Fair Allocation of Discrete Bandwidth Layers in Multicast Networks

Saswati Sarkar and *Leandros Tassiulas*
Dept. of Electrical and Computer Engineering and Institute for Systems Research
University of Maryland, College Park, MD, USA

Abstract— **We study fairness when receivers in a multicast network can not subscribe to fractional layers. This case arises when the source hierarchically encodes its signal and the hierarchical structure is predetermined. Unlike the case of the fractional layer allocation, which has been studied extensively in [14], bandwidth can be allocated in discrete chunks only. Fairness issues become vastly different. 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. We propose a polynomial complexity algorithm for computation of maximally fair rates allocated to various source-destination pairs. Even though, maximal fairness is a weaker notion of fairness, it coincides with lexicographic optimality and maxmin fairness, when maxmin fair rate allocation exists. So the algorithm for computing maximally fair rate allocation computes maxmin fair rate allocation, when the latter exists.**

I. Introduction

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. A single rate of transmission per session is likely to either overwhelm the slow receivers or starve the fast ones, in absence of additional provisions. Multirate transmission should be used to counter network heterogenity. Multirate transmission can be attained using a hierarchical or a layered transmission 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 we discuss later, the layer bandwidths are often predetermined and can not be tuned according to the needs of the network. A receiver either receives a layer fully or does not receive the layer, at all. It can not partially subscribe to a layer, unlike [14]. Effectively, the network can only allocate a discrete set of rates to the receivers, whereas a continuous set of rates can be allocated when receivers can subscribe to fractional layers[14]. We study fair allocation of 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[3] is a well accepted notion of fairness. A rate allocation is maxmin fair, if no receiver can be allocated a higher rate without hurting another receiver having equal or lower rate. A maxmin fair rate allocation may not exist in a discrete set. However, a maxmin fair rate allocation always exists in a continuous set. Lexicographic optimality is another notion of fairness. A lexicographically optimal rate vector is one which maximizes its minimum component in a feasible set, subject to this maximization, it maximizes the second minimum, etc*. Lexicographically optimal rate allocation exists in a discrete set, but as we prove later, its computation is an NP-hard problem. However lexicographically optimal rate allocation is identical to the maxmin fair rate allocation and is thus polynomial complexity computable in a continuous set[11][14]. We can compute a maximally fair rate allocation in a discrete set, instead. We introduce the concept of maximal fairness more formally later, but a rate allocation is maximally fair if no other rate allocation is "fairer" in some sense. 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. If a rate allocation is maximally fair, then any other rate allocation will be "unfair" or "less fair" to some receiver. Maximally fair rate allocation is a weaker notion of fairness as compared to maxmin fairness and lexicographic optimality. But, maximal fairness has various desirable properties 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 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 do not assume any properties specific to 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.

We review related work for discrete bandwidth layer allocation briefly. Well known network protocols for layered transmission, RLM (Receiver-driven Layered Multicast)[10] and LVMR (Layered Video Multicast with Retransmissions)[7] do not handle fairness among sessions very well, when there are multiple sessions competing for bandwidth[8]. An alternative layer allocation scheme has

*Maxmin fairness and lexicographic optimality will be defined formally later.

been proposed in [8]. There is empirical evidence that this scheme improves fairness among sessions for networks with multiple video sessions sharing only one link. But, there is no experimental or analytical evidence that the scheme works well for more complex networks, with sessions sharing several links with each other. Besides [8] does not compute actual rates or the number of layers allocated to the receivers in an arbitrary network, under some well defined notion of fairness. Rubenstein *et. al.* also points out that maxmin fair rate allocations may not exist for discrete bandwidth layers[11]. But, [11] suggests a policy of coordinated random add and drop of the highest layer for various receivers, as a remedy. This attains long term rates close to the maxmin fair allocation. However, this oscillation is likely to produce perceptually annoying distortion. The resulting perceptual quality may even be worse than not subscribing to the highest layer at all. Besides, [11] observes that this random add and drop of highest layer generally leads to underutilization of link capacity. We recommend use of maximally fair rate allocations instead.

We have omitted all proofs throughout, on account of space restrictions. Refer to technical report[15] for proofs and details.

II. Constraints on Signal Structure

We discuss the network scenarios under which the feasible set of rate allocations becomes discrete. In multirate transmission, source transmits at a rate equal to the maximum of the rates allocated to its receivers. At forking points, video gateways may be used to transcode signal into a lower bit rate such that the rate in every link is equal to the maximum of the fair rates allocated to the session receivers downstream[16]. Output rates for video gateways can be fine tuned to match the required receiver rates, if rate adaptive videogateways are used[1]. It may not be possible to know apriori which intermediate nodes need transcoding provisions, because this depends on fair rate allocations and fair rate allocations will change dynamically depending on the traffic conditions. So we may need to deploy transcoding gateways at all intermediate nodes which have a fanout (number of outgoing links) greater than one. This makes transcoding financially prohibitive. Intensive computation required for transcoding increases end to end delay. More importantly, transcoding can not be applied to a secure communication network without entrusting the network with the encryption key. Depending on the security risks, this may be totally unacceptable. In these cases, network heterogenity can be countered through hierarchical encoding only.

We assume that hierarchical coding is used, i.e., the source encodes its signal into a number of layers that can be incrementally combined to provide progressive refinement in quality. Hierarchical coding was first suggested for packet voice transmission[4]. Subsequently, several hierarchical coding schemes have been proposed for video, e.g., [17]. Continuous bandwidth allocation is possible in presence of hierarchical encoding if the layer structure is flexible. In this case, once the desired rates are computed, the source partitions its signal to form as many layers as there are distinct receiver rates and tunes the layer bandwidths to match the receiver rates. However, this tuning is not always possible as layer structure is fixed and predetermined in certain cases. This is because, certain coding schemes are not amenable to dynamic layer bandwidths because of their inherent structure, e.g., perceptually weighted wavelets using hierarchical vector quantization with wavelet decomposition(WWHVQ)[19]. In certain video codes, substreams can be extracted to produce a specific range of resolutions only, e.g., 3D Subband Video Coding[17]. Some coding schemes are particularly successful only when some apriori structure or hierarchy can be found in the problem[18], because computationally expensive optimizations, e.g., quantizer thresholds optimizations, can be carried out offline only. Some codecs, e.g., PVH codec of [9] are amenable to dynamic layer bandwidth adaptation, but the implementations still have a fixed layout strategy. In many cases, generating a layer requires a dedicated filter, and the number of filters employed by the source is fixed. Thus the number of layers is limited. Similarly, requirements for minimum packet size and limitations on packetization delay, may impose a lower bound on bandwidth assigned to a single layer and thus adversely affect layer granularity. Thus layer bandwidths are often fixed and a perfect match between the layer bandwidths and the required receiver rates is not feasible.

