, 5, 0, 0, 1, 1)$. Thus, incrementation in increasing order of layer allocation may lead to layer allocations that are not maximally fair.

by this lemma, if a layer allocation vector is maximally fair, then the number of layers allocated to a virtual session s can be increased only by decreasing the number of layers of one or more virtual sessions that have at most one layer more than s. From the definition of a pseudobottleneck link, if there is no requirement related to the minimum number of layers for any virtual session, then the number of layers, γ_s, of any virtual session s will be at least $\lceil \frac{C_l}{|n(l)|b} - 1 \rceil$ for some link l in its path. Thus, every virtual session is guaranteed a bandwidth close to the fair share of the capacity for at least one link on its path. In the presence of minimum rate requirements, γ_s must be at least

$$\left\lceil \left(\frac{C_l - \sum_{i \in \tau(l)} \mu_{il}}{|n(l) \setminus \tau(l)|b} - 1 \right) \right\rceil$$

layers, for some link l on its path, where $\tau(l) \subset n(l)$ is the set of sessions traversing link l with minimum session rate in link l exceeding the rate of virtual session s by more than b. Intuitively, this means that virtual session s receives an almost fair share of the residual link l bandwidth, after distributing the minimum rates to other sessions. We now deduce the expressions presented here. Let l be the pseudobottleneck link of virtual session s.

$$\lambda_{\chi(s)l} + \sum_{i \in \tau(l)} \mu_{il} + \sum_{\substack{i \neq \chi(s) \\ i \in n(l) \setminus \tau(l)}} \lambda_{il} > C_l - b$$

(pseudobottleneck conditions 1 and 3).

$$b\gamma_s + \sum_{i \in \tau(l)} \mu_{il} + \sum_{\substack{i \neq \chi(s) \\ i \in n(l) \setminus \tau(l)}} (b\gamma_s + b) > C_l - b$$

(pseudobottleneck conditions 2 and 3).

$$\gamma_s \geq \left\lceil \left(\frac{C_l - \sum_{i \in \tau(l)} \mu_{il}}{|n(l) \setminus \tau(l)|b} - 1 \right) \right\rceil.$$

In the absence of minimum layer requirement, $\mu_{il} = 0$ for each session i and link l and $\tau(l) = \phi$ from condition 3 in the definition for the bottleneck link. The expression in this case follows by making these substitutions.

We present a polynomial complexity algorithm for computing a maximally fair layer allocation in the next subsection. The corresponding rate allocation is also maximally fair.

4.2 Algorithm for Computation of Maximally Fair Layer Allocation

We first describe the challenges associated with designing an algorithm that would generate a maximally fair allocation of layers. Recall that the fairness complications described in Section 3 arise on account of the discrete nature of the feasible set. A maxmin fair allocation exists and can be computed in polynomial complexity under the same capacity and the minimum rate constraints, if the discreteness restriction is relaxed [15]. It is thus natural to ponder whether there exists a simple procedure to compute the maximally fair allocation from such a maxmin fair allocation. Interestingly, the answer is negative. Let r^c_s be the maxmin fair bandwidth allocated to virtual session s in the corresponding continuous feasible set. Consider the following intuitively appealing approach: Initially allocate $\lfloor \frac{r^c_i}{b} \rfloor$ layers to virtual session i, $\forall i$. Now, try to add a layer to virtual session $1, \dots, M$, in some predetermined order, e.g., increasing order of the layers allocated, without decreasing the number of layers allocated to any virtual session at any time. Repeat this process as long as the number of layers allocated to a virtual session can be increased without decreasing the layers allocated to some other virtual session. A counterexample provided in Fig. 3 demonstrates that this algorithm need not yield a maximally fair layer allocation. In fact, examples in Fig. 3 show that there may exist virtual sessions i, j such that $\gamma_i < \lfloor \frac{r^c_i}{b} \rfloor$ and, similarly, $\gamma_j > \lceil \frac{r^c_i}{b} \rceil$ in all maximally fair layer allocations. If, however, the maxmin fair rate allocations are computed with capacity C_l of link l replaced by $b\lfloor \frac{C_l}{b} \rfloor$, then the algorithm mentioned above may yield a maximally fair layer allocation if one tries to add a layer to the virtual sessions in some particular order. But, there is no obvious way to determine this order, particularly for multicast

