Fair Bandwidth Allocation for Multicasting in Networks with Discrete Feasible Set

Saswati Sarkar and Leandros Tassiulas

Abstract

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

Index Terms: Maximal fairness, Multicast, Discrete bandwidth allocation

S. Sarkar is with the department of Electrical and Systems Engineering in University of Pennsylvania, email: swati@ee.upenn.edu, Phone: 215 573 9071. Her work was supported in part by NSF grants ANI01-06984 and NCR02-38340.

L. Tassiulas is with the Computer Engineering and Telecommunications department in the University of Thessaly, Volos, Greece, and Department of Electrical and Computer Engineering and Institute for Systems Research in the University of Maryland, College Park. Email: leandros@inf.uth.gr

*Part of this paper has been presented at INFOCOM' 2000.

I. INTRODUCTION

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

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

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

Maxmin fairness [2] is a well accepted notion of fairness. A rate allocation is maxmin fair, if no receiver can be allocated a higher rate without reducing the rate of another receiver having equal or lower rate. We demonstrate later in this paper that a maxmin fair rate allocation may not exist in a discrete set. But, a maxmin fair rate allocation always exists in a continuous set [15]. Lexicographic optimality is another notion of fairness. A lexicographically optimal rate vector is one that maximizes its minimum component in a feasible set, subject to this maximization, it maximizes the second minimum, etc [*]. Lexicographically optimal rate allocation exists in a discrete set, but as we prove later, its computation is an NP-hard problem. Lexicographically optimal rate allocation is however identical to the maxmin fair rate allocation and is thus computable in polynomial complexity in a continuous set [12], [15]. We can instead compute a maximally fair rate allocation in a discrete set. A rate allocation is maximally fair if no other rate allocation is "fairer". (We describe the concept of relative fairness between two rate allocations in page 8.) That is, if a rate allocation is maximally fair, then to increase the rate of a receiver s, we must lower that of another receiver j to a value less than the new rate of s, and thus be "unfair" or "less fair" to receiver j. Maximal fairness is a weaker notion of fairness as compared to maxmin fairness and lexicographic optimality. But, maximal fairness has various desirable fairness properties, e.g., it coincides with maxmin fairness and lexicographic optimality when maxmin fair rate allocation exists. We discuss other desirable fairness properties of maximally fair allocations later. In a nutshell, maximal fair-

[*] Maxmin fairness and lexicographic optimality are defined formally in pages 10 and 11 respectively.

ness is probably the best we can achieve in the discrete case in view of the nonexistence of maxmin fair rate allocation and the computational complexity of lexicographically optimal rate allocation. We will present a polynomial complexity algorithm for computing maximally fair rate allocation in this paper. This algorithm yields a maxmin fair rate vector, if it exists. Our algorithm for computation of maximally fair allocation does not assume any property specific to internet or ATM. So it is applicable in a very general scenario. Keeping in mind ATM networks, we have incorporated minimum rate requirements and maximum rate constraints in our model.

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

This paper is organized as follows. We describe the network model in Section II. We develop notions of fairness in the discrete case in Section III. We present fairness properties that apply to any discrete feasible set in Section III. We present fairness properties specific to the layer allocation problem in Section IV-A. We present an algorithm for computation of the maximally fair rates in Section IV-B. We identify some directions for future research in the concluding section, Section V. Unless otherwise stated, the proofs can be found in the appendix.

