Layered bandwidth allocation for multicasting of hierarchically encoded sources*

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 of resource allocation 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 when receivers can not subscribe to fractional layers. Now, the network can allocate a discrete set of bandwidths to a receiver, whereas a continuous set of rates can be allocated when receivers can subscribe to fractional layers. Fairness issues vastly differ 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. 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.

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 consume significant bandwidth, and tax the network resources severely. Also, the users need to adapt to available bandwidth of the network, which fluctuates from time to time. For this purpose, 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 of quality of the received signal and additional bandwidth is needed to transport the combined stream. When a layer is dropped, there is graceful degradation in reception quality.

*A part of this paper has been presented at INFOCOM' 2000.
†Corresponding Author

Every receiver would like to receive as many layers as possible. However, networks have resource limitations, and delivering a large number of layers would cause acute congestion. Fairness of resource allocation becomes important in this situation.

Fair allocation of service does not penalize any user excessively, and naturally allocates a certain minimum quality of service to all users. User satisfaction is often a concave function, i.e., satisfaction increases rapidly with service rate increase at small service rates, and increases slowly with service rate increase at high service rates. Thus total user satisfaction is often improved if all users obtain an equitable quality of service, rather than some users obtaining high quality of service at the expense of others obtaining low quality of service. Also, networks often provide class based service. Users belonging to the same class have the same service agreements and pay the same revenues. In this case, fairness amongst members of the same class is essential. We would investigate fair allocation of service rates in multirate multicast networks.

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 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.

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 [19]. 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[19]. 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[4] 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[16][19]. 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"

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

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)[15] and LVMR (Layered Video Multicast with Retransmissions)[12] do not handle fairness among sessions very well, when there are multiple sessions competing for bandwidth[13]. An alternative layer allocation scheme has been proposed in [13]. 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 [13] 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[16]. But, [16] 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, [16] 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.

This paper is organized as follows. Section 2 discusses the network scenarios under which the assumption of a discrete feasible set of rate allocations becomes a necessity. Section 3 describes the network model we assume. Section 4 develops the notion of fairness in the discrete case. We present fairness definitions and properties which apply to any discrete feasible set in general, and to the layer allocation problem in particular in Section 4. Section 5 presents fairness properties specific to the layer allocation problem. This section also presents an algorithm for computation of the maximally fair rates. Section 6 identifies some directions for future research. This section concludes the paper.

2 Constraints on Signal Structure

We discuss the network scenarios under which the feasible set of rate allocations becomes discrete. Since this paper is a logical successor of [19], we discuss the contributions of [19] briefly. It assumes that a feasible rate allocation is one which satisfies the capacity constraints and the minimum and maximum rate constraints. Capacity constraint is as follows. Total rate allocated to a session in a

link is equal to the maximum of the rates of the session receivers downstream. Sum of the session rates on a link should not exceed the link capacity. There is a minimum rate at which a receiver must receive service. A receiver may not be able to receive service at a rate greater than a certain value, due to processing speed constraints. So the rate allocated to a receiver should be between its minimum and maximum allowed rate. These are the minimum and maximum rate constraints. Now, [19] gives a polynomial time algorithm for computing the maxmin fair rate allocation in the reference feasible set described above. It advocates several methods for allocation of the fair rates, once they are computed. Transcoding is one such option. It is assumed that video gateways are placed throughout the network to transcode signal into a lower bit rate coding, using either the same coding format with different parameters or a different coding scheme altogether. Source transmits at a rate equal to the maximum of the fair rates allocated to its receivers. At forking points, video gateways 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.

The other possibility is to have an adaptive hierarchical encoding. In this approach, an information stream is partitioned in to a base layer, comprising the information needed to represent the lowest fidelity media and a number of enhancement layers. Some number of these enhancement layers are combined by the decoder with the base layer to recover a signal of incremental fidelity. As each layer is added, there is an improvement of quality of the received signal and additional bandwidth is needed to transport the combined stream. Hierarchical coding was first suggested for packet voice transmission[5]. Subsequently, several hierarchical coding schemes have been proposed for video, [1], [9], [22], to name a few. Here, [19] assumes that the source coder is programmable, and the source can modify its signals hierarchical structure. Once the fair 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. Each receiver is allotted layers as per its fair share of bandwidth. Total number of layers of a session transmitted across a link is equal to the maximum number of layers allotted to the session receivers downstream of the link. As we would discuss below, a perfect match between the layer bandwidths and the maxmin fair receiver rates is not always feasible due to various reasons. The following approach is suggested in [19] in cases of fixed or partially adaptive signal hierarchy. Allocate to the receivers as many layers as permitted by the computed fair rate. If the total bandwidth consumed by the layers allocated to a receiver is strictly less than the fair 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. Figure 1 illustrates this concept.

Now we discuss the cases where the assumptions of [19] do not apply.

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[21]. Output rates for video gateways can be fine tuned to match the required receiver rates, if rate adaptive video gateways are used[2]. However, transcoding places additional computational and administrative burden on the network. 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 greater than 1 (number of outgoing links). This makes transcoding financially prohibitive. For

transcoding, internal nodes must perform intensive computational tasks and may need to process arbitrary coding algorithms. This increases end to end delay. More importantly, transcoding can not be applied to a secure communication without entrusting the network with the encryption key. Depending on the security risks, this may be totally unacceptable. In these cases, network heterogeneity 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[5]. Subsequently, several hierarchical coding schemes have been proposed for video, e.g., [1], [9], [22], to name a few. Layer structure may be fixed and predetermined because of several reasons. 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)[24]. In certain video codes, substreams can be extracted to produce a specific range of resolutions only, e.g., 3D Subband Video Coding[22]. Certain coding schemes are particularly successful only when some apriori structure or hierarchy can be found in the problem[23], because certain computationally expensive optimizations, e.g., in quantizer thresholds, can be carried out offline. Some codecs, e.g., PVH codec of [14] 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 a perfect match between the layer bandwidths and desired receiver rates is not always feasible due to various reasons. As we have discussed before, [19] advocates allocating an extra layer to a receiver, and letting the network drop a certain percentage of packets of the highest layer. But, 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 if they have residual bandwidth (like u_2 in Figure 1 subscribing to 2 layers, when its fair share is 1.5 units), the network will be congested and the routers will indiscriminately drop packets of all the layers. An important characteristic of hierarchical encoding is that the layers are generally ordered in some manner. An enhancement layer yields useful information only when the base layer and lower enhancements layers have been successfully decoded. So loss of packets of all the layers will adversely affect signal quality. Its 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[15]. 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 [7]. 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[20]. So [7] does not apply to our case.

3 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[8], CBT[3], 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 2 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 in the entire network. A layer allocation vector $\vec{\gamma}$ is an $M-$dimensional vector, with components γ_j (i.e., a layer allocation vector gives the layer allocations for all virtual sessions in the network).

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

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. $\iota_j \leq \gamma_j \leq p_j$, $\forall\ j$, $p_j \geq \iota_j \geq 0$.
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 $\vec{r}^{\gamma} = (r_1^{\gamma}, \ldots, r_M^{\gamma})$, defined as $r_j^{\gamma} = b\gamma_j$ is a feasible rate allocation. Feasibility conditions for rate and layer allocation vectors are equivalent, and there is a one to one correspondence between the set of feasible rate and layer allocation vectors. We will use λ_{il} to denote 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., $\lambda_{il} = \max_{j \in m(i,l)} r_j$. Figure 2 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.

We would like to comment on our assumption that all multicast packets of the same session move along the same tree. Different multicast layers are perceived as different multicast groups and trees for different groups may be completely different in general. However, trees for different multicast layers of the same session will remain the same if source rooted trees are used, as all these layers (multicast groups) have the same source. Besides, trees for different multicast layers of the same session should not differ very much, as that would complicate reconstruction of information at the receiver. For example, if we consider video transmission, and different layer packets of the same session traverse along different multicast trees, then different layer packets for the same frame may arrive at a receiver at different times, and frame reconstruction will involve a lot of packet reordering. This may incur an unacceptable delay jitter. Thus it may be a good idea to use source rooted trees in this case. Many video coders make the same assumption, e.g., [6].

4 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 and then we mention certain fairness properties for

discrete feasible sets. We emphasize that 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. However, for simplicity we mention layer allocation vectors explicitly and present examples for layer allocation vectors only.

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 4.1: The maxmin fair layer allocation vector in the network of Figure 2 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 4.2: A lexicographically optimal layer allocation vector in the network of Figure 2 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.[7]. Our definition of maxmin fairness have been suggested in [4]. 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.

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

same feasible set.

We will prove this lemma later in this section.

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. Refer to figure 3 for more insight.

If the feasibility set were continuous, as in [19], 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, computation of lexicographically optimal vector is an NP-hard problem in many feasible sets. We will prove this result in the next section for the layer allocation problem we are studying.

In view of this unfortunate result, we introduce a different notion of fairness, *maximal fairness.* We use the concept of *relative fairness* introduced in [17]. 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 2 stated below.

Lemma 2 *If a layer allocation vector is lexicographically optimal, then it is maximally fair.*

For example, both the layer allocation vectors $(1, 0, 3, 3)$ and $(0, 1, 2, 4)$ are maximally fair in the network of Figure 3. Applying the definitions of relative fairness, no other layer allocation vector is fairer than these two. However, $(1, 0, 3, 3)$ is lexicographically optimal, whereas $(0, 1, 2, 4)$ is not (but $(1, 0, 3, 3)$ is not fairer than $(0, 1, 2, 4)$). We will prove Lemma 2 later in this section.