Reference [14] suggests the following approach for continuous bandwidth allocation, in cases of fixed or partially adaptive signal hierarchy. Allocate to the receivers as many layers as permitted by the computed rate. If the total bandwidth consumed by the layers allocated to a receiver is strictly less than the computed rate, allocate one more layer to the receiver and let the network drop a certain portion of packets of the last layer at a forking point. This may not work always. This is because the network may not be well equipped to selectively replicate a certain percentage of packets of the highest layer or selectively drop a certain percentage of packets of the highest layer after replicating all the packets. Most of the current day routers follow random drop or drop tail policy, i.e., in event of congestion the routers drop any packet in the queue or the packet at the end of the queue, depending on the respective policy. So if the receivers oversubscribe, the network will be congested and the routers will indiscriminately drop packets of all the layers. In hierarchical encoding, a higher layer yields useful information only when the lower layers have been successfully decoded. So loss of packets of all the layers will adversely affect signal quality. It is a good idea not to subscribe to the highest layer at all in this case and keep the network free of congestion. Besides, if the highest layer employs a differential coding scheme, then even a small percentage loss of packets may garble the entire information in the final layer. Another point to keep in mind is that generally the layers are coded so that the loss of entire layer may cause graceful degradation in quality. However loss of a certain proportion of the highest layer packets may produce perceptually annoying distortions in certain coding schemes. In all these cases, "partial" subscription to a layer is useless and receivers can only subscribe to layers fully. Besides, sometimes a source simply transmits each layer of its signal on a separate multicast group[10]. A receiver either subscribes to a group or it does not. It can not "partially" subscribe.

So we assume that the bandwidths consumed by the layers are predetermined. A receiver can not be allotted a layer "partially". As a consequence the possible rates of a receiver form a discrete set. *Any* rate vector which satisfies the capacity constraints and the minimum and maximum rate constraints can not be allocated. Only a discrete subset of these rate vectors can be allocated. We need to make a fair allocation of rates in this discrete set. Equivalently, we can consider the problem of fair allocation of layers, where every layer has a predetermined bandwidth. It is worthwhile to mention in this context that a layer obtained from an available bandwidth can be looked upon as a function or "utility" of a bandwidth. This utility function is a stair case function because a layer essentially corresponds to a range of bandwidth, the range extending from the minimum bandwidth required to attain the layer to that required for the next higher layer. Thus this function is not strictly increasing. An algorithm for computation of rates for attaining maxmin fair utilities in unicast networks is proposed in [5]. We consider multicast networks here. More importantly, the algorithm attains maxmin fair utilities for strictly increasing utility functions only. The operation of this algorithm is ambiguous for non strictly increasing utility functions in general and staircase utility functions in particular[15]. So [5] does not apply to our case.

III. Network Model

We consider an arbitrary topology network with N multicast sessions. A multicast session is identified by the pair (v, U), where v is the source node of the session and U is the group of intended destination nodes. We assume that the traffic from node v is transported across a predefined multicast tree to nodes in U. The tree can be established during connection establishment phase if the network is connection oriented or can be established by some well known multicast routing protocol like DVMRP[6], CBT[2], etc. in an internet type network.

We call every source destination pair of a session a virtual session. For example, if a session n has source v_n and destination set U_n, where $U_n = \{u_{n1}, \ldots, u_{nt}\}$, then this session would correspond to t virtual sessions, $(v_n, u_{n1}), \ldots, (v_n, u_{nt})$. Refer to figure 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 those for the overall sessions. We assume that every virtual session (source-destination pair) has a minimum and a maximum layer requirement[†].

A layer allocation for the virtual sessions is 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 corresponds to virtual sessions $m_{n1}, \ldots, m_{nt}$ in link l, then the maximum of the bandwidths allocated to the virtual sessions $m_{n1}, \ldots, m_{nt}$ is the bandwidth consumed by session n in link l. Total bandwidth consumed by all sessions traversing through link l can not exceed the capacity of link l. Formal definition follows.

Let γ_j denote the number of layers allocated to virtual session j. Let there be M virtual sessions. A layer allocation vector $\vec{\gamma}$ is an $M-$dimensional vector, with components γ_j. Let $n(l)$ denote the set of sessions passing through link l, $m(k,l)$ denote the set of virtual sessions of session k passing through link l and C_l denote the capacity of link l. Let Γ_{il} denote the number of layers allocated to session i in link l under layer allocation vector $\vec{\gamma}$. It is actually the maximum of the layers allocated to the virtual sessions of the session traversing through link l, i.e., $\Gamma_{il} = \max_{j \in m(i,l)} \gamma_j$. Also ι_j and p_j denote the minimum and the maximum number of layers of virtual session j. For simplicity, we assume that every layer consumes the same amount of bandwidth, b units, independent of the session. A layer allocation vector $\vec{\gamma}$ is feasible if

1. γ_j is an integer for all virtual sessions j,
2. $\iota_j \leq \gamma_j \leq p_j$, $\forall\ j$, $p_j \geq \iota_j \geq 0$, p_j, ι_j are integers.
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).