TABLE 2
Summary of Symbols Used in the Rate Computation Algorithm

Symbol	Meaning
L_s	Set of links traversed by virtual session s
$\mathcal{L}$	Set of links in the network
$r_s(k)$	Bandwidth allocated to virtual session s at the end of the kth iteration
$\vec{r}(k)$	rate vector at the end of the kth iteration, with components $r_s(k)$
$\lambda_{il}(k)$	Bandwidth allocated to session i in link l at the end of the kth iteration $\lambda_{il}(k) = \max_{j \in m(i,l)} r_j(k)$ (session link bandwidth)
$\omega_s(k)$	An intermediate bandwidth allocated to virtual session s at the end of the kth iteration
$\vec{\omega}(k)$	An intermediate rate vector with components $\omega_s(k)$
$\Omega_{il}(k)$	Intermediate bandwidth allocated to session i in link l $\Omega_{il}(k) = \max_{j \in m(i,l)} \omega_j(k)$
$S(k)$	Set of unsaturated virtual sessions at the end of the kth iteration
$\Lambda(k)$	Set of virtual sessions which are saturated w.r.t. rate allocation $\vec{\omega}(k)$, $\Lambda(k) = \{s : \exists\, l \in L_s, \omega_s(k) = \Omega_{\chi(s)l}(k), \sum_{i \in n(l)} \Omega_{il}(k) > C_l - b\}$
$U_l(k)$	Set of unsaturated sessions passing through link l at the end of the kth iteration
$F_l(k)$	Total bandwidth consumed by the saturated sessions passing through link l at the end of the kth iteration
$\eta_l(k)$	Link control parameter of link l at the end of the kth iteration
$\eta_{il}(k)$	Session link parameter of session i in link l at the end of the kth iteration, $\eta_{il}(k) = \max(\eta_l(k), \mu_{il})$
$\vec{\gamma}$	output layer allocation vector

networks. Fig. 4 shows that trying to increment the number of layers of virtual sessions in increasing order of layers allocated may not always lead to a maximally fair layer allocation. It is not easy to know beforehand the right order. Trying all possible combinations of orders will yield an algorithm with exponential complexity in the worst case. Using a different approach, we have developed a polynomial complexity algorithm for computing a maximally fair layer allocation.

Definition 8 (Saturation). *Consider a virtual session s which belongs to session i. Virtual session s is* saturated *under a rate vector if it traverses a link in which the session link rate of session i is equal to the rate of the virtual session and if the difference between the bandwidth consumed in the link and the capacity of the link is less than b units. A session is* saturated *on a link l if all the virtual sessions of the session traveling through the link l are saturated.*

We summarize other notations used in the algorithm in symbol Table 2. We now briefly describe our approach in designing a fair bandwidth allocation algorithm. The algorithm computes the fair bandwidth in an iterative manner. In each iteration k, every link l computes a fair share $\eta_{il}(k)$ for each session i traversing the link l. This fair share, denoted as session link parameter, is computed with a view to equalizing the bandwidth of different sessions in the link as far as possible, without decreasing the bandwidth allocations from the previous iterations. Bandwidth allocation for a receiver is determined by the minimum value of the fair shares in its path. Bandwidth in a link may not be fully utilized because of the congestion experienced by sessions in other links. The fair shares are recomputed in the next iteration so as to redistribute this additional bandwidth. Bandwidth allotted to a receiver is thereby progressively improved in the iterations, until it saturates. Once a virtual session saturates, its bandwidth is not changed any further in the computation. The algorithm terminates when every virtual session is saturated. The layer allocations are obtained by dividing the bandwidth allocation by b. Lemma 7 shows that every saturated virtual session has a pseudobottleneck link. The operation of the algorithm also ensures that the layer allocations satisfy the feasibility constraints at each step. Thus, from the pseudobottleneck lemma (Lemma 5), the algorithm generates a maximally fair layer allocation.