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.

Maximal fairness has many nice 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. As Lemma 2 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). Refer to figure 5 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 continuous feasible set of [19]. 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.

Now we prove Lemmas 1 and 2. We first present Proposition 1 and Lemma 3 which we would require in proving lemmas 1 and 2.

Proposition 1 *Consider two $M-$dimensional vectors $\vec{A}$ and $\vec{B}$. If $\vec{A}$ is fairer than $\vec{B}$ then 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.*

This proposition has been proved in [17].

Lemma 3 *If $\vec{A}$ is fairer than $\vec{B}$, then $\vec{A}$ is lexicographically greater than $\vec{B}$.*

Proof of Lemma 3: For lexicographical comparison, we can ignore the components of $\vec{A}$ and $\vec{B}$ which are componentwise equal. Wlog $a_j \neq b_j$ for all components j. Let $\vec{A}$ be fairer than $\vec{B}$. It follows from Proposition 1 that there exists a component s such that $a_s = \min_j a_j$ and $a_s > b_s$. Clearly $b_s \geq \min_j b_j$. Thus $\min_j a_j > \min_j b_j$. Thus the lemma follows. □

Now we prove lemmas 1 and 2 using lemma 3.

Proof of Lemma 1: If $\vec{A}$ is maxmin fair, then by definition of maxmin fairness and maximal fairness, $\vec{A}$ is fairer than any other feasible vector. Lexicographical optimality follows from Lemma 3.□

Proof of Lemma 2: Let $\vec{A}$ be lexicographically optimal in a feasible set, but not maximally fair. Thus there exists another feasible vector $\vec{B}$, fairer than $\vec{A}$. It follows that $\vec{B}$ is lexicographically greater than $\vec{A}$ (by Lemma 3). This contradicts the lexicographic optimality of $\vec{A}$. □

5 Fair Allocation of Discrete Bandwidth Layers

We have studied fairness properties which apply to any discrete subset in the previous section. Hence, these fairness properties apply to the layer allocation problem as well. Now, we will study the fairness problem in the resource allocation problem in particular. We will present fairness properties which are specific to the resource allocation problem in the first subsection. We will show that computation of a lexicographically optimal layer allocation is NP-hard. Next we will present another intuitively appealing fairness property of a maximally fair layer allocation. Finally, we will present a polynomial complexity algorithm for computing the maximally fair layer allocation in the next subsection.

5.1 Resource allocation specific fairness properties

We would first present the result that computation of lexicographically optimal layer allocation is NP-hard.

Theorem 1 (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. We expect this property to hold in other discrete feasible sets as well.

Proof of Theorem 1 (NP-hardness Theorem): Let there exist a fictitious blackbox which can compute the lexicographic optimal layer allocation for any network, as long as there exists a feasible layer allocation, the complexity being a polynomial function of the input size. We will show that this implies that $P = NP$.

Consider an undirected graph G with N vertices and $|E|$ edges. The vertices are numbered $1, \ldots, N$. We call an edge e_{ij} if it connects vertices i and j in G, $i < j$. Wlog there is no edge connecting vertex i to itself. Construct a directed network with N unicast sessions, numbered $1, \ldots, N$. For every edge e_{ij} in G, there exists two nodes v_{ij1}, v_{ij2} and an edge directed from v_{ij1} to v_{ij2}. In addition there is an edge directed from v_{ij2} to

1. v_{ik1}, if $k > j$ and there does not exist a vertex l in G such that $j < l < k$, and l is adjacent to i in G,

2. v_{kj1}, if $k > i$ and there does not exist a vertex l in G such that $i < l < k$ and l is adjacent to j in G,

3. v_{jk1}, if there does not exist an l in the adjacency set of j such that $i < l < k$.

Every edge has unit capacity. No session has any minimum number of layers requirement. Let $\{i_1, \ldots, i_p, k_1, \ldots, k_t\}$ constitute the adjacency set of j and $i_1 < i_2 < \ldots < i_p < j < k_1 < k_2 < \ldots < k_t$. The path of session j consists of nodes $v_{i_1 j1}, v_{i_1 j2}, v_{i_2 j1}, v_{i_2 j2}, \ldots, v_{i_p j1}, v_{i_p j2}, v_{jk_1 1}, v_{jk_1 2}, \ldots, v_{jk_t 1}, v_{jk_t 2}$ in the same sequence. Every layer consumes unit bandwidth. Note that

$$n(l) = \begin{cases} \{p, q\} & \text{if } l = (v_{pq1}, v_{pq2}) \\ \{p\} & \text{if } l = (v_{ps2}, v_{pu1}) \text{ or} \\ & l = (v_{sp2}, v_{up1}) \text{ or } l = (v_{ps2}, v_{up1}). \end{cases}$$

Every edge in G corresponds to two nodes in the network. Thus the network has $2|E|$ nodes, at most $4|E|^2$ links, and N sessions. Clearly the network can be constructed in polynomial time (construction complexity is a polynomial function of the input graph G size).

A layer allocation $\vec{\gamma}^S$ is defined as follows:

$$\gamma_j^S = \begin{array}{ll} 1 & j \in S \\ 0 & \text{otherwise.} \end{array}$$

We will prove that layer allocation $\vec{\gamma}^S$ is feasible iff S is an independent set in G. By construction, two sessions i, j share a link iff vertices i and j are adjacent in G. (Wlog $i < j$.) This follows because if i, j share a link then there exists a l such that $n(l) = \{i, j\}$ and this means that $l = (v_{ij1}, v_{ij2})$. This implies that there exists an edge between vertices i, j in G. If i and j are adjacent in G, then by construction, both sessions i, j traverse through link l, where $l = (v_{ij1}, v_{ij2})$.

Let S not be an independent set. Thus there exists i, j such that $i, j \in S$ and i, j are adjacent in G. Thus sessions i and j share a link, say l. $\gamma_i^S + \gamma_j^S = 2 > 1 = C_l$. This violates the capacity constraint for link l. Thus $\vec{\gamma}^S$ is infeasible.

Now let S be independent in G. We will show that $\vec{\gamma}^S$ is feasible. γ_i is an integer for all i. The minimum number of layer requirements is trivially satisfied. Consider any link l in the network. If $|n(l)| = 1$, then the capacity condition holds for link l for any $0-1$ layer allocation vector. If $|n(l)| \neq 1$, $|n(l)| = 2$. Let $n(l) = \{i, j\}$. Capacity condition is clearly satisfied if at most one of i, j is in S. Since sessions i and j share link l, i and j are adjacent in G. It follows that both i and j can not be in S, from independence of set S in G. Thus capacity condition holds for every link l in the network. Thus $\vec{\gamma}^S$ is feasible.

Note that any feasible layer allocation vector has $0-1$ components only, because every layer consumes 1 unit of bandwidth and edges have unit capacity. Thus every feasible layer allocation is of the form $\vec{\gamma}^S$, where S is some subset of $\{1, \ldots, N\}$. Now let $\vec{\gamma}$ be a lexicographic optimal layer allocation vector. It assigns 1 layer to maximum possible number of sessions. From previous arguement, $\vec{\gamma} = \vec{\gamma}^S$, where $S = \{i : \gamma_i = 1\}$. We will show that S is a maximum independent set. Since $\vec{\gamma}^S$ is feasible, $S = \{i : \gamma_i = 1\}$ is an independent set. Let S not be a maximum independent set in G. Thus there exists an independent set S_1 such that $|S_1| > |S|$. $\vec{\gamma}^{S_1}$ is a feasible layer allocation vector. It assigns 1 layer to $|S_1|$ sessions and 0 layer to the rest. Thus $\vec{\gamma}^{S_1}$ is lexicographically greater than $\vec{\gamma}^S$, which assigns 1 layer to $|S|$ sessions and 0 layer to the rest. This contradicts the lexicographic optimality of $\vec{\gamma}^S$. So S is a maximum independent set.

So given an undirected graph construct the network described above in polynomial time. The size of the network is a polynomial function of that of G. Since minimum number of layers requirement does not exist, the feasible set of layer allocations is nonempty. Use the blackbox to obtain a lexicographically optimal layer allocation $\vec{\gamma}$. Define $S = \{i : \gamma_i = 1\}$. From the arguement above, S is the maximum independent set in G. Thus the maximum independent set has been generated in polynomial time. Generation of maximum independent set of a graph G is a known NP-hard problem. Thus $P = NP$. □

In view of the nonexistence of maxmin fair layer allocation and the computational complexity for lexicographically optimal layer allocation,, maximally fair layer allocation is possibly the best one can attain. As we argued in the previous section, maximal fairness is a good notion of fairness. We shall present another intuitively appealing property of maximally fair allocation, which holds in the layer allocation problem we are studying. We first introduce the concept of pseudo-bottleneck links. This is analogous to the concept of bottleneck links for a continuous feasible set[19]. 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 $\vec{\gamma}$. $r_j^{\gamma} = b\gamma_j$. Let λ_{il}^{γ} be the bandwidth allocated to session i in link l, under layer allocation $\vec{\gamma}$. It is actually the maximum of the rates allocated to virtual sessions of session i traversing link l, i.e., $\lambda_{il}^{\gamma} = \max_{j\in m(i,l)} r_j^{\gamma} = b\Gamma_{il}$. Let μ_s be the bandwidth 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:

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

Refer to Figure 4 for an example illustrating the concept of pseudo-bottleneck links.