Equivalently, a layer allocation vector $\vec{\gamma}$ is feasible if the corresponding rate vector $\vec{r}^{\gamma}$, defined as $r_j^{\gamma} = b\gamma_j$ is a feasible rate allocation. A rate-vector $(r_1, \ldots, r_M)$ is a feasible rate allocation if

1. r_i is a multiple of b, the layer bandwidth, for all virtual sessions i,
2. $\mu_i \leq r_i \leq \sigma_i\ \forall\ i$, where μ_i and σ_i are respectively the minimum and maximum rates of virtual session i, $\sigma_i \geq \mu_i \geq 0$, $\mu_i = b\iota_i$ and $\sigma_i = bp_i$,
3. Total bandwidth consumed by various sessions in a link should not exceed the link capacity, i.e., $\sum_{i \in n(l)} \lambda_{il} \leq C_l$ (capacity condition).
   Here λ_{il} denotes the rate allocated to session i on link l under rate allocation $\vec{r}$. It is the maximum of the rates allocated to the virtual sessions of session i traversing link l, i.e., $\lambda_{il} = \max_{j \in m(i,l)} r_j$.

[†]Absence of these requirements can be incoporated by choosing minimum layer requirement as 0 and maximum layer requirement as ∞.

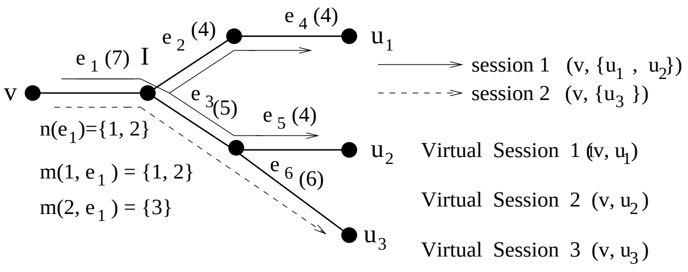

Fig. 1. Session 1 includes virtual sessions, 1, 2. Session 2 includes virtual session 3. The figure shows some sample $n(e_i)$ and $m(i, e_j)$s. The numbers in brackets, () denote the capacities of the respective links. Every layer consumes 0.5 units of bandwidth. The capacity constraint for link e_1 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 $0 \leq r_3 \leq 5$.

Figure 1 illustrates en example network with a few capacity and maximum and minimum rate constraints.

Henceforth we shall ignore the maximum layer constraints. This does not cause any loss in generality because maximum layer constraints can be incorporated by adding artificial links between receivers with maximum layer constraints and the rest of the network. Capacity of such an artificial link is equal to the bandwidth consumed by the maximum number of layers of the respective receiver. The size of the augmented network is comparable to that of the given network. So complexity of any algorithm for computation of fair layer allocation in a network should remain the same, if we use the augmented network instead.

IV. Fairness in Discrete Feasible set

Fairness in a discrete feasible set have not been studied before. First, we describe various useful notions of fairness in a discrete feasible set. 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 that of any other virtual session which has equal or lower number of layers. More formally, a feasible layer allocation vector $\vec{\gamma}^1$ is maxmin fair if it satisfies the following property with respect to any other feasible layer allocation vector $\vec{\gamma}^2$: if there exists i such that the ith component of $\vec{\gamma}^2$, γ_i^2 is strictly greater than that of $\vec{\gamma}^1$, γ_i^1 ($\gamma_i^2 > \gamma_i^1$), then there exists j such that the jth component of $\vec{\gamma}^1$, γ_j^1 is less than or equal to the ith component of $\vec{\gamma}^1$, γ_i^1 ($\gamma_j^1 \leq \gamma_i^1$) and the jth component of $\vec{\gamma}^2$ (γ_j^2) is strictly less than the jth component of $\vec{\gamma}^1$ ($\gamma_j^2 < \gamma_j^1$). The components of $\vec{\gamma}^2$ are more unequal than those of $\vec{\gamma}^1$ in some sense.

Example IV.1: The maxmin fair layer allocation vector in the network of Figure 1 is $(8, 5, 5)$. It is easy to check that this layer allocation vector is feasible. 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 $\gamma_2 = \gamma_3$ in the allocation $(8, 5, 5)$.

Given an $M-$ dimensional vector $\vec{V}$, define its lexicographically ordered version $\hat{V}$ as follows: $\hat{v}_j = v_k$ for some 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 $\vec{\gamma}^1$ is lexicographically greater than a layer allocation vector $\vec{\gamma}^2$ if there exists i such that $\hat{\gamma}_i^1 > \hat{\gamma}_i^2$ and $\hat{\gamma}_j^1 = \hat{\gamma}_j^2$ if $j \leq i$. $\vec{\gamma}^1$ and $\vec{\gamma}^2$ are lexicographically equal if $\hat{\gamma}^1 = \hat{\gamma}^2$. $\vec{\gamma}^1$ is lexicographically less than $\vec{\gamma}^2$ if $\vec{\gamma}^2$ is lexicographically greater than $\vec{\gamma}^1$. A layer allocation vector $\vec{\gamma}$ is lexicographically optimal if it is feasible and if every feasible layer allocation vector is lexicographically less than or equal to $\vec{\gamma}$. Informally, a layer allocation is lexicographically optimal, if its smallest component is the largest amongst the smallest components of all feasible layer allocations, subject to this it has the largest second smallest component and so on.

Example IV.2: A lexicographically optimal layer allocation vector in the network of Figure 1 is $(8, 5, 5)$. The minimum component of every layer allocation vector must be less than or equal to 5, because of the capacity constraint of link e_3. If the minimum component is equal to 5, then the second minimum must be equal to 5 also, because of the capacity constraint of link e_3. Capacity constraint of link e_2 forces the largest component to be 8.

Maxmin fairness and lexicographic optimality are not entirely equivalent, though they have been used interchangeably in many places. Certain references have defined maxmin fair vector as the lexicographic optimal one, e.g.[5]. Our definition of maxmin fairness have been suggested in [3]. Going by this definition, maxmin fairness is stronger than lexicographic optimality. In general, if we consider finite dimensional vectors, with the feasibility set closed and bounded, a lexicographically optimal vector always exists, but a maxmin fair vector may not exist. However as Lemma 1 stated below proves, if a maxmin fair vector exists, it is lexicographically optimal. This holds for arbitrary feasible sets of $k-$dimensional vectors.

Lemma 1: If a vector $\vec{A}$ is maxmin fair in a feasible set, then it is lexicographically optimal in the same feasible set.