The algorithm follows.

1. $k = 0$, $\eta_l(0) = 0$, $F_l(0) = 0$, $U_l(0) = n(l) \; \forall$ link l, $S(0) = \{1, \ldots, M\}$, $r_j(0) = \mu_j$, $\forall j \in S(0)$,
$$\lambda_{il}(0) = \max_{j \in m(i,l)} r_j(0).$$
2. $k \rightarrow k+1$.
3. For every link l in the network, compute the link control parameter:
If $U_l(k-1) \neq \phi$, then
$$\eta_l(k) = \max_\theta \left(F_l(k-1) + \sum_{i \in U_l(k-1)} \max(\theta, \lambda_{il}(k-1)) = C_l \right), \tag{1}$$
else $\eta_l(k) = \eta_l(k-1)$.
For all unsaturated virtual sessions s passing through link l, $(s \in S(k-1) \cap (\cup_{i \in \{1, \ldots N\}} m(i,l)))$, session link parameter,
$$\eta_{\chi(s)l}(k) = \max(\eta_l(k), \lambda_{\chi(s),l}(k-1)).$$
4. If virtual session $s \in S(k-1)$, $\omega_s(k) = b\lfloor \frac{\min_{l \in L_s} \eta_{\chi(s)l}(k)}{b} \rfloor$, else $\omega_s(k) = r_s(k-1)$.
5. For every link l in the network and every session i in $n(l)$, $\Omega_{il}(k) = \max_{s \in m(i,l)} \omega_s(k)$.
6. Compute the set of virtual sessions saturated during the kth iteration under rate vector $\vec{\omega}(k)$:
$$\Lambda(k) = \Big\{ s : s \in S(k-1), \exists \, l \in L_s, \omega_s(k) = \Omega_{\chi(s)l}(k), \sum_{i \in n(l)} \Omega_{il}(k) > C_l - b \Big\}.$$
7. If $\Lambda(k) \neq \phi$, $r_s(k) = \omega_s(k)$, $\forall s$ and go to Step 9.
8. If possible, find a virtual session $s \in S(k-1)$, s.t.
$$\omega_s(k) < \min_{l \in L_s} \eta_l(k) \text{ and}$$
$$\Omega_{il}(k) \geq b \left\lfloor \frac{\eta_l(k)}{b} \right\rfloor \; \forall i \in U_l(k-1) \text{ if } l \in L_s \text{ and}$$
$$\min_{l_1 \in L_s} \left\lfloor \frac{\eta_{l_1}(k)}{b} \right\rfloor = \left\lfloor \frac{\eta_l(k)}{b} \right\rfloor.$$
If no such s is found in $S(k-1)$, again $r_j(k) = \omega_j(k)$ for all virtual sessions, j, otherwise, compute $r_j(k)$ for all virtual sessions j as
$$r_j(k) = \begin{cases} \omega_j(k) & j \neq s \\ \omega_j(k) + b & \text{otherwise.} \end{cases}$$
9. For every link l and every session in $n(l)$, session link rate $\lambda_{il}(k) = \max_{s \in m(i,l)} r_s(k)$.
10. Compute the set of virtual sessions unsaturated after the kth iteration,
$$S(k) = S(k-1) \setminus \Big\{ s : \exists \, l \in L_s, \text{ s.t. } \sum_{i \in n(l)} \lambda_{il}(k) > C_l - b \text{ and } r_s(k) = \lambda_{\chi(s)l}(k) \Big\}.$$
11. If $S(k) = \phi$, i.e., all virtual sessions are saturated, compute the layer allocation vector $\vec{\gamma}$ via $\gamma_j = \frac{r_j(k)}{b}$ and the algorithm terminates, else go to the next step.
12. For every link l, compute the set of unsaturated sessions passing through the link l at the end of the kth iteration:
$$U_l(k) = \{ n : n \in \{1, \ldots, N\}, m(n,l) \cap S(k) \neq \phi \}.$$
13. For every link l, for which $U_l(k) \neq \phi$, compute the bandwidth consumed by the saturated sessions passing through link l, $F_l(k) = \sum_{i \in n(l) \setminus U_l(k)} \lambda_{il}(k)$.
14. Go to Step 2.