II. Network Model

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

Every source destination pair of a session is called a virtual session. For example, if a session n has source v and destination set $\mathcal{U}$, where $\mathcal{U}= \{u_1, \ldots, u_t\}$, then n has t virtual sessions, $(v, u_1), \ldots, (v, u_t)$. Refer to figure 1 for an example of sessions and virtual sessions. To ensure fairness in a multirate network, we need to consider fair layer allocation for the virtual sessions separately, instead of considering only the layer allocations for the overall session. Every virtual session (source-destination pair) has a

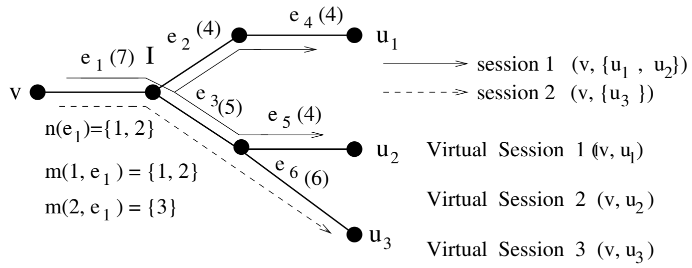

Fig. 1. This figure illustrates the concept of sessions and virtual sessions, and shows some sample feasibility constraints, $n(e_i)$ and $m(i, e_j)$s. Session 1 includes virtual sessions, $1, 2$. Session 2 includes virtual session 3. The numbers in brackets, () denote the capacities of the respective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

minimum and a maximum layer requirement[†]. We assume that every layer consumes the same amount of bandwidth, b units, independent of the session. This assumption simplifies the mathematical framework without altering the nature of the results.

A layer allocation for the virtual sessions is said to be feasible, if the number of layers for every virtual session is between the minimum and the maximum possible number of layers for the virtual session. Besides if session n corresponds to virtual sessions $m_1, \ldots, m_t$ in link l, then the bandwidth consumed by session n in link l is the maximum of the bandwidth allocated to the virtual sessions $m_1, \ldots, m_t$. The total bandwidth consumed by all sessions traversing through link l can not exceed the capacity of link l. We next define this formally. Let p_j denote the maximum number of layers of virtual session j.

Definition 1 (Feasibility Condition): A layer allocation vector $\vec{\gamma}$ is feasible if

1) γ_j is an integer for all virtual sessions j,

2) $\iota_j \leq \gamma_j \leq p_j, \forall\, j, p_j \geq \iota_j \geq 0$,

3) total bandwidth consumed by sessions traversing through link l does not exceed the capacity of

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

link l, i.e., $b\sum_{i\in n(l)}\Gamma_{il}\leq C_l$ (Capacity condition).

A rate allocation vector $\vec{r}$ is feasible if the corresponding layer allocation vector $\vec{\gamma}^r = (\gamma_1^r,\ldots,\gamma_M^r)$, defined as $\gamma_j = r_j/b$ is feasible. The feasibility conditions for rate and layer allocation vectors are equivalent, and there is a one to one correspondence between the set of the feasible rate and layer allocation vectors. Figure 1 illustrates an example network with capacity, maximum and minimum rate constraints.

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

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

Once the fair layer allocation is computed, the receivers subscribe to the appropriate multicast groups.

III. Fairness in Discrete Feasible set

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

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

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

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

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

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

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

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

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

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

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

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

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

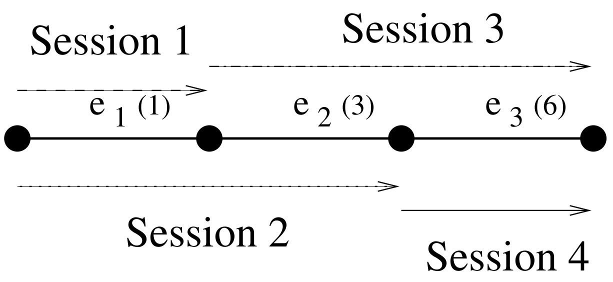

Fig. 2. This figure shows an example network which is used in the text to illustrate the difference between the three notions of fairness: maxmin fairness, lexicographic optimality and maximal fairness. The sources emit unit bandwidth layers.

$\forall i \in \{1, \ldots, M\}$. Vector $\vec{A}$ is lexicographically less than $\vec{B}$ if $\vec{B}$ is lexicographically greater than $\vec{A}$.