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

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

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.

We first outline the proof. First, we show that if a virtual session does not have a pseudo-bottleneck link under a feasible layer allocation vector $\vec{\gamma}^1$, then it is possible to construct another feasible layer allocation vector which is fairer. Thus, $\vec{\gamma}^1$ is not maximally fair. This proves the necessary part. Next, we show that if every virtual session has a pseudo-bottleneck link under a feasible layer allocation vector $\vec{\gamma}^1$, then it is not possible to construct another feasible layer allocation vector which is fairer. In other words, any other fairer layer allocation vector is infeasible. This proves the sufficiency. The formal proof follows.

Proof of Lemma 4 (Pseudo-Bottleneck Lemma): Let a virtual session s not have a pseudo-bottleneck link under a feasible layer allocation vector $\vec{\gamma}^1$. We will show that it can not be a maximally fair layer allocation vector. Let

$$\begin{aligned} Y &= \{l : l \in L_s, b \sum_{i \in n(l)} \Gamma^1_{il} \leq C_l - b\} \\ Z &= \{l : l \in L_s, \gamma^1_s < \Gamma^1_{\chi(s)l}\} \\ W &= \{j : \exists\, l \in L_j \cap (L_s \setminus (Y \cup Z)) \text{ s.t. } \gamma^1_j = \Gamma^1_{\chi(j)l} > \iota_{\chi(j)l},\ \gamma^1_j > \gamma^1_s + 1\}, \end{aligned}$$

Consider a layer allocation vector $\vec{\gamma}^2$ defined as follows:

$$\gamma^2_j = \begin{cases} \gamma^1_j + 1 & j = s \\ \gamma^1_j - 1 & j \in W \\ \gamma^1_j & \text{otherwise.} \end{cases}$$

We will prove that $\vec{\gamma}^2$ is a feasible layer allocation vector and is fairer than $\vec{\gamma}^1$. First we show the feasiblity. Since $\vec{\gamma}^1$ is feasible, γ^1_j is an integer for all j and hence γ^2_j is an integer for all j.

$\gamma^2_j \geq \gamma^1_j$, if $j \notin W$. Since $\gamma^1_j \geq \iota_j$ from feasibility of $\vec{\gamma}^1$, $\gamma^2_j \geq \iota_j$, if $j \notin W$. Now let $j \in W$. $\gamma^1_j > \iota_{\chi(j)l}$ for some $l \in L_j$. Thus $\gamma^1_j > \iota_{\chi(j)l} \geq \iota_j$. Since γ^1_j and ι_j are integers, $\gamma^1_j \geq \iota_j + 1$. It follows that $\gamma^2_j \geq \iota_j$. Thus $\vec{\gamma}^2$ satisfies the minimum layer requirements.

Now we will show that $\vec{\gamma}^2$ satisfies the capacity constraints. Consider a link $l \in L_s$. Since $\gamma^2_j \leq \gamma^1_j$, if $j \neq s$ and $\gamma^2_s = \gamma^1_s + 1$,

$$\begin{aligned} \Gamma^2_{il} &\leq \Gamma^1_{il}, \text{ if } s \notin m(i,l) \text{ and} && (1) \\ \Gamma^2_{\chi(s)l} &\leq \max(\gamma^1_s + 1, \Gamma^1_{\chi(s)l}) \\ &\leq \Gamma^1_{\chi(s)l} + 1 && (2) \end{aligned}$$

1. Let $l \in Y$. Thus $\sum_{i \in n(l)} \Gamma^2_{il} \leq \sum_{i \in n(l)} \Gamma^1_{il} + 1$. It follows that

$$b \sum_{i \in n(l)} \Gamma^2_{il} \leq b \sum_{i \in n(l)} \Gamma^1_{il} + b$$

$$\leq \quad C_l \text{ (since } l \in Y) \tag{3}$$

2. Let $l \in Z$. Since γ_j^1 is an integer for all j, so is Γ_{il}^1 for all sessions i and link l. Thus $\gamma_s^1 < \Gamma_{\chi(s)l}^1$, means that $\gamma_s^1 + 1 \leq \Gamma_{\chi(s)l}^1$. Thus $\Gamma_{\chi(s)l}^2 \leq \max(\gamma_s^1 + 1, \Gamma_{\chi(s)l}^1) = \Gamma_{\chi(s)l}^1$. Thus from (1) $\sum_{i \in n(l)} \Gamma_{il}^2 \leq \sum_{i \in n(l)} \Gamma_{il}^1$. Again it follows that

$$\begin{aligned} b \sum_{i \in n(l)} \Gamma_{il}^2 &\leq b \sum_{i \in n(l)} \Gamma_{il}^1 \\ &\leq C_l \text{ (from the feasibility of } \vec{\gamma}^1) \end{aligned} \tag{4}$$

3. Let $l \in L_s \setminus (Y \cup Z)$. Thus l satisfies pseudo-bottleneck conditions (1) and (2). Since l is not a pseudo-bottleneck link w.r.t. virtual session s, l can not satisfy pseudo-bottleneck condition (3). Thus there exists a virtual session p such that $\chi(p) \in n(l)$, $\gamma_p^1 > \gamma_s^1 + 1$ and $\gamma_p^1 > \iota_{\chi(p)l}$. Consider all virtual sessions $j \in \chi(p)$ such that $\gamma_j^1 = \Gamma_{\chi(p)l}^1$. It follows that $\gamma_j^1 \geq \gamma_p^1$. Thus $\gamma_j^1 > \iota_{\chi(j)l}$, $\gamma_j^1 > \gamma_s^1 + 1$. Hence $j \in W$ if $\gamma_j^1 = \Gamma_{\chi(p)l}^1$, $j \in \chi(p)$. Now $\gamma_j^2 = \gamma_j^1 - 1$ for all $j \in W$. It follows that $\Gamma_{\chi(p)l}^2 = \Gamma_{\chi(p)l}^1 - 1$.

$$\begin{aligned} \sum_{i \in n(l)} \Gamma_{il}^2 &= \sum_{i \in n(l) \setminus \{\chi(p), \chi(s)\}} \Gamma_{il}^2 + \Gamma_{\chi(s)l}^2 + \Gamma_{\chi(p)l}^2 \\ &\leq \sum_{i \in n(l) \setminus \{\chi(p), \chi(s)\}} \Gamma_{il}^1 + \Gamma_{\chi(s)l}^1 + 1 + \Gamma_{\chi(p)l}^1 - 1 \text{ (from (1) and (2))} \\ &= \sum_{i \in n(l)} \Gamma_{il}^1 \\ \text{Thus } b \sum_{i \in n(l)} \Gamma_{il}^2 &\leq b \sum_{i \in n(l)} \Gamma_{il}^1 \\ &\leq C_l \text{ (from the feasibility of } \vec{\gamma}^1) \end{aligned} \tag{5}$$

4. Let $l \notin L_s$.

$$\begin{aligned} b \sum_{i \in n(l)} \Gamma_{il}^2 &\leq b \sum_{i \in n(l)} \Gamma_{il}^1 \text{ (from (1) and since } l \notin L_s) \\ &\leq C_l \text{ (from feasibility of } \vec{\gamma}^1) \end{aligned} \tag{6}$$

Thus from (3), (4), (5) and (6) $\vec{\gamma}^2$ satisfies the capacity condition for feasibility. Thus $\vec{\gamma}^2$ is a feasible layer allocation vector.

Clearly $\vec{\gamma}^1 \neq \vec{\gamma}^2$. Let $\gamma_j^1 > \gamma_j^2$. Thus $j \in W$, $\gamma_j^2 = \gamma_j^1 - 1$ and $\gamma_j^1 > \gamma_s^1 + 1$. Thus $\gamma_j^1 - 1 > \gamma_s^1 = \gamma_s^2 - 1$. It follows that $\gamma_j^2 > \gamma_s^2 - 1$. Since γ_j^2 and γ_s^2 are both integers, $\gamma_j^2 \geq \gamma_s^2$. Thus if there exists j such that $\gamma_j^2 < \gamma_j^1$, then there exists s such that $\gamma_s^2 \leq \gamma_j^2$ and $\gamma_s^1 < \gamma_s^2$. Thus $\vec{\gamma}^2$ is fairer than $\vec{\gamma}^1$. Hence $\vec{\gamma}^1$ is not maximally fair.