In general a maxmin fair layer allocation may not exist. Since layer allocation is a finite dimensional vector with components having integer values and the feasible set is bounded, a lexicographically optimal layer allocation exists though. Consider the following simple example for more insight:

Example IV.3: A network has a single link and two sessions traversing through the link. Every layer consumes 1 unit of bandwith. The capacity of the link is 1 unit. There is no minimum and maximum layer requirements. So possible layer allocation vectors are $(0,1)$, $(1,0)$, $(0,0)$. None of these vectors are maxmin fair. For each of these vectors, it is possible to maintain feasiblity and increase the number of layers of one session, possibly hurting another session, but the other session has higher number of layers. For example, consider the allocation $(0,1)$. It is possible to increase the number of layers of session 1, by hurting session 2. However session 2 has higher number of layers than session 1 in this allocation. Both $(1,0)$ and $(0,1)$ are lexicographically optimal.

If the feasibility set were continuous, as in [14], maxmin fair rate allocation vector would always exist and would hence be lexicographically optimal(Lemma 1). The definitions of maxmin fairness and lexicographic optimality can be used interchangeably in this case. However, in our case, we need to distinguish between the two. In view of the nonexistence of maxmin fair layer allocation, lexicographically optimal layer allocation is the best one can hope for in this case. However as NP-hardness lemma shows, computation of lexicographic optimal layer allocation is an NP-hard problem in this case.

Lemma 2 (NP-hardness) Computation of lexicographically optimal layer allocation vector is NP-hard.

The idea behind the proof is that, given any arbitrary graph G, one can construct a network such that the computation of a lexicographically optimal layer allocation in the network has the same complexity as the computation of the largest independent set in the graph G. It is well known that the latter is an instance of an $NP-$hard problem.

In view of this unfortunate result, we can consider a different notion of fairness, *maximal fairness.* We use the concept of *relative fairness* introduced in [12]. A layer allocation vector $\vec{\gamma}^1$ is *fairer* than another layer allocation vector $\vec{\gamma}^2$, if for every virtual session i which has higher number of layers under $\vec{\gamma}^2$ than under $\vec{\gamma}^1$, there is some other virtual session j whose number of layers was already no more than that of i under $\vec{\gamma}^1$, and has been decreased further by $\vec{\gamma}^2$. A more formal definition of relative fairness follows.

A layer allocation vector $\vec{\gamma}^1$ is *fairer* than another layer allocation vector $\vec{\gamma}^2$ if

- $\vec{\gamma}^1 \neq \vec{\gamma}^2$ and
- if there exists an i such that $\gamma_i^1 < \gamma_i^2$, then there exists a j such that $\gamma_j^1 \leq \gamma_i^1$ and $\gamma_j^2 < \gamma_j^1$ (γ_k^l is the kth component of $\vec{\gamma}^l$, $l=1,2$.)

A layer allocation $\vec{\gamma}$ is maximally fair if it is feasible and if no other feasible layer allocation is fairer than $\vec{\gamma}$. There can be more than one maximally fair layer allocation vectors. One or more of these maximally fair layer allocation vectors are lexicographically optimal, as per Lemma 3[‡] stated below.

Lemma 3: If a layer allocation vector is lexicographically optimal, then it is maximally fair.

For example, both the layer allocation vectors $(1,0)$ and $(0,1)$ are maximally fair and lexicographically optimal in the network of Example IV.3.

The definitions for maxmin fairness, lexicographic optimality and maxmimal fairness of rate allocation vectors are the same as those for layer allocation vectors. A layer allocation vector is maxmin fair (lexicographic optimal/maximally fair) if and only if the corresponding rate allocation vector is maxmin fair (lexicographic optimal/maximally fair).

As we show later, maximally fair layer allocation exists and can be computed in polynomial complexity. One objective could be to allocate layers as per some maximally fair layer allocation. But, first we investigate, whether maximal fairness is a good notion of fairness. To this effect, we introduce the concept of pseudo-bottleneck links. This is analogous to the concept of bottleneck links for a continuous feasible set[14]. Consider a layer allocation vector $\vec{\gamma}$. A link l is said to be ***pseudo-bottlenecked*** with respect to a virtual session k traversing l, for layer allocation vector $\vec{\gamma}$, if the following conditions are met:

1. Capacity of the link is *almost* fully utilized, i.e., the difference between the capacity of the link and the sum of the rates allocated to the sessions travelling across the link must be less than b, i.e., $b\sum_{i\in n(l)}\Gamma_{il} > C_l - b$.
2. The virtual session has the maximum number of layers amongst all virtual sessions of the same session traversing the link, i.e., $\gamma_k = \Gamma_{\chi(k)l}$, where $\chi(k)$ is the session of virtual session k.
3. If the number of layers assigned to any other virtual session j traversing through link l is higher than that of virtual session k by two or more layers, then the number of layers assigned to virtual session j is less than or equal to the minimum number of layers required for some virtual session in $m(\chi(j),l)$ (set of virtual sessions traversing link l and belonging to the same session as virtual session j). Formalizing this, if $\gamma_j > \iota_{\chi(j)l}$, where $\iota_{il} = \max_{j\in m(i,l)} \iota_j$ then $\gamma_j \leq \gamma_k + 1$. This means that the difference between number of layers assigned to some other virtual session j and virtual session k can exceed 1 only because of the minimum layer constraint of the session of j. If there were no minimum number of layers requirement, the number of layers assigned to any other virtual session j traversing through link l would not exceed that of virtual session k by more than one.