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

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

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

Lemma 1: If $\vec{A}$ is fairer than $\vec{B}$, then $\vec{A}$ is lexicographically greater than $\vec{B}$. The converse is not true. We prove the first part in the appendix. A counter example shows that the converse is not true. In figure 2, applying the definitions of relative fairness and lexicographical ordering, neither of the layer allocations $(1, 0, 3, 3)$ and $(0, 1, 2, 4)$ is fairer than the other, but the first is lexicographically greater than the second.

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

But as Lemma 2 stated below proves, if a maxmin fair vector exists, it is lexicographically optimal.

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

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

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

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

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

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

We prove the forward part in appendix. We show the converse using a counter example. In figure 2, both $(1, 0, 3, 3)$ and $(0, 1, 2, 4)$ are maximally fair. The second layer allocation is however not lexicographically optimal as it is lexicographically less than the first.

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

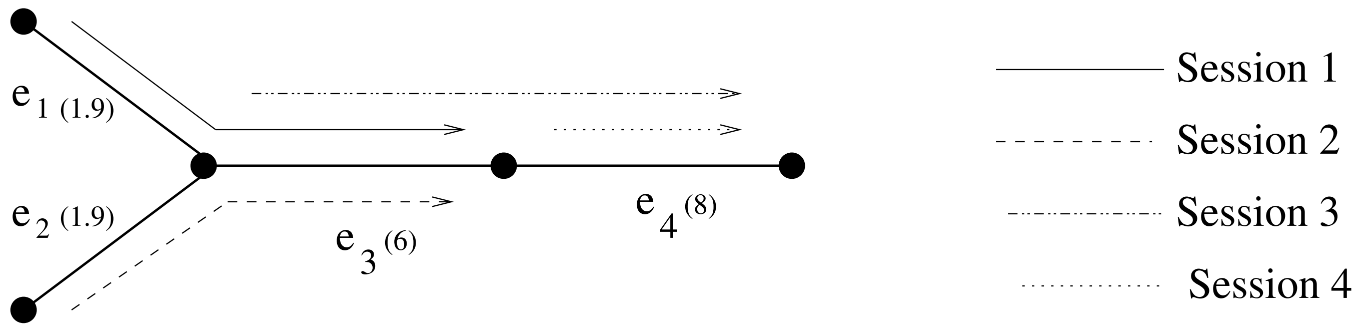

Fig. 3. This figure shows that the maxmin fair rate allocation in a discrete feasible set may be different from that in the corresponding continuous feasible set (continuous feasible set obtained by removing the constraint that the layer allocations must be integers), even if the discrete feasible set contains a maxmin fair allocation. The maxmin fair rate allocation is $(1.9, 1.9, 2.2, 5.8)$ assuming that the feasible set is continuous. Let every layer consume 1 unit of bandwidth. Now, $(1, 1, 4, 4)$ is the maxmin fair allocation in the discrete set.

This figure also furnishes an example illustrating that it is not possible to obtain a maximally fair allocation in the discrete feasible set, by rounding up the components of the maxmin fair allocation in the corresponding continuous feasible set. The intuitive layer allocation algorithm suggested in Section 4.2, page 22, would either yield a layer allocation vector $(1, 1, 3, 5)$ or $(1, 1, 2, 6)$ depending on the order in which the sessions are chosen for allocation of additional layers. But $(1, 1, 4, 4)$ is the maxmin fair allocation and hence maximally fair as well. It is also fairer than both $(1, 1, 3, 5)$ and $(1, 1, 2, 6)$, neither of which is thus maximally fair. Also, the maximally fair layer allocation for component 3 is greater than $\lceil r_3^c \rceil$ and component 4 is less than $\lfloor r_4^c \rfloor$.

Summarizing the observations made before, this network does not have a maxmin fair layer allocation, has two maximally fair layer allocations, $(1, 0, 3, 3)$ and $(0, 1, 2, 4)$, and one lexicographically optimal allocation, $(1, 0, 3, 3)$.

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

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

IV. Fair Allocation of Discrete Bandwidth Layers

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

A. Resource allocation specific fairness properties

Theorem 1—NP-hardness: Computation of lexicographically optimal layer allocation vector is NP-hard.