Now let every virtual session have a pseudo-bottleneck link for feasible layer allocation vector $\vec{\gamma}^1$. Let there be a feasible layer allocation vector, $\vec{\gamma}^2$, fairer than $\vec{\gamma}^1$. Define the set τ of virtual sessions as follows. $\tau = \{j : \gamma_j^1 \neq \gamma_j^2\}$. Since $\vec{\gamma}^2$ is fairer than $\vec{\gamma}^1$, there exists a virtual session s

such that $\gamma_s^2 = \min_{j \in \tau} \gamma_j^2$ and $\gamma_s^2 > \gamma_s^1$. This follows from Proposition 1. Since γ_s^1 and γ_s^2 are both integers, $\gamma_s^2 > \gamma_s^1$ means that

$$\gamma_s^2 \geq \gamma_s^1 + 1 \tag{7}$$

Let l be the pseudo-bottleneck link w.r.t. virtual session s for layer allocation vector $\vec{\gamma}^1$. $l \in L_s$

$$\begin{aligned} \Gamma^2_{\chi(s)l} &\geq \gamma_s^2 \\ &\geq \gamma_s^1 + 1 \text{ (by (7))} \\ &= \Gamma^1_{\chi(s)l} + 1 \text{ (by pseudo-bottleneck property (2))} \end{aligned} \tag{8}$$

If $\Gamma^2_{il} \geq \Gamma^1_{il}$, $\forall i \in n(l)$, then

$$\begin{aligned} \sum_{i \in n(l)} \Gamma^2_{il} &\geq \sum_{i \in n(l)} \Gamma^1_{il} + 1 \text{ (from (8))} \\ \text{Thus } b \sum_{i \in n(l)} \Gamma^2_{il} &> C_l \text{ (from pseudo-bottleneck property (3))} \end{aligned} \tag{9}$$

(9) contradicts the feasibility of layer allocation vector $\vec{\gamma}^2$. Thus $\Gamma^2_{il} < \Gamma^1_{il}$ for some i. $i \neq \chi(s)$ since $\Gamma^2_{\chi(s)l} > \Gamma^1_{\chi(s)l}$ from (8). From feasibility of $\vec{\gamma}^2$, $\iota_{il} \leq \Gamma^2_{il}$. It follows that $\Gamma^1_{il} > \iota_{il}$. Consider virtual session $j \in m(i,l)$ such that $\gamma_j^1 = \Gamma^1_{il}$. $j \neq s$ as $i \neq \chi(s)$. $\gamma_j^1 > \iota_{\chi(j)l}$. From pseudo-bottleneck property (3), $\gamma_j^1 \leq \gamma_s^1 + 1$.

$$\begin{aligned} \gamma_j^2 &\leq \Gamma^2_{il} \text{ (since } j \in m(i,l)) \\ &< \Gamma^1_{il} \text{ (from choice of } i) \\ &= \gamma_j^1 \text{ (from choice of } j) \qquad (10) \\ &\leq \gamma_s^1 + 1 \\ &\leq \gamma_s^2 \text{ (from (7))} \qquad (11) \end{aligned}$$

From (10) $\gamma_j^2 \neq \gamma_j^1$. Thus $j \in \tau$. From (11) $\gamma_j^2 < \gamma_s^2$. This contradicts the minimality of γ_s^2 for $s \in \tau$. Thus there does not exist a feasible layer allocation vector fairer than $\vec{\gamma}^1$. Hence $\vec{\gamma}^1$ is a maximally fair layer allocation vector. □

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

5.2 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 [19] 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 $\gamma_s \in \{\lfloor \frac{r_s^c}{b} \rfloor, \lceil \frac{r_s^c}{b} \rceil\}$. Figures 5 and 6 furnish counter examples 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 [19] 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 continuous 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 at 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. Figure 6 contains a counter example. Counter examples in figures 5 and 6 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. Figure 7 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, we have developed 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 7, 7, 13 and 13 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. Vector $\vec{\omega}(k)$ denotes the rate vector at the end of the kth iteration, with components, $\omega_s(k)$.

$\Omega_{il}(k)$ 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. Vector $\vec{r}(k)$ 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. $k \rightarrow k + 1$
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.

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

   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 control 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(Lemma 10). 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 (proof of Theorem 3). 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 5.1: Consider the network of Figure 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 2 (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.*

The following theorem proves that the algorithm terminates in finite 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.*

We prove Theorem 2 below. We have proved Theorem 3 in the appendix. We first give the gist of the proof. 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 4 that any algorithm which outputs a maximally fair layer allocation, outputs a maxmin fair layer allocation, if one such exists. On account of space limitations, we present a sketch of proof here. Refer to technical report [20] for details and proofs of Lemmas 5 and 7.

Lemma 5 *The rate allocation $\vec{r}(k)$ at the end of the kth iteration is feasible, $k \geq 0$.*

Lemma 5 states that the rate allocation at the end of every iteration is feasible. This is intuitive, because every step of the algorithm respects the feasibility constraints.

Lemma 6 *If $s \notin S(k)$, $k \geq 0$, there exists a link $l \in L_s$ such that $\sum_{i \in n(l)} \lambda_{il}(k) > C_l - b$, $r_s(k) = \lambda_{\chi(s)l}(k)$ and for any virtual session j traversing the link ($\chi(j) \in n(l)$), if $r_j(k) > \mu_{\chi(j)l}$ then $r_j(k) \leq r_s(k) + b$.*

Lemma 6 states that once a virtual session is saturated, there exists a pseudo-bottleneck link for the virtual session w.r.t the rate allocations for every subsequent iteration.

Proof of Theorem 2 (Maximal Fairness Theorem): Let the algorithm terminate in k iterations. Since the rate allocation $\vec{r}(k)$ is feasible by Lemma 5, the corresponding layer allocation vector, $\vec{\gamma}$ is feasible as well. Since $S(k) = \phi$, there does not exist a virtual session s in $S(k)$. Thus from Lemma 6 for every virtual session s there exists a link which satisfies the properties of a pseudo-bottleneck link w.r.t. virtual session s for rate vector $\vec{r}(k)$ and corresponding layer allocation vector $\vec{\gamma}$. Thus $\vec{\gamma}$ is a maximally fair layer allocation vector by Lemma 4. Thus (1) follows.

If a maxmin fair layer allocation exists, then it is fairer than any other layer allocation vector, by definition of relative and maxmin fairness. Thus it is the only maximally fair layer allocation vector in the feasible set, by definition of maximal fairness. Since the output layer allocation vector, $\vec{\gamma}$ is a maximally fair under all circumstances, (2) 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 $O(|\mathcal{L}|^2 M^2)$.

The choice of an unsaturated virtual session s for possible incrementation of rate is crucial for attaining maximal fairness in the end. Figure 8 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.

6 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, [15] 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 may 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 granularity 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 [18]. In fact, [18] 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 [18] 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 [18]. 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 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.

7 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 M. Vetterli. A layered DCT coder for Internet video. *Proceedings of IEEE International Conference on Image Processing,* pages $13-16$, Lusanne, Switzerland, September, 1996.

[2] 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.

[3] 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

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

[5] 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. $325-333$

[6] T. Brown, S. Sazzad, C. Schoeder, P. Cantrell and J. Gibson. Packet Video For Heterogeneous Network Using CU-Seeme, *Proceedings of IEEE International Conference on Image Processing,* Vol. 1, September, 1996, pp. $9-12$.

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

[8] 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.

[9] M. Ghanbari. Two-Layer Coding of Video Signals for VBR Networks. *IEEE Journal on Selected Areas in Communications*, VOL. 7. No. 5. June 1989.

[10] F. Kelly, A. Maulloo, and D. Tan. "Rate control for communication networks: Shadow prices, proportional fairness and stability *Journal of Operations Research society,* 49(3):$237-252$, March 1998.

[11] S. Low and D. Lapsley. Optimization Flow Control, I: Basic Algorithm and Convergence, *IEEE/ACM Transactions on Networking,* 7(6), Dec. 1999.

[12] 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

[13] 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

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

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

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

[17] 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

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

[19] 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

[20] 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.

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

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

[23] 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.

[24] 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.

A Proof of Finite-Termination Theorem

Theorem 3 ensures that the algorithm terminates in a finite number of iterations. 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 iterations. We show that the second can take place for at most $|\mathcal{L}|M$ iterations. The result follows. The worst case termination time of this algorithm is less than the worst case completion time of the generalized algorithm of [18], which computes the maximally fair allocation of layers, in presence of nonuniform layer bandwidths. We prove it using the following lemmas.

Lemma 7 *$r_s(k+1) \geq r_s(k)$ if $k \geq 0$ for all virtual sessions s. $\lambda_{il}(k+1) \geq \lambda_{il}(k)$ if $k \geq 0$ for all sessions i and links l*

This lemma states that the rates allocated to a virtual session is a nondecreasing function of the iteration number. It follows from the fact that

$$\text{Thus } \omega_s(k+1) \geq r_s(k) \ \forall \ s \tag{12}$$

This has been proved in technical report [20].

Lemma 8 *$r_s(k)$ and $\lambda_{il}(k)$ are multiples of b for all virtual sessions s, sessions i, link l and iterations k.*

Lemma 8 is intuitive from the algorithm and has been proved in technical report [20].