We explain the algorithm here.

(Step 1): The rates of the virtual sessions are initialized to the respective minimum rates. All sessions and virtual sessions are unsaturated.

(Step 3): The algorithm computes the session link parameters. The session link parameter for session i in the kth iteration is the maximum of the session link bandwidth allocated in the $k-1$th iteration to session i in link l and the link control parameter, $\eta_l(k)$. The significance of the link control parameter is the following: Assume that a session i traversing link l has no bandwidth constraint in other links, no minimum rate requirements and can receive any continuous bandwidth. Then, the bandwidth allocated to i in the kth iteration is the link control parameter, $\eta_l(k)$. As we discuss in subsequent steps, in the presence of the above-mentioned constraints, the bandwidth allocated to a session in a link is a function of the link control parameter. Now, we discuss the computation of the link control parameter. If all sessions are saturated in the link, then the link control parameter is the same as that in the previous iteration. Otherwise, the link control parameter is computed via (1) which we explain using a water-filling analogy. The residual bandwidth in link l in iteration k is the difference between the link capacity and the sum of the session link bandwidth allocated to all sessions in iteration $k-1$. If the additional bandwidth were to behave like a fluid and allowed to flow among the unsaturated sessions, it would flow so as to equalize the bandwidth level among these sessions. Thus, bandwidth would first flow into the unsaturated session with the lowest amount of bandwidth and then into the unsaturated sessions with the lowest and the second lowest amount of bandwidth and so on. The link control parameter is the amount of bandwidth of the sessions that have received this additional bandwidth; note that all these sessions have an equal amount of bandwidth at the end of the water-filling process.

(Step 4): The algorithm computes the minimum of the session link parameters in the path of each virtual session. A virtual session is assigned a rate equal to the greatest multiple of b not exceeding this minimum. The resulting bandwidth allocation is $\vec{\omega}(k)$.

(Steps 5 and 6): The algorithm determines which virtual sessions become saturated under the new bandwidth allocation $\vec{\omega}(k)$.

(Step 7): If $\vec{\omega}(k)$ saturates at least one virtual session that was not saturated earlier, then this allocation is not changed any further in this iteration, $\vec{r}(k) = \vec{\omega}(k)$.

(Step 8): If no virtual session is saturated, ($\Lambda(k) = \phi$), then the algorithm tries to find a virtual session s that satisfies the properties mentioned in Step 8. At least one such virtual session exists in this case (Lemma 11). The rate of such a virtual session is incremented by b. This is done because, otherwise, the same session link parameters will be computed in the next iteration and the algorithm will continue indefinitely.

(Steps 9-14): The algorithm updates states under the new allocation, $\vec{r}(k)$ The states include the session link rate **(Step 9)**, the set of unsaturated virtual sessions **(Step 10)**, the set of unsaturated sessions **(Step 12)**, and the bandwidth consumed by the saturated sessions **(Step 13)**. These states are used in the computation of the link control parameters in Step 3 at the beginning of the next iteration. The algorithm terminates if all virtual sessions are saturated **(Step 11)**.

The algorithm generates a maximally fair layer allocation vector (Theorem 2) in a finite number of iterations (Theorem 3).

An example illustrating the operation of the algorithm follows.