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

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

1) $b \sum_{i \in n(l)} \Gamma_{il} > C_l - b$ (equivalently, $\sum_{i \in n(l)} \lambda_{il} > C_l - b$),
2) $\gamma_k = \Gamma_{\chi(k)l}$ (equivalently, $r_k = \lambda_{\chi(k)l}$),

3) If $\gamma_j > \iota_{\chi(j)l}$, where $\iota_{il} = \max_{j \in m(i,l)} \iota_j$ then $\gamma_j \leq \gamma_k + 1$ (equivalently, if $r_j > \mu_{\chi(j)l}$, then $r_j \leq r_k + b$).

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

Example IV.1: Consider the network in figure 1 for an example illustrating the concept of pseudo-bottleneck links. Let every layer consume 1 unit of bandwidth each, i.e., $b = 1$. Minimum layer constraints are $4, 2, 0$ for virtual sessions $1, 2$ and 3 respectively. There are no maximum rate constraints. Consider the layer allocation $(4, 2, 3)$. Link e_1 is 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, but e_1's capacity is 7 units, which is 2 units more than the capacity utilized, and $b = 1$. Similarly, link e_6's utilized capacity is 1 unit, but its actual capacity is 6 units. This violates the pseudo-bottleneck condition (1). Link $e_2(e_3)$ is however pseudo-bottlenecked w.r.t. virtual session $1(2.)$

Lemma 5—pseudo-bottleneck lemma: A feasible layer allocation vector is maximally fair iff every virtual session has a pseudo-bottleneck link.

The pseudo-bottleneck lemma serves as a test for maximal fairness of a feasible layer allocation vector. There exists a similar result for maxmin fairness in a continuous feasible set which says that a rate allocation is maxmin fair if and only if every virtual session has a bottleneck link [15]. The definitions of bottleneck links are similar in both cases. The pseudo-bottleneck lemma strengthens our contention that maximal fairness is a good notion of fairness. Note that by this lemma, if a layer allocation vector is maximally fair, then the number of layers allocated to a virtual session s can be increased only by decreasing the number of layers of one or more virtual sessions that have at most one layer more than s. From the definition of a pseudo-bottleneck link, if there is no requirement related to the minimum number of layers for any virtual session, then the number of layers γ_s of any virtual session s will be at least $\lceil \frac{C_l}{|n(l)|b} - 1 \rceil$ for some link l in its path. Thus every virtual session is guaranteed a bandwidth close to the fair share of the capacity for at least one link on its path. In presence of minimum rate requirements, γ_s must be at least $\lceil \left(\frac{C_l - \sum_{i \in \tau(l)} \mu_{il}}{|n(l) \setminus \tau(l)| b} - 1 \right) \rceil$ layers, for some link l on its path, where $\tau(l) \subset n(l)$ is the set of sessions traversing link l with minimum session rate in link l exceeding the rate of virtual session s by more than b. Intuitively, this means that virtual session s receives an almost fair share of the residual link l bandwidth, after distributing the minimum rates to other sessions. We now deduce the expressions presented here. Let l be the pseudo-bottleneck link of virtual session s.

$$
\begin{aligned}
\lambda_{\chi(s)l} + \sum_{i \in \tau(l)} \mu_{il} + \sum_{\substack{i \neq \chi(s) \\ i \in n(l) \setminus \tau(l)}} \lambda_{il} &> C_l - b \text{ (pseudo-bottleneck conditions (1) and (3)).} \\
b\gamma_s + \sum_{i \in \tau(l)} \mu_{il} + \sum_{\substack{i \neq \chi(s) \\ i \in n(l) \setminus \tau(l)}} (b\gamma_s + b) &> C_l - b \text{ (pseudo-bottleneck conditions (2) and (3)).} \\
\gamma_s &\geq \lceil \left(\frac{C_l - \sum_{i \in \tau(l)} \mu_{il}}{|n(l) \setminus \tau(l)| b} - 1 \right) \rceil.
\end{aligned}
$$