Lemma 9 *If $\Lambda(k) = \phi$, $\omega_s(k) < \min_{l \in \cup_{p \in S(k-1)} L_p} \eta_l(k)$, for some $s \in S(k-1)$.*

Lemma 9 states that if no new virtual session saturates w.r.t. rate allocation $\vec{\omega}(k)$ in iteration k, then there exists an unsaturated virtual session s, for which the rate allocation $\omega_s(k)$ is strictly less than the link control parameters of all the links carrying unsaturated virtual sessions. The proof below considers the link $l_{\min}$ which has the minimum link control parameter in iteration k, amongst all links carrying unsaturated virtual sessions. Next, it considers the session $i_{\min}$ which has the minimum session link rate amongst all sessions traversing link $l_{\min}$ and are unsaturated at the end of iteration $k-1$. Any unsaturated virtual session s of $i_{\min}$ traversing $l_{\min}$ has $\omega_s(k)$ strictly less than this minimum link control parameter (link control parameter of $l_{\min}$).
Proof of Lemma 9: Let, if possible, $\Lambda(k) = \phi$ but $\omega_s(k) \geq \min_{l \in \cup_{p \in S(k-1)} L_p} \eta_l(k)$, for all $s \in S(k-1)$. Let $l_{\min} = \arg\min_{l \in \cup_{j \in S(k-1)} L_j} \eta_l(k)$. $l_{\min}$ is well defined as $S(k-1) \neq \phi$. If more than one link attain the minimum, choose any one of them as $l_{\min}$.

$$\omega_j(k) \geq \eta_{l_{\min}}(k) \ \forall j \in S(k-1) \tag{13}$$

Since $l_{\min} \in \cup_{s \in S(k-1)} L_s$, at least one virtual session in $S(k-1)$ (unsaturated virtual session) traverses through link $l_{\min}$. Thus $\Xi_{l_{\min}}(k-1) \neq \phi$. Thus $\eta_{l_{\min}}(k)$ is the maximum θ which satisfies $F_{l_{\min}}(k-1) + \sum_{i \in \Xi_{l_{\min}}(k-1)} \max(\theta, \lambda_{il_{\min}}(k-1)) \leq C_{l_{\min}}$ and observe that

$$F_{l_{\min}}(k-1) + \sum_{i \in \Xi_{l_{\min}}(k-1)} \max\left((\min_{j \in \Xi_{l_{\min}}(k-1)} \lambda_{jl_{\min}}(k-1)), \lambda_{il_{\min}}(k-1) \right) \leq F_{l_{\min}}(k-1) + \sum_{i \in \Xi_{l_{\min}}(k-1)} \max\left(\theta, \lambda_{il_{\min}}(k-1)\right) \ \forall \ \theta$$

since $\min_{i \in \Xi_{l_{\min}}(k-1)} \lambda_{il_{\min}}(k-1) \leq \lambda_{jl_{\min}}(k-1) \ \forall \ j \in \Xi_{l_{\min}}(k-1)$. Thus

$$\eta_{l_{\min}}(k) \geq \min_{i \in \Xi_{l_{\min}}(k-1)} \lambda_{il_{\min}}(k-1) \tag{14}$$

Note that for any session i and link l on the path of the session

$$\Omega_{il}(k) \leq \max(\eta_l(k), \lambda_{il}(k-1)), \ \forall \ i \in n(l) \tag{15}$$

This is because, if virtual session $j \in S(k-1)$, and virtual session j traverses through link l, then $\omega_j(k) \leq \max(\eta_l(k), \lambda_{\chi(j),l}(k-1))$. If $j \notin S(k-1)$, and virtual session j traverses through link l, then $\omega_j(k) = r_j(k-1) \leq \lambda_{\chi(j),l}(k-1) \leq \max(\eta_l(k), \lambda_{\chi(j),l}(k-1))$. Hence (15) holds.

Let $i_{\min} = \arg\min_{i \in \Xi_{l_{\min}}(k-1)} \lambda_{il_{\min}}(k-1)$. $i_{\min}$ is well defined as $\Xi_{l_{\min}}(k-1) \neq \phi$. If more than one session attain the minimum, choose any one of them as $i_{\min}$. Consider a $s \in m(i_{\min}, l_{\min}) \cap$

$S(k-1)$ ($m(i_{\min}, l_{\min}) \cap S(k-1) \neq \phi$ as $i_{\min} \in \Xi_{l_{\min}}(k-1)$.

$$\begin{aligned}
\eta_{i_{\min}l_{\min}}(k) &= \max(\eta_{l_{\min}}(k), \lambda_{i_{\min}l_{\min}}(k-1)) \\
&= \eta_{l_{\min}}(k) \text{ (from (14) and the definition of } i_{\min}). \qquad (16) \\
&\leq \eta_l(k) \ \forall\ l \in L_s, \text{ (since } s \in m(i_{\min}, l_{\min}) \cap S(k-1), \\
&\qquad \text{and from the definition of } l_{\min}) \\
&\leq \eta_{i_{\min}l}(k) \ \forall\ l \in L_s \qquad (17) \\
\text{Thus } \min_{l \in L_s} \eta_{i_{\min}l}(k) &= \eta_{i_{\min}l_{\min}}(k) \text{ (from } l_{\min} \in L_s \text{ and (17))} \\
\text{Thus } \omega_s(k) &= b\lfloor \frac{\eta_{i_{\min}l_{\min}}(k)}{b} \rfloor \text{ (since } s \in m(i_{\min}, l_{\min}) \cap S(k-1), \\
&\qquad \omega_s(k) = b\lfloor \frac{\min_{l \in L_s} \eta_{i_{\min}l}(k)}{b} \rfloor) \\
&= b\lfloor \frac{\eta_{l_{\min}}(k)}{b} \rfloor \text{ (from (16))} \\
&\leq \eta_{l_{\min}}(k) \qquad (18) \\
\text{Thus } \omega_s(k) &= \eta_{l_{\min}}(k) \text{ (from (13), (18) and since } s \in S(k-1)) \\
&= \max(\eta_{l_{\min}}(k), \lambda_{\chi(s)l_{\min}}(k-1)) \qquad (19) \\
&\qquad \text{(from (16) and since } \chi(s) = i_{\min}) \\
&\geq \Omega_{\chi(s)l_{\min}}(k) \text{ (from (15))} \\
\text{Also } \omega_s(k) &\leq \Omega_{\chi(s)l_{\min}}(k) \\
\text{Thus } \omega_s(k) &= \Omega_{\chi(s)l_{\min}}(k) \qquad (20)
\end{aligned}$$

Consider any session $i \in \Xi_{l_{\min}}(k-1)$. Let $\max(\eta_{l_{\min}}(k), \lambda_{il_{\min}}(k-1)) = \lambda_{il_{\min}}(k-1)$.

$$\begin{aligned}
\Omega_{il_{\min}}(k) &\geq \lambda_{il_{\min}}(k-1) \text{ (from (12))} \\
&= \max(\eta_{l_{\min}}(k), \lambda_{il_{\min}}(k-1)) \\
\text{Thus } \Omega_{il_{\min}}(k) &= \max(\eta_{l_{\min}}(k), \lambda_{il_{\min}}(k-1)) \text{ (from (15))} \qquad (21)
\end{aligned}$$

Now let $\max(\eta_{l_{\min}}(k), \lambda_{il_{\min}}(k-1)) = \eta_{l_{\min}}(k)$. It follows that $\omega_j(k) = \max(\eta_{l_{\min}}(k), \lambda_{il_{\min}}(k-1))$, $\forall j\ \in m(i, l_{\min}) \cap S(k-1)$ ($m(i, l_{\min}) \cap S(k-1) \neq \phi$ as $i \in \Xi_{l_{\min}}(k-1)$). The reasoning is exactly the same as that behind (19). $\Omega_{il_{\min}}(k) = \max_{j \in m(i,l_{\min})} \omega_j(k) \geq \max(\eta_{l_{\min}}(k), \lambda_{il_{\min}}(k-1))$, since $m(i, l_{\min}) \cap S(k-1) \neq \phi$. From (15),

$$\Omega_{il_{\min}}(k) = \max(\eta_{l_{\min}}(k), \lambda_{il_{\min}}(k-1)) \qquad (22)$$

From (21) and (22)

$$\Omega_{il_{\min}}(k) = \max(\eta_{l_{\min}}(k), \lambda_{il_{\min}}(k-1)) \ \forall\ i \in \Xi_{l_{\min}}(k-1) \qquad (23)$$

If $i \in n(l_{\min}) \setminus \Xi_{l_{\min}}(k-1)$, $j \notin S(k-1)$, $\forall j \in m(i, l_{\min})$ and thus $\omega_j(k) = r_j(k-1)\ \forall j \in m(i, l_{\min})$. It follows that $\Omega_{il_{\min}}(k) = \lambda_{il_{\min}}(k-1)$. Thus

$$\sum_{i \in n(l_{\min}) \setminus \Xi_{l_{\min}}(k-1)} \Omega_{il_{\min}}(k) = \sum_{i \in n(l_{\min}) \setminus \Xi_{l_{\min}}(k-1)} \lambda_{il_{\min}}(k-1)$$

$$
\begin{aligned}
&= F_{l_{\min}}(k-1) && (24)\\
\sum_{i\in n(l_{\min})} \Omega_{il_{\min}}(k) &= \sum_{i\in n(l_{\min})\setminus \Xi_{l_{\min}}(k-1)} \Omega_{il_{\min}}(k) + \sum_{i\in \Xi_{l_{\min}}(k-1)} \Omega_{il}(k)\\
&\qquad \text{(since } \Xi_{l_{\min}}(k-1) \subseteq n(l_{\min})\text{)}\\
&= F_{l_{\min}}(k-1) + \sum_{i\in \Xi_{l_{\min}}(k-1)} \max(\eta_{l_{\min}}(k), \lambda_{il_{\min}}(k-1))\\
&\qquad \text{(from (23) and (24))}\\
&= C_l \qquad (\text{since } \Xi_{l_{\min}}(k-1) \neq \phi) && (25)
\end{aligned}
$$

Thus from (20) and (25) $s \in \Lambda(k)$, but this contradicts the assumption that $\Lambda(k) = \phi$. Thus $\omega_s(k) \geq \min_{l\in\cup_{p\in S(k-1)}L_p} \eta_l(k)$, can not hold for all $s \in S(k-1)$, if $\Lambda(k) = \phi$. □

Lemma 10 *If $\Lambda(k) = \phi$, there exists a virtual session $s \in S(k-1)$ such that*

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

Lemma 10 states that if no new virtual session saturates w.r.t. rate allocation $\vec{\omega}(k)$ in iteration k, then there exists a virtual session s which satisfies the incrementation requirements of step (8) of the rate computation algorithm. This lemma is a key result required for the proof of Theorem 3.