Example 4.2. Consider the network in Example 4.1. $L_1 = \{e_1, e_2, e_4\}$, $L_2 = \{e_1, e_3, e_5\}$, $L_3 = \{e_1, e_3, e_6\}$. Link control parameters can be computed as, $\eta_{e_1}(1) = 3$, $\eta_{e_2}(1) = 4$, $\eta_{e_3}(1) = 2.5$, $\eta_{e_4}(1) = 4$, $\eta_{e_5}(1) = 4$, $\eta_{e_6}(1) = 6$. We explain the computation of $\eta_{e_1}(1)$ and $\eta_{e_3}(1)$. The bandwidth allocated to sessions 1 and 2 in iteration 0 in link e_1 are their minimum session link rates, i.e., 4 and 0, respectively. Thus, the residual bandwidth is $C_{e_1} - 4$, which is 3 units. Using the water-filling argument, the entire residual bandwidth of 3 units is added to session 2 which has lower bandwidth all through the water-filling process. The link control parameter is the bandwidth of session 2 at the end of the process, which is 3 units. Now consider link e_3. The bandwidth allocated to sessions 1 and 2 in iteration 0 are 2 and 0, respectively. The additional bandwidth is 3 units. First, this bandwidth is added to session 2 until its bandwidth equals that of session 1, which is 2 units. One additional unit of bandwidth is still available which is split equally between sessions 1 and 2 by the water-filling process. Now, sessions 1 and 2 have 2.5 units of bandwidth each and, thus, the link control parameter is 2.5.

The session link parameters are: $\eta_{1e_1}(1) = 4$, $\eta_{2e_1}(1) = 3$, $\eta_{1e_2}(1) = 4$, $\eta_{1e_3}(1) = 2.5$, $\eta_{2e_3}(1) = 2.5$, $\eta_{1e_4}(1) = 4$, $\eta_{1e_5}(1) = 4$, $\eta_{2e_6}(1) = 6$. Computing the $\omega_s(1)$s as per Step 4, we have $\omega_1(1) = 4$, $\omega_2(1) = 2$, $\omega_3(1) = 2$. Observe that virtual session 1 is saturated w.r.t. $\vec{\omega}(1)$. So, $\vec{r}(1) = \vec{\omega}(1)$. Virtual sessions 2 and 3 are not saturated w.r.t. $\vec{r}(1)$. Here, $S(1) = \{2, 3\}$, $U_{e_1}(1) = U_{e_3}(1) = \{1, 2\}$, $U_{e_5}(1) = \{1\}$, $U_{e_6}(1) = \{2\}$, and $U_l(1) = \phi$, if $l \in \{e_2, e_4\}$. Also, $F_{e_l}(1) = 0$ if $l \notin \{e_2, e_4\}$ and $F_{e_l}(1) = 4$ otherwise.

Computations for the next iteration are as follows: $\eta_{e_1}(2) = 3$, $\eta_{e_3}(2) = 2.5$, $\eta_{e_5}(2) = 4$, $\eta_{e_6}(2) = 6$, and $\eta_l(2) = \eta_l(1)$ for the rest of the links. The session link parameters are as follows: $\eta_{1e_1}(2) = 4$, $\eta_{2e_1}(2) = 3$, $\eta_{1e_2}(2) = 4$, $\eta_{1e_3}(2) = 2.5$, $\eta_{2e_3}(2) = 2.5$, $\eta_{1e_4}(2) = 4$, $\eta_{1e_5}(2) = 4$, $\eta_{2e_6}(2) = 6$. $\omega_2(2) = \omega_3(2) = 2$. $\omega_1(1) = r_1(1) = 4$. No new virtual session is saturated w.r.t. $\vec{\omega}(2)$. Both virtual sessions 2 and 3 satisfy the conditions for incrementation in Step 8. We arbitrarily choose to increment the layer allocation of virtual session 2. Then, $r_1(2) = 4, r_2(2) = 3, r_3(2) = 2$. This saturates both the virtual sessions 2 and 3. All virtual sessions are saturated and, hence, the algorithm terminates.

Theorem 2 (Maximal-Fairness Theorem). *The output layer allocation vector $\vec{\gamma}$ is*

1. *maximally fair and*
2. *maxmin fair, if a maxmin fair layer allocation exists.*

We prove Theorem 2 using the following lemmas.