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

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

B. Algorithm for Computation of Maximally Fair Layer Allocation

We first describe the challenges associated with designing an algorithm that would generate a maximally fair allocation of layers. Recall that the fairness complications described in Section III arise on account of the discrete nature of the feasible set. A maxmin fair allocation exists and can be computed in polynomial complexity under the same capacity and the minimum rate constraints, if the discreteness restriction is relaxed [15]. It is thus natural to ponder whether there exists a simple procedure to compute the maximally fair allocation from such a maxmin fair allocation. Interestingly, the answer is negative. Let r_s^c be the maxmin fair bandwidth allocated to virtual session s in the corresponding continuous feasible set. Consider the following intuitively appealing approach: initially allocate $\lfloor \frac{r_i^c}{b} \rfloor$ layers to virtual session i, $\forall i$. Now try to add a layer to virtual session $1, \ldots, M$, in some predetermined order, e.g., increasing order of the layers allocated, without decreasing the number of layers allocated to any virtual session at any time. Repeat this process as long as the number of layers allocated to a virtual session can be increased without decreasing the layers allocated to some other virtual session. A counter example provided in figure 3 demonstrates that this algorithm need not yield a maximally fair layer allocation. In fact, examples in figure 3 show that there may exist virtual sessions i, j such that $\gamma_i < \lfloor \frac{r_i^c}{b} \rfloor$ and similarly $\gamma_j > \lceil \frac{r_i^c}{b} \rceil$ in all maximally fair layer allocations. If, however, the maxmin fair rate allocations are computed with capacity C_l of link l replaced by $b\lfloor \frac{C_l}{b} \rfloor$, then the algorithm mentioned above may yield a maximally fair layer allocation if one tries to add a layer to the virtual sessions in some particular order. But there is no obvious way to determine this order, particularly for multicast networks. Figure 4 shows

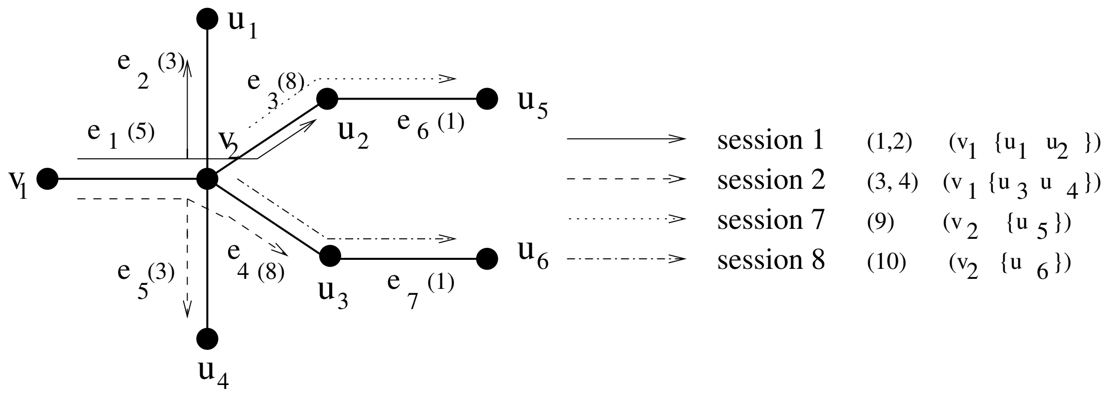

Fig. 4. This figure indicates that the intuitively appealing algorithm presented in Section 4.2, for generating the maximally fair allocation fails. The numbers next to the sessions refer to the virtual sessions belonging to the sessions, e.g., session 1 consists of virtual sessions $1, 2$ with receivers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

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

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

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

The algorithm follows.