Let $\vec{r}^{\gamma}$ be the rate vector for layer allocation $\vec{\gamma}$. $r_j^{\gamma} = b\gamma_j$. $\lambda_{il}^{\gamma} = \max_{j\in m(i,l)} r_j^{\gamma} = b\Gamma_{il}$. Let μ_s be the bandwidth

‡Lemmas 1 and 3 hold for arbitrary feasible sets of $k-$dimensional vectors, for any integer k and not just for layer allocation vectors. Hence they hold for rate allocation vectors as well.

consumed by the minimum number of layers necessary for virtual session s, i.e., $\mu_s = b\iota_s$, $\mu_{il} = \max_{s \in m(i,l)} \mu_s = b\iota_{il}$. The pseudo-bottleneck conditions can be translated in terms of rate conditions as follows. A link l is said to be *pseudo-bottlenecked* with respect to a virtual session k traversing l, for layer allocation vector $\vec{\gamma}$, if the following conditions are met:

$$\begin{aligned} \sum_{i \in n(l)} \lambda_{il}^{\gamma} &> C_l - b \\ r_k^{\gamma} &= \lambda_{\chi(k)l}^{\gamma} \\ \text{If } r_j^{\gamma} > \mu_{\chi(j)l}, \text{ then } r_j^{\gamma} &\leq r_k^{\gamma} + b. \end{aligned}$$

Example IV.4: Consider the network of Figure 1. Let every layer consume 1 unit of bandwidth each ($b = 1$). Ignore the maximum rate constraints. Consider the layer allocation $(4, 2, 3)$. Link e_1 is pseudo-bottlenecked w.r.t. virtual sessions 1, 3 and link e_3 is pseudo-bottlenecked w.r.t. virtual session 2. Now consider the layer allocation vector $(4, 4, 1)$. Virtual session 3 does not have a pseudo-bottleneck link. This is because it traverses through links e_1, e_3 and e_6. Virtual session 2 traversing through link e_3 has 3 more layers than virtual session 3 and does not have a minimum number of layers requirement. This violates pseudo-bottleneck condition (3). Total bandwidth consumed by the sessions traversing through link e_1 is 5 units in all but the capacity of the link is 7 units, 2 units more than the capacity utilized and $b = 1$. Similarly, utilized capacity of link e_6 is 1 unit, while actual capacity of e_6 is 6 units. This violates pseudo-bottleneck condition (1). However link $e_2(e_3)$ is pseudo-bottlenecked w.r.t. virtual session 1(2.)

Lemma 4 (pseudo-bottleneck lemma) A feasible layer allocation vector is maximally fair iff every virtual session has a pseudo-bottleneck link.

This lemma shows that maximal fairness is a good notion of fairness. Note that 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 at the expense of one or more virtual sessions which have at most one layer more than s. From the definition of a pseudo-bottleneck link, if there is no minimum number of layers requirement for any virtual session, every virtual session will be assigned at least $\lceil \left(\frac{C_l}{|n(l)|b} - (1 - |n(l)|^{-1}) \right) \rceil$ layers for some link l in its path, where $n(l)$ is the set of sessions traversing link l. This indicates that every virtual session is guaranted a bandwidth close to the fair share of capacity for at least one link on its path. For large C_l, a virtual session gets at least $\lceil \frac{C_l}{|n(l)|b} \rceil$ (approximately) layers for some link l on its path. In general, networks have large capacities. Thus maximally fair allocation is fair to every virtual session in some sense. In presence of minimum rate requirements, every virtual session s is guaranted $\lceil \left(\frac{C_l - \sum_{i \in \tau(l)} \mu_{il}}{|n(l) \setminus \tau(l)| b} - (1 - |n(l) \setminus \tau(l)|^{-1}) \right) \rceil$ layers, for some link l on its path, where $\tau(l)$ is the set of sessions traversing link l with minimum session rate in link l exceeding the rate of virtual session s ($\tau(l) \subset n(l)$). 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. This lemma also 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. The definitions of bottleneck links are similar in both cases.

Maximal fairness has other nice properties. As Lemma 3 shows, a lexicographically optimal layer allocation belongs to the set of maximally fair layer allocations. Also by definition, a maxmin fair layer allocation, (if one exists), is fairer than all other feasible layer allocations. Thus if a maxmin fair layer allocation exists, it is the only maximally fair layer allocation. So 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 our discrete feasible set, it may be different from the maxmin fair rate allocation in the continuous feasible set (i.e., feasible set when the rate allocations must satisfy only the capacity and minimum rate constraints). Technical report [15] furnishes an example to this effect. 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 continuous feasible set of [14]. 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 present a polynomial complexity algorithm for computation of a maximally fair layer allocation in the next section. The corresponding rate allocation is also maximally fair.

V. Algorithm for Computation of Maximally fair Layer Allocation

An algorithm that would obtain a maximally fair layer allocation is not entirely obvious. For example, let every layer consume b units of bandwidth and r_s^c be the maxmin fair bandwidth allocated to virtual session s if we can allocate any rate subject to the capacity and the minimum rate constraints. Maxmin fair rates can be computed using an algorithm presented in [14] in this case. It is tempting to think that any maximally fair rate allocation will allocate γ_s layers to the sth virtual session, where $\gamma_s \in \{\lfloor \frac{r_s^c}{b} \rfloor, \lceil \frac{r_s^c}{b} \rceil\}$. Technical report [15] furnishes a counter example to indicate that it is not always so. Another intuitively appealing algorithm is as follows: obtain the maxmin fair rates assuming the feasible set to be continuous (as [14] assumes). Initially allocate $\lfloor \frac{r_i^c}{b} \rfloor$ layers to virtual session i, $\forall i$, where r_i^c is the maxmin fair rate of virtual session i, under continous set assumption. Now try to add a layer to virtual session $1, \ldots, M$, if there are M virtual sessions, in some predetermined order (could be increasing order of the layers allocated) without decreasing the number of layers allocated to any virtual session any time. Repeat this process as long as the number of layers allocated to a virtual session i can be increased without decreasing that allocated to some other virtual session j, for some i. It appears that this algorithm will always yield a maximally fair layer allocation. However that is not quite true. Again technical report [15] contains a counter example. Counter examples in [15] indicate that it is possible that all maximally fair layer allocations have $\gamma_i < \lfloor \frac{r_i^c}{b} \rfloor$ or $\gamma_i > \lceil \frac{r_i^c}{b} \rceil$ for some virtual session i. However, if 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 networks. An example in technical report [15] 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 ahead the right order. Trying all possible orders will yield an exponential complexity algorithm in the worst case. However there exists a polynomial complexity algorithm for computation of a maximally fair layer allocation.

$n(l)$, $m(i,l)$, $\chi(s)$ and μ_{il} are as defined in pages 3, 3, 5 and 6 respectively. We introduce some additional terminologies.

L_s is the set of links traversed by virtual session s.