Proof of Lemma 10: Let $\Lambda(k) = \phi$. From Lemma 9, $\omega_s(k) < \min_{l\in\cup_{p\in S(k-1)}L_p} \eta_l(k)$, for some $s \in S(k-1)$. It follows that $\omega_s(k) < \min_{l\in L_s} \eta_l(k)$.

$$
\begin{aligned}
\min_{l\in\cup_{p\in S(k-1)}L_p} \eta_l(k) &> \omega_s(k)\\
&= b\lfloor \frac{\min_{l\in L_s} \eta_{\chi(s)l}(k)}{b} \rfloor \text{ (since } s \in S(k-1)\text{)}\\
&\geq b\lfloor \frac{\min_{l\in L_s} \eta_l(k)}{b} \rfloor\\
\text{Thus } \lfloor \frac{\min_{l\in\cup_{p\in S(k-1)}L_p} \eta_l(k)}{b} \rfloor &\geq \lfloor \frac{\min_{l\in L_s} \eta_l(k)}{b} \rfloor\\
\lfloor \frac{\min_{l\in\cup_{p\in S(k-1)}L_p} \eta_l(k)}{b} \rfloor &\leq \lfloor \frac{\min_{l\in L_s} \eta_l(k)}{b} \rfloor \text{ since } s \in S(k-1), L_s \subseteq \cup_{p\in S(k-1)}L_p\\
\text{Thus } \lfloor \frac{\min_{l\in\cup_{p\in S(k-1)}L_p} \eta_l(k)}{b} \rfloor &= \lfloor \frac{\min_{l\in L_s} \eta_l(k)}{b} \rfloor && (26)
\end{aligned}
$$

Consider any $l \in L_s$, s.t. $\lfloor \frac{\eta_l(k)}{b} \rfloor = \min_{l_1\in L_s} \lfloor \frac{\eta_{l_1}(k)}{b} \rfloor$. Also $\min_{l_1\in L_s} \lfloor \frac{\eta_{l_1}(k)}{b} \rfloor = \lfloor \frac{\min_{l_1\in L_s} \eta_{l_1}(k)}{b} \rfloor$. Thus from (26)

$$
\lfloor \frac{\eta_l(k)}{b} \rfloor = \lfloor \frac{\min_{l_1\in\cup_{p\in S(k-1)}L_p} \eta_{l_1}(k)}{b} \rfloor \qquad (27)
$$

Consider any unsaturated virtual session j traversing link l ($j \in S(k-1)$, $\chi(j) \in n(l)$).

$$
\omega_j(k) = b\lfloor \frac{\min_{l_1\in L_j} \eta_{\chi(j)l_1}(k)}{b} \rfloor \text{ (since } j \in S(k-1)\text{)}
$$

$$
\begin{aligned}
&\geq b\lfloor \frac{\min_{l_1 \in L_j} \eta_{l_1}(k)}{b} \rfloor \\
&\geq b\lfloor \frac{\min_{l_1 \in \cup_{p \in S(k-1)} L_p} \eta_{l_1}(k)}{b} \rfloor \text{ since } j \in S(k-1), L_j \subseteq \cup_{p \in S(k-1)} L_p \\
&= b\lfloor \frac{\eta_l(k)}{b} \rfloor \text{ (from (27)).} \qquad (28)
\end{aligned}
$$

If session $i \in \Xi_l(k-1)$, there exists virtual session j s.t. $j \in m(i,l) \cap S(k-1)$. Thus from (28) and the fact that $\Omega_{il}(k) = \max_{j \in m(i,l)} \omega_j(k)$, it follows that $\Omega_{il}(k) \geq b\lfloor \frac{\eta_l(k)}{b} \rfloor \ \forall \ i \in \Xi_l(k-1)$. This holds for all links l for which $\lfloor \frac{\eta_l(k)}{b} \rfloor = \min_{l_1 \in L_s} \lfloor \frac{\eta_{l_1}(k)}{b} \rfloor$. The result follows. □

Lemma 11 *Let $T = \{k : r_s(k) = \omega_s(k) + b$, for some $s\}$. The cardinality of T is at most $|\mathcal{L}|M$ where $\mathcal{L}$ is the set of links and M is the number of virtual sessions.*

Lemma 11 indicates that the incrementation in step (8) of the rate computation algorithm can be executed at most $|\mathcal{L}|M$ times. This will be used to bound the completion time of the rate computation algorithm.

Proof of Lemma 11: For every iteration $k \in T$ there exists one s such that $r_s(k) = \omega_s(k) + b$. We denote such an s as $s(k)$. Note that $s(k) \in S(k-1)$. There exists a link $l \in L_{s(k)}$ such that $\omega_s(k) = b\lfloor \frac{\eta_{\chi(s(k))l}(k)}{b} \rfloor$. We call such a link, $l(k)$. If more than one link satisfies the above property, $l(k)$ is chosen amongst them arbitrarily. Elements in T define a sequence of links $l(k)$. We show that this sequence has at most $|\mathcal{L}|M$ elements.

Let $\Theta_l(k)$ denote the residual capacity of link l, at the end of the kth iteration, i.e., $\Theta_l(k) = C_l - \sum_{i \in n(l)} \lambda_{il}(k)$. We will show that $\Theta_{l(k)}(k-1) - \Theta_{l(k)}(k) \geq b$. From feasibility of $\vec{r}(k)$, $\Theta_l(k) \geq 0$, for all links l and iterations k. Thus a link l can occur in the sequence described above, at most $\lfloor \frac{\Theta_l(k)}{b} \rfloor + 1$ times, where k is the iteration in which it appears the first time in the sequence. We would next show that $\Theta_{l(k)}(k) < Mb$ for all k. Thus it follows that a link can occur at most M times in the sequence. There are $|\mathcal{L}|$ links in all. So the sequence can have at most $|\mathcal{L}|\mathcal{M}$ elements.

Now we prove that $\Theta_{l(k)}(k) - \Theta_{l(k)}(k-1) \geq b$, for all iterations $k \in T$. We first show that $\omega_{s(k)}(k) = \Omega_{\chi(s(k))l(k)}(k)$. We know that $\omega_{s(k)}(k) = b\lfloor \frac{\eta_{\chi(s(k))l(k)}(k)}{b} \rfloor$. For any virtual session $j \in m(\chi(s), l(k)) \cap S(k-1)$, $\omega_j(k) \leq b\lfloor \frac{\eta_{\chi(s(k))l(k)}(k)}{b} \rfloor$. If $j \in m(\chi(s), l(k)) \setminus S(k-1)$, $\omega_j(k) = r_j(k-1) \leq \lambda_{\chi(s(k))l(k)}(k-1) \leq \max(b\lfloor \frac{\eta_{l(k)}(k)}{b} \rfloor, \lambda_{\chi(s(k))l(k)}(k)(k-1)) = b\lfloor \frac{\eta_{\chi(s(k))l(k)}(k)}{b} \rfloor$. The last equality follows from the divisibility of $\lambda_{\chi(s(k))l(k)}(k)(k-1)$ by b (Lemma 8). Thus $\Omega_{\chi(s(k))l(k)}(k) \leq b\lfloor \frac{\eta_{\chi(s(k))l(k)}(k)}{b} \rfloor$. It follows that $\omega_{s(k)}(k) = \Omega_{\chi(s(k))l(k)}(k)$.

$$
\begin{aligned}
\lambda_{\chi(s)l(k)}(k) &\geq r_{s(k)}(k) \\
&= \omega_{s(k)}(k) + b \\
&= \Omega_{\chi(s(k))l(k)}(k) + b \\
&\geq \lambda_{\chi(s(k))l(k)}(k-1) + b \text{ (from (12))} \qquad (29) \\
\lambda_{il(k)}(k) &\geq \lambda_{il(k)}(k-1) \ \forall \ i \in n(l(k)) \text{(from Lemma 7)} \qquad (30) \\
\sum_{i \in n(l(k))} \lambda_{il(k)}(k) &\geq b + \sum_{i \in n(l(k))} \lambda_{il(k)}(k-1) \text{ (from (29) and (30))} \qquad (31)
\end{aligned}
$$

$$\begin{aligned}
\Theta_{l(k)}(k) &= C_{l(k)} - \sum_{i \in n(l(k))} \lambda_{il(k)}(k) \\
&\leq C_{l(k)} - \sum_{i \in n(l(k))} \lambda_{il(k)}(k-1) - b \text{ (from (31))} \\
&= \Theta_{l(k)}(k-1) - b \qquad (32)
\end{aligned}$$