1) $k = 0$, $\eta_l(0) = 0$, $F_l(0) = 0$, $U_l(0) = n(l)$ $\forall$ link l, $S(0) = \{1, \ldots, M\}$, $r_j(0) = \mu_j$, $\forall j \in S(0)$, $\lambda_{il}(0) = \max_{j \in m(i,l)} r_j(0)$.
2) $k \rightarrow k + 1$
3) For every link l in the network compute the link control parameter:

If $U_l(k-1) \neq \phi$, then

$$\eta_l(k) = \max_{\theta} \left(F_l(k-1) + \sum_{i \in U_l(k-1)} \max(\theta, \lambda_{il}(k-1)) = C_l \right), \tag{1}$$

else $\eta_l(k) = \eta_l(k-1)$.

For all unsaturated virtual sessions s passing through link l, $(s \in S(k-1) \cap (\cup_{i \in \{1,\ldots N\}} m(i,l)))$, session link parameter, $\eta_{\chi(s)l}(k) = \max(\eta_l(k), \lambda_{\chi(s),l}(k-1))$

4) If virtual session $s \in S(k-1)$, $\omega_s(k) = b\lfloor \frac{\min_{l \in L_s} \eta_{\chi(s)l}(k)}{b} \rfloor$,

   else $\omega_s(k) = r_s(k-1)$.

5) For every link l in the network, and every session i in $n(l)$, $\Omega_{il}(k) = \max_{s \in m(i,l)} \omega_s(k)$.

6) Compute the set of virtual sessions saturated during the kth iteration under rate vector $\vec{\omega}(k)$: $\Lambda(k) = \{s : s \in S(k-1), \exists\, l \in L_s, \omega_s(k) = \Omega_{\chi(s)l}(k), \sum_{i \in n(l)} \Omega_{il}(k) > C_l - b\}$.

7) If $\Lambda(k) \neq \phi$, $r_s(k) = \omega_s(k)$, $\forall s$ and go to step (9).

8) If possible, find a virtual session $s \in S(k-1)$, s.t.

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

   If no such s is found in $S(k-1)$, again $r_j(k) = \omega_j(k)$ for all virtual sessions, j, otherwise compute $r_j(k)$ for all virtual sessions j as

$$r_j(k) = \begin{cases} \omega_j(k) & j \neq s \\ \omega_j(k) + b & \text{otherwise.} \end{cases}$$

9) For every link l and every session in $n(l)$, session link rate $\lambda_{il}(k) = \max_{s \in m(i,l)} r_s(k)$.

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

We explain the algorithm here.

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

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

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

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

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

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

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

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

An example illustrating the operation of the algorithm follows.

Example IV.1: Consider the network in example IV-A, pp. 17. $L_1 = \{e_1, e_2, e_4\}$, $L_2 = \{e_1, e_3, e_5\}$, $L_3 = \{e_1, e_3, e_6\}$. Link control parameters can be computed as, $\eta_{e_1}(1) = 3$, $\eta_{e_2}(1) = 4$, $\eta_{e_3}(1) = 2.5$,

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

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

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

Theorem 2—Maximal-Fairness Theorem: The output layer allocation vector $\vec{\gamma}$ is

1) maximally fair, and

2) maxmin fair, if a maxmin fair layer allocation exists.

We prove theorem 2 using the following lemmas.

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

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

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

At the end of every iteration, either the rate of at least one virtual session increases by b units or the number of unsaturated virtual sessions decreases by at least one (proof of Theorem 3, page 45). Neither of these two can continue indefinitely. Hence, the algorithm terminates in a finite number of iterations. Theorem 3 establishes a stronger result, that the algorithm terminates in a polynomial number of iterations.

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

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

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

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

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

V. Conclusion and Discussion

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

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

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

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

VI. Acknowledgement

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

References

[1] 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

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

[3] J. Byers, M. Luby, M. Mitzenmatcher, "Fine-Grained Leyered Multicast," *Proceedings of IEEE INFOCOM,* 2001, pp. $1143 - 1151$, Anchorage, Alaska, 2001

[4] 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.