A virtual session is *saturated* under a rate vector if it traverses a link in which the session rate is equal to its bandwidth 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 travelling through the link l are saturated.

$\eta_l(k)$ denotes the link control parameter of link l at the end of the kth iteration. Link control parameter is an iterate used in computation of the maximally fair rates. It is an estimate of the fair share of the bandwidth of the link which can be allocated to the unsaturated virtual sessions traversing the link.

$\eta_{il}(k)$ denotes the session link parameter of session i traversing through link l. It is the bandwidth assigned to session i, if there were no bandwidth constraints for any of its virtual sessions on other links and feasible rate allocations need only satisfy capacity and minimum rate constraints (i.e., feasible rates need not be multiple of b).

$\omega_s(k)$ is the bandwidth that is assigned to virtual session s at the end of the kth iteration if it is restricted to receive no more than any of its session link control parameters, $\eta_{\chi(s)l}(k)$, on its path. Here, it is the largest multiple of b not exceeding its minimum session link control parameter on its path.

$\Omega_{il}(k)$ is the bandwidth allocated to session i in link l, under the rate vector $\vec{\omega}(k)$. $\Omega_{il}(k) = \max_{j \in m(i,l)} \omega_j(k)$.

$r_s(k)$ is the bandwidth allocated to virtual session s at the end of the kth iteration. $\vec{r}(k)$ denotes the rate vector at the end of the kth iteration, with components, $r_s(k)$.

$\lambda_{il}(k)$ is the rate allocated to the session i on link l at the end of the kth iteration. It is actually the maximum of the rates allocated to the virtual sessions in $m(i,l)$ at the end of the kth iteration.

$\Lambda(k)$ is the set of virtual sessions which are saturated w.r.t. rate allocation $\vec{\omega}(k)$

$$\begin{aligned}\Lambda(k) &= \{s : \exists\, l \in L_s, \omega_s(k) = \Omega_{\chi(s)l}(k), \\ &\quad \sum_{i \in n(l)} \Omega_{il}(k) > C_l - b\}\end{aligned}$$

$S(k)$ denotes the set of unsaturated virtual sessions at the end of the kth iteration.

$\Xi_l(k)$ denotes the set of unsaturated sessions traversing link l at the end of the kth iteration.

$F_l(k)$ denotes the total bandwidth consumed by the saturated sessions traversing link l at the end of the kth iteration.

$\vec{\gamma}$ is the output layer allocation vector.

The algorithm follows.

1. $k = 0$, $\eta_l(0) = 0$, $F_l(0) = 0$, $\Xi_l(0) = n(l)$ $\forall$ 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. $k \to k+1$
3. For every link l in the network compute the link control parameter. If $\Xi_l(k-1) \neq \phi$, then $\eta_l(k)$ is the maximum possible θ, which satisfies the equation, $F_l(k-1) + \sum_{i \in \Xi_l(k-1)} \max(\theta, \lambda_{il}(k-1)) = C_l$ else $\eta_l(k) = \eta_l(k-1)$. For all unsaturated virtual sessions s passing through link l, $(s \in S(k-1) \cap (\cup_{i \in \{1,\ldots N\}} m(i,l)))$, $\eta_{\chi(s)l}(k) = \max(\eta_l(k), \lambda_{\chi(s),l}(k-1))$ (N is the number of sessions.)
4. Compute $\omega_s(k)$ for all virtual sessions $s \in S$, where $\omega_s(k) = b\lfloor \frac{\min_{l \in L_s} \eta_{\chi(s)l}(k)}{b} \rfloor$, if $s \in S(k-1)$, else $\omega_s(k) = r_s(k-1)$.
5. For every link l in the network, compute $\Omega_{il}(k)$ for 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 $\vec{\omega}(k)$.

$\Lambda(k) = \{s : s \in S(k-1), \exists\, l \in L_s, \omega_s(k) = \Omega_{\chi(s)l}(k), \sum_{i \in n(l)} \Omega_{il}(k) > C_l - b\}$.

7. If $\Lambda(k) \neq \phi$, compute the rates allocated to the virtual sessions after the kth iteration, via, $r_s(k) = \omega_s(k)$, $\forall s$ and go to step (9).
8. If possible, find a virtual session $s \in S(k-1)$, s.t.

$$\begin{aligned} \omega_s(k) &< \min_{l \in L_s} \eta_l(k) \text{ and} \\ \Omega_{il}(k) &\geq b\lfloor \frac{\eta_l(k)}{b} \rfloor \; \forall i \in \Xi_l(k-1) \text{ if } l \in L_s \text{ and} \\ & \min_{l_1 \in L_s} \lfloor \frac{\eta_{l_1}(k)}{b} \rfloor = \lfloor \frac{\eta_l(k)}{b} \rfloor. \end{aligned}$$

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 in the network compute the session rate in link l, for every session in $n(l)$, $\lambda_{il}(k) = \max_{s \in m(i,l)} r_s(k)$.
10. Compute the set of virtual sessions unsaturated after the kth iteration, $S(k) = S(k-1) \setminus \{s : \exists\, l \in L_s, \text{ s.t. } \sum_{i \in n(l)} \lambda_{il}(k) > C_l - b \text{ and } r_s(k) = \lambda_{\chi(s)l}(k)\}$.
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: $\Xi_l(k) = \{n : n \in \{1, \ldots, N\}, m(n,l) \cap S(k) \neq \phi\}$.
13. For every link l, for which $\Xi_l(k) \neq \phi$, compute the bandwidth consumed by the saturated sessions passing through link l, $F_l(k) = \sum_{i \in n(l) \setminus \Xi_l(k)} \lambda_{il}(k)$.
14. Go to step (2).

At every iteration k, the algorithm computes a "fair share" of the link bandwidth for every session, i, the session link parameter, $\eta_{il}(k)$. Since a virtual session can have rates only in multiples of b, bandwidth $b\lfloor \frac{\eta_{il}(k)}{b} \rfloor$ is offered to all virtual sessions of session i traversing the link. Every session releases the remaining bandwidth i.e., $\eta_{il}(k) - b\lfloor \frac{\eta_{il}(k)}{b} \rfloor$. If no virtual session saturates in the current iteration, then this residual bandwidth may be used to increment the rate of some virtual session traversing this link. If a virtual session is constrained to have a rate less than its fair share because it is assigned a lower bandwidth on another link, then it can not use some of this bandwidth. If all virtual sessions of the same session release some bandwidth because of constraints on other links, then there is some more residual bandwidth. The residual bandwidth is split fairly among other sessions in the next iteration and the process continues. We describe this process in greater details below.