Now we show that $\Theta_{l(k)}(k) < Mb$. We will show that $\lfloor \frac{\eta_{l(k)}(k)}{b} \rfloor = \min_{l \in L_{s(k)}} \lfloor \frac{\eta_{l(k)}(k)}{b} \rfloor$. Since $r_s(k) = \omega_s(k) + b$, it would follow that $\Omega_{il(k)}(k) \geq b \lfloor \frac{\eta_{l(k)}(k)}{b} \rfloor \; \forall i \in \Xi_{l(k)}(k-1)$. From (12), $\Omega_{il(k)}(k) \geq \lambda_{il(k)}(k-1)$. Thus

$$\begin{aligned}
\Omega_{il(k)}(k) &\geq \max(b \lfloor \frac{\eta_{l(k)}(k)}{b} \rfloor, \lambda_{il(k)}(k-1)), \; \forall i \in \Xi_{l(k)}(k-1) \qquad (33) \\
\Theta_{l(k)}(k) &= C_{l(k)} - \sum_{i \in n(l(k))} \lambda_{il(k)}(k) \\
&= C_{l(k)} - \left(F_l(k-1) + \sum_{i \in \Xi_{l(k)}(k-1)} \lambda_{il(k)}(k) \right) \\
&\leq C_{l(k)} - \left(F_l(k-1) + \sum_{i \in \Xi_{l(k)}(k-1)} \Omega_{il(k)}(k) \right) \text{ (since } \lambda_{il(k)}(k) \geq \Omega_{il(k)}(k)) \\
&\leq C_{l(k)} - \left(F_l(k-1) + \sum_{i \in \Xi_{l(k)}(k-1)} \max(b \lfloor \frac{\eta_{l(k)}(k)}{b} \rfloor, \lambda_{il(k)}(k-1)) \right) \text{ (from (33))} \\
&< C_{l(k)} - \left(F_l(k-1) + \sum_{i \in \Xi_{l(k)}(k-1)} \max(\eta_{l(k)}(k) - b, \lambda_{il(k)}(k-1)) \right) \\
&\leq |\Xi_{l(k)}(k-1)| b + C_{l(k)} - \left(F_l(k-1) + \sum_{i \in \Xi_{l(k)}(k-1)} \max(\eta_{l(k)}(k), \lambda_{il(k)}(k-1)) \right) \\
&= |\Xi_{l(k)}(k-1)| b \\
&\leq Mb \qquad (34)
\end{aligned}$$

Now we show that $\lfloor \frac{\eta_{l(k)}(k)}{b} \rfloor = \min_{l \in L_{s(k)}} \lfloor \frac{\eta_l(k)}{b} \rfloor$.

$$\begin{aligned}
\lfloor \frac{\eta_{l(k)}(k)}{b} \rfloor &\leq \lfloor \frac{\eta_{\chi(s) l(k)}(k)}{b} \rfloor \\
&= \frac{\omega_s(k)}{b} \\
&< \frac{\min_{l \in L_{s(k)}} \eta_l(k)}{b} \text{ since } r_s(k) = \omega_s(k) + b \qquad (35) \\
\text{We know that } \lfloor \frac{\eta_{l(k)}(k)}{b} \rfloor &\geq \lfloor \frac{\min_{l \in L_{s(k)}} \eta_l(k)}{b} \rfloor \text{ (since } l(k) \in L_{s(k)}) \\
\text{If } \lfloor \frac{\eta_{l(k)}(k)}{b} \rfloor &> \lfloor \frac{\min_{l \in L_{s(k)}} \eta_l(k)}{b} \rfloor
\end{aligned}$$

$$\text{then } \lfloor \frac{\eta_{l(k)}(k)}{b} \rfloor \geq \frac{\min_{l \in L_{s(k)}} \eta_l(k)}{b} \tag{36}$$

But (36) contradicts (35). Thus

$$\begin{aligned} \lfloor \frac{\eta_{l(k)}(k)}{b} \rfloor &= \lfloor \frac{\min_{l \in L_{s(k)}} \eta_l(k)}{b} \rfloor \\ &= \min_{l \in L_{s(k)}} \lfloor \frac{\eta_l(k)}{b} \rfloor \end{aligned}$$

The result follows. □

Proof of Theorem 3 (Finite Termination Theorem): We will show that for every iteration $k \geq 1$ at least one of the following holds:

1. $r_s(k) = \omega_s(k) + b$ for some virtual session s
2. $S(k) \subset S(k-1)$ (proper subset)

Clearly $S(k) \subseteq S(k-1)$ for all $k \geq 1$. So (2) can hold for at most M iterations, where $S(0) = \{1, \ldots, M\}$. Lemma 11 shows that (1) can hold for at most $|\mathcal{L}|M$ iterations. Thus the algorithm must terminate in $M + |\mathcal{L}|M$ iterations.

We show that at least one of (1) and (2) holds in every iteration k, s.t. $S(k-1) \neq \phi$. If $\Lambda(k) \neq \phi$, $S(k)$ is a proper subset of $S(k-1)$ and (2) holds. If $\Lambda(k) = \phi$, $r_s(k) = \omega_s(k) + b$ for some virtual session s from Lemma 10 and the algorithm. Thus (1) holds in this case. □

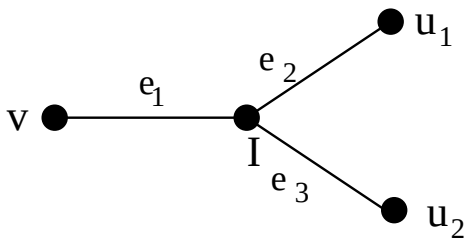

Figure 1: This figure indicates how fractional layers may be allocated in certain cases. Let u_1 and u_2 be two receivers of a session and v_1 be the source. Let the fair rates for u_1 and u_2 be 1 and 1.5 units respectively. Let each layer consume 1 unit of bandwidth. So u_1 will be allocated 1 layer and u_2, 2 layers. Intermediate node I should not replicate 50% of packets of layer 2. All packets of layer 2 are transmitted across e_2 and only 50% of layer 2 packets are transmitted across e_3. So 50% of layer 2 packets should not be replicated at node I. Thus receiver u_2 gets benefit of 1.5 layers only. Receiver I must be capable of selective replication.

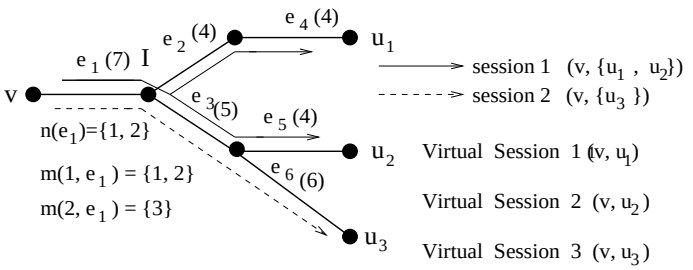

Figure 2: This figure illustrates the concept of sessions and virtual sessions, and shows some sample feasibility constraints. Session 1 includes virtual sessions, $1, 2$. Session 2 includes virtual session 3. The figure shows some sample $n(e_i)$ and $m(i, e_j)$s. The numbers in brackets, () denote 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 $0 \leq r_3 \leq 5$.

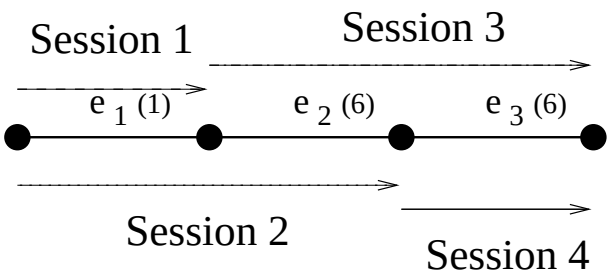

Figure 3: This figure illustrates the difference between maxmin fairness and lexicographic optimality. It shows an example network which does not have a maxmin fair layer allocation, but has a lexicographically optimal layer allocation. The sources emit unit bandwidth layers. Hence, there are only a finite number of layer allocations. 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 $(1, 0, 3, 3)$. It is possible to increase the number of layers of session 2, by hurting session 1. However session 1 has higher number of layers than session 2 in this allocation. However, $(1, 0, 3, 3)$ is lexicographically optimal.

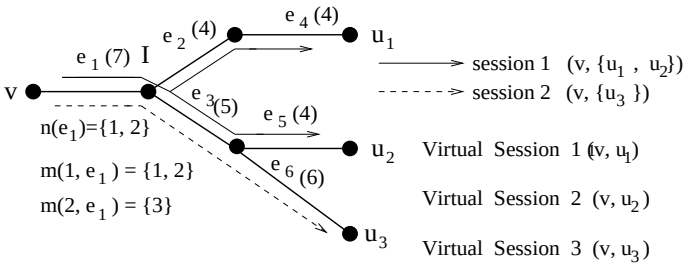

Figure 4: This figure illustrates the concept of pseudo bottleneck links. Let every layer consume 1 unit of bandwidth each ($b = 1$). Minimum layer constraints are $4, 2, 0$ for virtual sessions $1, 2$ and 3 respectively. There are no maximum rate constraints. Consider the layer allocation $(4, 2, 3)$. Link e_1 is 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.)