Lemma 6. *The rate allocation $\vec{r}(k)$ at the end of the kth iteration is feasible, $k \geq 0$.*

Lemma 7. *If a virtual session s is saturated with respect to allocation $\vec{r}(k)$,, then it has a pseudobottleneck link.*

Proof of Theorem 2. The algorithm terminates only when all the virtual sessions saturate. Thus, the maximal fairness of the final rate and layer allocation vectors follow from Lemmas 6, 7, and the pseudobottleneck lemma (Lemma 5). The last part of the theorem follows from Lemma 3. □

At the end of every iteration, either the rate of at least one virtual session increases by b units or the number of unsaturated virtual sessions decreases by at least one (proof of Theorem 3 in the Appendix (which can be found on the Computer Society Digital Library at http://computer.org/tc/archives.htm)). Neither of these two can continue indefinitely. Hence, the algorithm terminates in a finite number of iterations. Theorem 3 establishes a stronger result, that the algorithm terminates in a polynomial number of iterations.

Theorem 3 (Finite-Termination Theorem). *The algorithm terminates in at most $M + |\mathcal{L}|M$ number of iterations, where $\mathcal{L}$ is the set of links and M is the number of virtual sessions.*

Every step of this algorithm has a complexity of $O(|\mathcal{L}|M)$. The algorithm must terminate in $M + |\mathcal{L}|M$ iterations. Thus, the overall complexity of this algorithm is $O(|\mathcal{L}|^2 M^2)$.

Note that the number of iterations does not depend on the value of the layer bandwidth b, even though the bandwidths of the virtual sessions may increase by b at every step. It can be shown that this happens in at most $|\mathcal{L}|M$ iterations (Lemma 12). In the remaining iterations, at least one new virtual session saturates. Since there are M virtual session, the algorithm terminates in at most $M + |\mathcal{L}|M$ iterations.

Finally, we point out a few salient features of the algorithm. Appropriate selection of an unsaturated virtual session s whose rate will be increased is crucial for attaining maximal fairness. We present an example that shows that, if $\Lambda(k) = \phi$ and the rate of a virtual session s is incremented by b units without satisfying the requirements of Step 8, then the output layer allocation may not be maximally fair.

Consider the network topology in Fig. 2 with different link capacities. The capacities of links e_1, e_2, e_3 are $1, 4, 6.2$, respectively. Every layer consumes unit bandwidth. There is no minimum and maximum layer requirement. Note that $\eta_{e_1}(1) = 0.5$, $\eta_{e_2}(1) = 2$, and $\eta_{e_3}(1) = 3.1$. Now, $\omega_1(1) = \omega_2(1) = 0$, $\omega_3(1) = 2$, and $\omega_4(1) = 3$. Observe that the difference between bandwidth consumed and capacity is at least 1 unit for every link. Thus, $\Lambda(1) = \phi$. For sessions 1, 2, 4, $\omega_j(1) < \min_{l \in L_j} \eta_l(1)$. Only sessions 1 and 2 satisfy the other criterion for incrementation though. If, however, we increment by 1 the rate of session 4 instead of either session 1 or 2, we obtain a rate vector $\vec{r}(1) = (0, 0, 2, 4)$. Sessions 3 and 4 are both saturated now. So, $S(1) = \{1, 2\}$. In iteration 2, $\eta_{e_1}(2) = 0.5$, $\eta_{e_2}(2) = 2$. Note that $\omega_s(2) = 0, \ s \in \{1, 2\}$. The difference between the bandwidth consumed and capacity is again at least 1 unit for both links e_1 and e_2. Thus, $\Lambda(2) = \phi$. Both sessions 1 and 2 satisfy the criteria for incrementation. Incrementing the rate of session 1 (selected arbitrarily among sessions $1, 2$) by 1 unit saturates sessions 1 and 2 and the algorithm terminates with rate allocation $(1, 0, 2, 4)$. The layer allocation vector is $(1, 0, 2, 4)$ as well. Now, $(1, 0, 3, 3)$ is a feasible layer allocation vector and is fairer than $(1, 0, 2, 4)$. Thus, the output of the algorithm $(1, 0, 2, 4)$ is not a maximally fair layer allocation vector. This is because of the selection of session 4 as a candidate for rate incrementation in iteration 1, despite the fact that $\lfloor \eta_{e_3}(1) \rfloor = \min_{l \in L_4} \lfloor \eta_{e_3}(1) \rfloor$ and $\Omega_{3e_3}(1) = \omega_3(1) < \lfloor \eta_{e_3}(1) \rfloor = 3$.