Initially all the link control parameters are assigned zero values. The rates of the virtual sessions are initialized to the respective minimum rates. All sessions and virtual sessions are unsaturated. Next the algorithm computes the link control parameters as per step (3). If there are no minimum rate requirements, the link control parameter for link l at the first iteration is the capacity of the link per session traversing the link. The next step is to compute session link parameters for every session at every link. Session link control parameter for a session traversing a link is the maximum of the link control parameter and the previous iteration session link rate. If a virtual session traversing link l, had no bandwidth constraint on other links, then it is assigned a rate equal to the greatest multiple of b not exceeding its session link control parameter. On account of the bandwidth constraint in other links, the virtual session gets a rate equal to the greatest multiple of b not exceeding the minimum of its session link parameters on its path. This ensures that virtual session rates are multiples of b. If no virtual session is saturated, $(\Lambda(k) = \phi)$, then try to find a virtual session s which satisfies the properties mentioned in step (8). At least one such virtual session exists in this case[15]. Increment the rate of such a virtual session by b. This is done because otherwise, the algorithm can continue forever. This is because in the next iteration, the same link control parameter will be computed and the process repeats again and again. A session is saturated if all its virtual sessions are saturated. The bandwidth consumed by the saturated sessions, if any, are computed. This bandwidth is subtracted from the link capacity, and the link control parameters are recomputed at the beginning of every iteration as per step (3) and the process continues. The algorithm determines which virtual session and sessions are saturated. A new iteration starts if there are still unsaturated virtual sessions. Either the rate of at least one virtual session increases by b units or the number of unsaturated virtual session decreases by at least one, at the end of every iteration[15]. Neither of these two can continue indefinitely. So the algorithm terminates in finite number of iterations (at most $|\mathcal{L}|M$ iterations, $\mathcal{L}$ is the set of all links, M is the number of virtual sessions). Upon termination, we have a maximally fair rate vector and the corresponding layer allocation is maximally fair as well. An example illustrating the operation of the algorithm follows.

Example V.1: Consider the network of Example IV.4. $L_1 = \{e_1, e_2, e_4\}$, $L_2 = \{e_1, e_3, e_5\}$, $L_3 = \{e_1, e_3, e_6\}$. Link control parameters are as follows. $\eta_{e_1}(1) = 3$, $\eta_{e_2}(1) = 4$, $\eta_{e_3}(1) = 2.5$, $\eta_{e_4}(1) = 4$, $\eta_{e_5}(1) = 4$, $\eta_{e_6}(1) = 6$. Now, the session link control parameters are as follows. $\eta_{1e_1}(1) = 4$, $\eta_{2e_1}(1) = 3$, $\eta_{1e_2}(1) = 4$, $\eta_{1e_3}(1) = 2.5$, $\eta_{2e_3}(1) = 2.5$, $\eta_{1e_4}(1) = 4$, $\eta_{1e_5}(1) = 4$, $\eta_{2e_6}(1) = 6$.

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 session 2 and 3 are not saturated w.r.t. $\vec{r}(1)$. $S(1) = \{2,3\}$. $\Xi_{e_1}(1) = \Xi_{e_3}(1) = \{1,2\}$, $\Xi_{e_5}(1) = \{1\}$, $\Xi_{e_6}(1) = \{2\}$ and $\Xi_l(1) = \phi$, if $l \in \{e_2, e_4\}$. $F_{e_l}(1) = 0$, 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 control 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 choose virtual session 2 for incrementation arbitrarily. It follows that $r_1(2) = 4, r_2(2) = 3, r_3(2) = 2$. This saturates both virtual sessions 2 and 3. All virtual sessions are saturated and the algorithm terminates.

The following theorems prove that the algorithm terminates in a finite number of iterations and yields a maximally fair layer allocation upon termination. If there exists a maxmin fair layer allocation vector, then the output layer allocation vector is maxmin fair.

Theorem 1 (Maximal-Fairness Theorem) If the algorithm terminates, then the output layer allocation vector $\vec{\gamma}$ is

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

We give the gist of the proof here. We first show that the rate allocation at the end of every iteration is feasible. We show that if a virtual session saturates in an iteration, then it has a pseudo-bottleneck link in all subsequent iterations. Maximal fairness of the final rate allocation vector and hence the output layer allocation vector follows from the pseudo-bottleneck lemma (Lemma 4), since the algorithm terminates only when all virtual sessions saturate. The last part of the theorem follows from the observation made in Section IV that any algorithm which outputs a maximally fair layer allocation, outputs a maxmin fair layer allocation, if one such exists. The following theorem proves that the algorithm terminates in finite number of iterations.

Theorem 2 (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.
The idea behind the proof is as follows. We show that in every iteration either the number of unsaturated virtual sessions decrease by at least one or the rate of an unsaturated virtual session increase by at least b units as compared to that in the previous iteration. The first can take place for at most M iterrations. We show that the second can take place for at most $|\mathcal{L}|M$ iterations. The result follows.

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 $O(|\mathcal{L}|^2M^2)$.

The choice of an unsaturated virtual session s for possible incrementaion of rate is crucial for attaining maximal fairness in the end. Technical report [15] has an example which shows that if $\Lambda(k) = \phi$, and the rate of a virtual session s is incremented by b units but it does not satisfy the requirements of Step (8) then the output layer allocation may not be maximally fair.

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. If no receiver subscribes to a particular layer, then the source does not transmit it.

VI. Conclusion and Discussion

Summarizing, this is the first study in fairness with the assumption that bandwidths can be allocated in discrete chunks only. This makes the feasible set discrete. Fairness in a discrete feasible set is vastly different from that in a continuous feasible set and have not been explored before, in multicast, or even unicast scenario. Maxmin fair rate allocation may not exist in this case. We have shown that computation of lexicographically optimal layer allocation is NP-hard. We have introduced the notion of maximally fair rate allocation and shown that a maximally fair layer allocation has many nice properties with respect to fairness. We have presented a polynomial complexity algorithm for computation of a maximally fair rate allocation. Our results apply to both internet and ATM like networks. We present some interesting topics for future research.