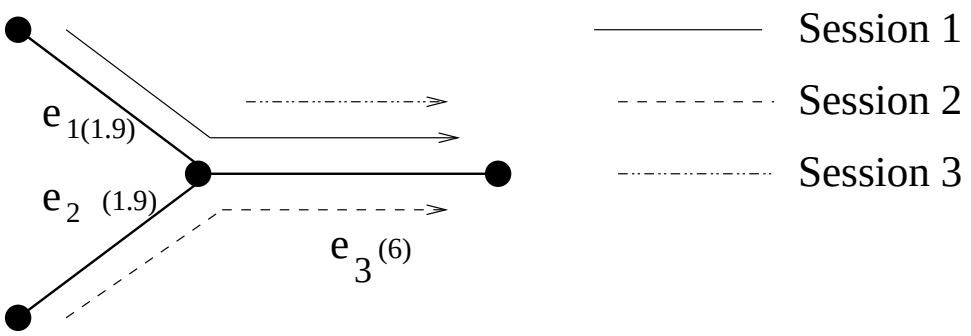

Figure 5: This figure shows that maxmin fair rate allocation in a discrete feasible set, may be different from that in the corresponding continuous feasible set (continuous feasible set obtained by removing the constraint that the layer allocations must be integers), even if the discrete feasible set contains a maxmin fair allocation. If we had allowed allocation of any rates, subject to the capacity and the minimum rate constraints(i.e., the feasible set were continuous), the maxmin fair rate allocation would be $(1.9, 1.9, 2.2)$. Now let it be possible only to allocate layers (as we have assumed in this paper). Let every layer consume 1 unit of bandwidth. Maxmin fair layer and rate allocations exist and are both equal to $(1, 1, 4)$. There is no minimum layer requirement in this case.

This figure also shows that it is not possible to obtain maximally fair allocation in the discrete feasible set, by rounding up the components of the maxmin fair allocation in the corresponding continuous feasible set Since, $(1, 1, 4)$ is maxmin fair it is also maximally fair. Consider the layer allocation scheme which allocates γ_i layers to session i, where $\gamma_i = \lceil r_i^c \rceil$ or $\gamma_i = \lfloor r_i^c \rfloor$ subject to feasibility. Since $C_{e_i} = 1.9$, for $i < 3$, $\gamma_i = 1$ $i < 3$ and either $\gamma_3 = \lfloor r_3^c \rfloor = 2$ or $\gamma_3 = \lceil r_3^c \rceil = 3$. Thus these changes lead to $(1, 1, 2)$ or $(1, 1, 3)$. Neither is maximally fair. Hence, the maximally fair layer allocation for component 3 is greater than $\lceil r_3^c \rceil$.

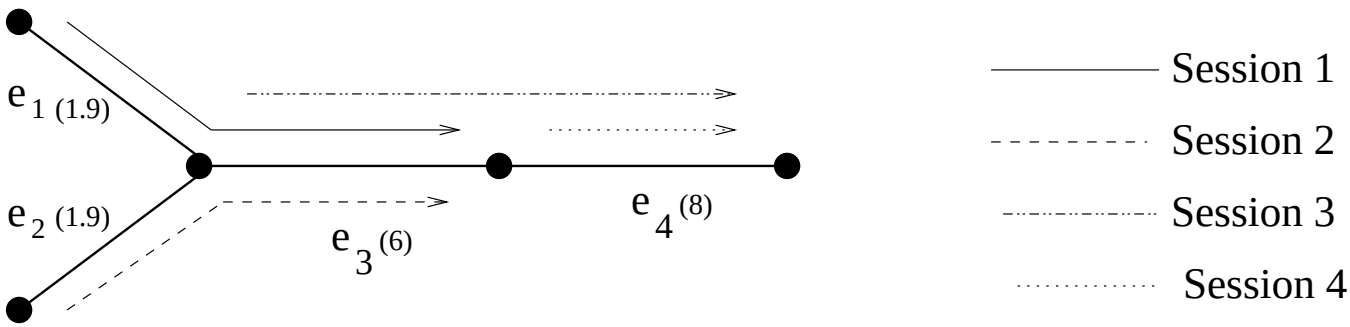

Figure 6: This figure furnishes another example illustrating that it is not possible to obtain maximally fair allocation in the discrete feasible set, by rounding up the components of the maxmin fair allocation in the corresponding continuous feasible set. The maxmin fair rate allocation, assuming that the feasible set is continuous is $(1.9, 1.9, 2.2, 5.8)$. Let every layer consume 1 unit of bandwidth. The intuitive layer allocation algorithm suggested in Section 5 would either yield a layer allocation vector $(1, 1, 3, 5)$ or $(1, 1, 2, 6)$ depending on the order in which sessions are chosen for allocation of additional layers. However $(1, 1, 4, 4)$ is a feasible layer allocation vector which is fairer than both $(1, 1, 3, 5)$ and $(1, 1, 2, 6)$. In fact maxmin fair layer allocation exists and is equal to $(1, 1, 4, 4)$ in this case. Thus neither $(1, 1, 3, 5)$ nor $(1, 1, 2, 6)$ is a maximally fair layer allocation vector. Also, the maximally fair layer allocation for component 4 is less than $\lfloor r_4^c \rfloor$.

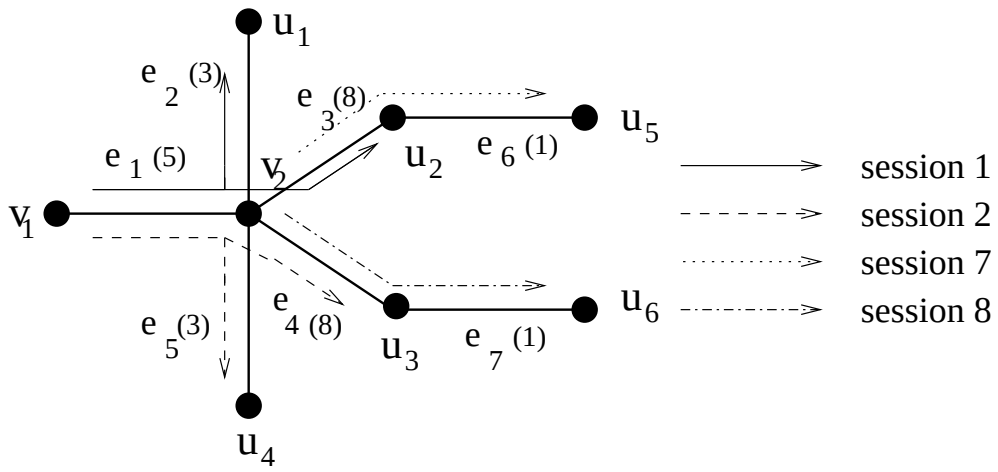

Figure 7: This figure indicates that an intuitively appealing algorithm for generating the maximally fair allocation fails. This intuitively appealing algorithm is presented in Section 5.2, page 16. Sessions $3, 4, 5, 6$ span one link each, e_3, e_4, e_6, e_7 respectively, and have not been shown in the figure. Session 1 consists of two virtual sessions, virtual sessions 1 (source: v_1, destination: u_1) and 2 (source: v_1, destination: u_2). Session 2 consists of two virtual sessions, virtual sessions 3 (source: v_1, destination: u_3) and 4 (source: v_1, destination: u_4). Sessions $3, 4, 5, 6$ consist of 1 virtual session each, virtual sessions $5, 6, 7, 8$ respectively. Sessions 7 (source: v_2, destination: u_5), and 8 (source: v_2, destination: u_6) consist of 1 virtual session each, virtual sessions $9, 10$ respectively, traversing the same edges as the respective sessions. Every layer consumes 1 unit of bandwidth ($b = 1$). Note that $C_l = b\lfloor \frac{C_l}{b} \rfloor$ for every link l. The maxmin fair rate allocation assuming the feasible set to be continuous (i.e., without the restriction that virtual session rates are integers in this case) is given by $\vec{r}$, where $r_i = 2.5$, $i \in \{1, \ldots, 4\}$, 5, $i \in \{5, 6\}$, and 0.5 otherwise. Going by this algorithm, we initially allocate γ_i layers to virtual session i, where $\gamma_i = 2i \in \{1, \ldots, 4\}$, $5i \in \{5, 6\}$ 0, otherwise. Virtual sessions $7, \ldots, 10$ 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 amongst $7, \ldots, 10$, because of feasibility. Let us select virtual sessions $9, 10$ arbitrarily amongst those assigned the minimum number of layers. 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 choose 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 which are not maximally fair. If we had incremented the number of layers of virtual sessions $7, 8$ instead of $9, 10$, in then we would have either $(3, 3, 2, 2, 5, 5, 1, 1, 0, 0)$ or $(2, 2, 3, 3, 5, 5, 1, 1, 0, 0)$, both maximally fair layer allocations, but it is not easy to know ahead the right order.

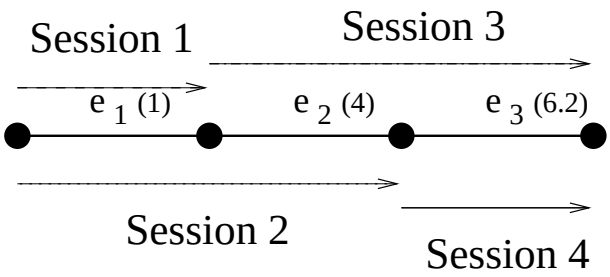

Figure 8: This figure shows that the choice of an unsaturated virtual session s for possible incrementation of rate in the maximal fairness algorithm of Section 5.2 is crucial for attaining maximal fairness in the end. If the choice does not satisfy the requirements of step (8) of this algorithm, then the final output may not be maximally fair. 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 $\Lambda(1) = \phi$. $\omega_j(1) < \min_{l \in L_j} \eta_l(1)$ for sessions $1, 2, 4$. Only sessions 1 and 2 satisfy the other criterion for incrementation though. However, if we choose to increment the rate of session 4 by 1 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 $S(1) = \{1, 2\}$. Consider 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 (chosen arbitrarily amongst 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 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 choice 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 $\Omega_{3e_3}(1) = \omega_3(1) < \lfloor \eta_{e_3}(1) \rfloor = 3$.