5 Conclusion and Discussion

To the best of our knowledge, this is the first study in fairness in a discrete feasible set. The situation arises when bandwidth can be allocated in discrete chunks only. Fairness in a discrete feasible set is vastly different from that in a continuous feasible set and has not been explored before, for both multicasting, as well as unicasting. Maxmin fair rate allocation may not exist in the discrete feasible set. Lexicographically optimal allocation can be considered as a discrete version of maxmin fairness. We have shown that the computation of lexicographically optimal layer allocation is NP-hard in the discrete case. We then introduced the notion of maximally fair rate allocation and showed that a maximally fair layer allocation has many nice properties with respect to fairness. Summarizing, the appealing properties are 1) pareto optimality, 2) min-order property, and 3) pseudobottleneck property. Furthermore, maximal fairness and maxmin fairness lead to the same allocation, when a maxmin fair layer allocation exists, and a lexicographically optimal allocation is maximally fair. We have presented a polynomial complexity algorithm for computing a maximally fair rate allocation. We present some interesting topics for future research.

Developing computationally simple approximation algorithms for allocation of rates close to a lexicographically optimal layer allocation w.r.t. some useful metric is an interesting area for future research. It is not even known whether such algorithms can exist. Good heuristics for this purpose may also be useful.

We have so far assumed that every layer of every source consumes the same bandwidth, i.e., b units. This assumption has been made elsewhere, as well, e.g., while simulating the RLM internet protocol, McCanne et al. [10] assume that every layer consumes 32 kb/s bandwidth. This assumption, however, does not hold in all coding schemes and fairness in these scenarios becomes more technical. This is beyond the scope of the current paper. We have considered this more general case in [14]. The algorithm presented there is, however, technical and fairly complicated. The algorithm we present here is simpler to implement and gives the essential intuition, even though it addresses a special case of [14]. Also, this algorithm has a complexity of $O(L^2M^2)$,, but the more general algorithm has a complexity of $O(L^2M^3)$.

Our algorithm is amenable to distributed implementation. The criteria for determination of the rate of a virtual session mainly uses information along the path of the virtual session. The algorithm uses global information only when it ensures that the rate of at most one virtual session, s, $r_s(k)$ is increased as $r_s(k) = \omega_s(k) + b$. This feature of the algorithm is not crucial to the proof of maximal fairness of the output and is a matter of convenience. The rates of multiple virtual sessions can be increased, subject to feasibility and as long as the virtual sessions satisfy the criteria of Step 8 and the algorithm will still provide a maximally fair layer allocation. The challenge will be to design a distributed rate incrementation procedure that maintains feasibility. This is a topic of future research.

Acknowledgments

The authors wish to thank S. Khuller for discussions on the proof for NP-hardness of lexicographically optimal allocation. The work of Saswati Sarkar was supported in part by US National Science Foundation grants AN101-06984 and NCR02-38340. Part of this paper was presented at INFOCOM 2000.

References

[1] T. Ballardie, P. Francis, and J. Crowcroft, "Core Based Trees: An Architecture for Scalable Inter-Domain Multicast Routing," *Proc. ACM SIGCOMM*, pp. 85-95, Sept. 1993.

[2] D. Bertsekas and R. Gallager, *Data Networks.* Englewood Cliffs, N.J.: Prentice Hall, 1987.

[3] J. Byers, M. Luby, and M. Mitzenmatcher, "Fine-Grained Leyered Multicast," *Proc. IEEE INFOCOM*, pp. 1143-1151, 2001.

[4] S. Deering and D. Cheriton, "Multicast Routing in Datagram Internetworks and Extended LANs," *ACM Trans. Computer Systems,* vol 8, no. 2, pp. 54-60, Aug. 1994.

[5] M.R. Garey and D.S. Johnson, *Computers and Intractability.* Freeman, 1979.