A lexicographically optimal layer allocation is the best one can hope for, when the feasible set is discrete. However, as we have shown, the computation of a lexicographically optimal layer allocation for discrete bandwidth layers is an NP-hard problem. Developing computationally simple approximation algorithms for allocation of rates close to a lexicographically optimal layer allocation w.r.t. some useful metric may form interesting topics of future research. It is not even known whether there can exist such algorithms. Good heuristics for this purpose may also be interesting.

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, [10] assumes that every layer consumes 32 kb/s bandwidth. However, this assumption does not hold in all coding schemes. There may be more complicated situations where the hierarchical signal structure of some sessions is flexible while it is predetermined for some others. For instance, the service rates of receivers of some sessions have a continuous range, while those of receivers of some other sessions can only take discrete values. Also, it is possible that layer bandwidths can be fine tuned in certain ranges, while the granulartity is coarse in other ranges. Fairness in these scenarios becomes more technical, and is beyond the scope of the current paper. We have considered this more general case in [13]. In fact, [13] studies a more general fairness objective, namely, fairness of utilities. End users are more concerned about quality of service rather than service rates. So, we considered utility functions connecting rates with quality of service. The utility of a bandwidth can be bandwidth itself, layers allocated, perceptual quality or distortion. The utility function need not be continuous nor strictly increasing. In general, bandwidth allocation will be very different for different fairness objectives and the fairness objective can be decided on a case by case basis only. We presented an algorithm for computing the maximally fair allocation of utilities, for arbitrary utility functions. Since layers can be considered utilities of bandwidth, the algorithm of [13] applies to fair allocation of discrete bandwidth layers as well. However, this algorithm is 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 [13].

Our algorithm is amenable to distributed implementation. The criteria for determination of rate of a virtual session uses information along the path of the virtual session mainly. The only place where the algorithm uses global information is that $r_s(k) = \omega_s(k) + b$, for at most one virtual session, s. This feature of the algorithm is not crucial to the proof of maximal fairness of the output and is a matter of convenience. This increase in rate can be carried out for multiple virtual sessions, subject to feasibility and as long as they satisfy the criteria of step (8) and the algorithm will still output a maximally fair layer allocation. However, the details for the distributed implementation need to be worked out.

VII. Acknowledgement

The authors wish to thank S. Khuller for discussions on the proof for NP-hardness of lexicographically optimal allocation.

References

[1] E. Amir, S. McCanne and H. Zhang An Application Level Video Gateway *Proceedings of ACM Multimedia' 95*, pp 255 – 265, San Francisco, CA, November 1995. ACM.

[2] T. Ballardie, P. Francis, and J. Crowcroft. Core based trees: an architecture for scalable inter-domain multicast routing, *Proceedings of ACM SIGCOMM,* Sept. 1993, pp. 85-95

[3] D. Bertsekas and R. Gallager, *Data Networks*, Englewood Cliffs, NJ:Prentice-Hall, 1987.

[4] T. Bially, B. Gold, and S. Seneff. A technique for Adaptive Voice Flow Control in Integrated Packet Networks, *IEEE Transactions on Communications,* Vol. 28, No. 3, March 1980, pp. 325 – 333

[5] Z. Cao and E. Zegura. Utility Max-Min: An Application-Oriented Bandwidth Allocation Scheme *IEEE INFOCOM'99*

[6] S. Deering and D. Cheriton. Multicast routing in datagram internetworks and extended LANs, ACM Transactions on Computer Systems, vol 8, no. 2, pp. 54-60, Aug. 1994.

[7] X. Li, S. Paul and M. Ammar. Layered Video Multicast with Retransmission (LVMR): Evaluation of Hierarchical rate control, *Proceedings of IEEE Infocom' 98*, March 1998

[8] 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 Institute of Technology, '98

[9] S. McCanne, Scalable Compression and Transmission of Internet Multicast Video. *PhD thesis*, Univ. of California, Berkeley, December 1996.

[10] S. McCanne, V. Jacobson and M. Vetterli. Receiver-Driven Layered Multicast, *Proceedings of ACM SIGCOMM '96*, Stanford, CA, September 1996

[11] D. Rubenstein, J. Kurose and D. Towsley. The Impact of Multicast Layering on Network Fairness *Proceedings of ACM SIGCOMM '99, Cambridge, MA, September,* 1999

[12] S. Sarkar and K. N. Sivarajan: Fairness in Cellular Mobile Networks *Proceedings of the 34th Annual Allerton Conference on Communication, Control and Computing,*1996, *pp* 457 – 469

[13] S. Sarkar and L. Tassiulas: Fair Allocation of Utilities in Multirate Multicast Networks *Proceedings of the 37th Annual Allerton Conference on Communication, Control and Computing,*1999

[14] S. Sarkar and L. Tassiulas: Distributed Algorithms for Computation of Fair Rates in Multirate Multicast Trees *Proceedings of IEEE INFOCOM'* 2000, Tel Aviv, Israel, March' 2000

[15] S. Sarkar and L. Tassiulas: Fair Allocation of Discrete Bandwidth Layers in Multicast Networks *Technical Report* TR 99 – 43, Institute for Systems Research and University of Maryland, 1999.

[16] T. Turletti and J. C. Bolot Issues with multicast video distribution in heterogeneous packet networks, *Packet Video Workshop*, '94, Portland.

[17] D. Taubman and A. Zakhor. Multirate 3-D Subband Coding of Video, *IEEE Transactions on Image Processing*, Vol 3, No. 5, September 1994.

[18] K. M. Uz, M. Vetterli and D. J. LeGall. Interpolative Multiresolution Coding of Advanced Television with Compatible Subchannels, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol 1, No. 1, March 1991.

[19] M. Vishwanath and P. Chou. An efficient algorithm for hierarchical compression of video. *Proceedings of IEEE International Conference on Image Processing,* Austin, TX, November, 1994.