[6] A. Legout and E.W. Biersack, "PLM: Fast Convergence for Cumulative Layered Multicast Transmission Schemes," *Proc. ACM SIGMETRICS 2000*, pp. 13-22, 2000.

[7] A. Legout, J. Nonnenmacher, and E.W. Biersack, "Bandwidth Allocation Policies for Unicast and Multicast Flows," *IEEE/ACM Trans. Networking,* vol. 9, no. 4, pp. 464-478, Aug. 2001.

[8] X. Li, S. Paul, and M. Ammar, "Layered Video Multicast with Retransmission (LVMR): Evaluation of Hierarchical Rate Control," *Proc. IEEE Infocom '98,* pp. 1062-1072, Mar. 1998.

[9] X. Li, S. Paul, and M. Ammar, "Multi-Session Rate Control for Layered Video Multicast," Technical Report GT-CC-98-21, College of Computing, Georgia Inst. of Technology, 1998.

[10] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-Driven Layered Multicast," *Proc. ACM SIGCOMM '96,* 1996.

[11] G. Rouskas and I. Baldine, "Multicast Routing with End-to-End Delay and Delay Variation Constraints," *IEEE J. Selected Areas in Comm.,* vol. 15, no. 3, pp. 346-356, Apr. 1997.

[12] D. Rubenstein, J. Kurose, and D. Towsley, "The Impact of Multicast Layering on Network Fairness," *IEEE Trans. Networking,* vol. 10, no. 2, pp. 169-182, Apr. 2002.

[13] S. Sarkar and K.N. Sivarajan, "Fairness in Wireless Mobile Networks," *IEEE Trans. Information Theory,* vol. 48, no. 8, pp. 2412-2426, Aug. 2002.

[14] S. Sarkar and L. Tassiulas, "Fair Allocation of Utilities in Multirate, Multicast Networks: A Framework for Unifying Diverse Fairness Objectives," *IEEE Trans. Automatic Control,* special issue on control problems in comm. networks, vol. 47, no. 6, pp. 931-944, June 2002.

[15] S. Sarkar and L. Tassiulas, "Distributed Algorithms for Computation of Fair Rates in Multirate Multicast Trees," *Proc. IEEE INFOCOM 2000,* pp. 52-62, Mar. 2000.

[16] L. Vicisano, L. Rizzo, and J. Crowcroft, "TCP-Like Congestion Control for Layered Multicast Data Transfer," *Proc. INFOCOM '98,* pp. 996-1003, 1998.

Saswati Sarkar received the Master of Engineering degree in electrical communication engineering from the Indian Institute of Science in 1996 and the PhD degree in electrical and computer engineering from the University of Maryland, College Park, in 2000. She is currently an assistant professor in the Department of Electrical and Systems Engineering at the University of Pennsylvania. Her research interests are in resource allocation and performance analysis in communication networks. She received a US National Science Foundation Career award in 2003 for her research in wireless ad hoc networks. She is an associate editor of the *IEEE Transactions on Wireless Communications*. She is a member of the IEEE.

Leandros Tassiulas received the Diploma in electrical engineering from the Aristotelian University of Thessaloniki, Greece, in 1987 and the MS and PhD degrees in electrical engineering from the University of Maryland, College Park, in 1989 and 1991, respectively. From 1991 to 1995, he was an assistant professor in the Department of Electrical Engineering, Polytechnic University, Brooklyn, New York. In 1995, he joined the Department of Electrical Engineering, University of Maryland, where he is now an associate professor. He holds a joint appointment with the Institute for Systems Research and is a member of the Center for Satellite and Hybrid Communication Networks, established by NASA. His research interests are in computer and communication networks, with emphasis on wireless communications (terrestrial and satellite systems) and high-speed network architectures and management, in control and optimization of stochastic systems in parallel and distributed processing. He coauthored a paper that received the INFOCOM 1994 Best Paper Award. He received a US National Science Foundation (NSF) Research Initiation Award in 1992, an NSF Faculty Early Career Development Award in 1995, and a US Office of Naval Research Young Investigator Award in 1997.