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

[6] A. Legout, and E. W. Biersack. PLM: Fast Convergence for Cumulative Layered Multicast Transmission Schemes. *Proceedings of ACM SIGMETRICS'2000,* pp. $13 - 22$, Santa Clara, CA, USA,

[7] A. Legout, J. Nonnenmacher, and E. W. Biersack, Bandwidth Allocation Policies for Unicast and Multicast Flows, *IEEE/ACM Transactions on Networking,* 9(4), pp. $464 - 478$, August 2001.

[8] 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, pp. $1062 - 1072$, San Francisco, CA, 1998

[9] X. Li, S. Paul and M. Ammar. Multi-session Rate Control for Layered Video Multicast *Technical Report* GT-CC-98 $- 21$, College of Computing, Georgia Institute of Technology, '98

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

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

[12] D. Rubenstein, J. Kurose and D. Towsley. The Impact of Multicast Layering on Network Fairness *IEEE Transactions on Networking,* Vol. 10, No. 2, pp. $169-182$, April 2002

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

[14] S. Sarkar and L. Tassiulas: Fair Allocation of Utilities in Multirate Multicast Networks *Proceedings of the* 37*th Annual Allerton Conference on Communication, Control and Computing,*1999, pp. $718-728$, October 1999

[15] S. Sarkar and L. Tassiulas: Distributed Algorithms for Computation of Fair Rates in Multirate Multicast Trees *Proceedings of IEEE INFOCOM'* 2000, pp. $52-62$, Tel Aviv, Israel, March' 2000

[16] L. Vicisano, L. Rizzo, J. Crowcroft, TCP-like congestion control for layered multicast data transfer, *Proceedings of INFOCOM, 98*, San Francisco, CA, 98, pp. $996-1003$

Appendix

We first present Proposition 2 which we would require in the proofs.

Proposition 2: If $\vec{A}$ is fairer than $\vec{B}$, then $\vec{B}$ is not fairer than $\vec{A}$.

Proposition 2 has been proved in [13].

Proof of Lemma 1: For lexicographical comparison, we can ignore the components of $\vec{A}$ and $\vec{B}$ which are componentwise equal. Without loss of generality, $A_j \neq B_j$ 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 first part of the lemma follows. We have already shown that the converse is not true. ∎

We use Lemma 1 in the proof of Lemmas 2 and 4.

Proof of Lemma 2: If $\vec{A}$ is maxmin fair, then $\vec{A}$ is fairer than any other feasible vector. Lexicographical optimality of $\vec{A}$ follows from Lemma 1. ∎

Proof of Lemma 3: Proposition 2 and the definitions of maxmin fairness and maximal fairness show that a maxmin fair vector is maximally fair as well. Let there exist a maxmin fair vector, $\vec{A}$. Consider another feasible vector $\vec{B}$. From definition of maxmin fairness, $\vec{A}$ is fairer than $\vec{B}$ and thus $\vec{B}$ can not be maximally fair. Thus $\vec{A}$ is the only maximally fair vector in this case. ∎

Proof of Lemma 4: We prove the forward part here. We have already shown that the converse does not hold using a counter example.

Let $\vec{A}$ be lexicographically optimal in a feasible set, but not maximally fair. Thus there exists another feasible vector $\vec{B}$, fairer than $\vec{A}$. From Lemma 1, $\vec{B}$ is lexicographically greater than $\vec{A}$. This contradicts the lexicographic optimality of $\vec{A}$. ∎

We show the NP-hardness via a reduction of the maximum independent set problem to the current problem. The former is a well known instance of an NP-complete problem [5]. 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 $\{i_1, \ldots, i_p, k_1, \ldots, k$ 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 $v_{i_1j1}, v_{i_1j2}, v_{i_2j1}, v_{i_2j2}, \ldots, v_{i_pj1}, v_{i_pj2}, v_{jk_11}, v_{jk_12}, \ldots, v_{jk_t1}, v_{jk_t2}$ 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:

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

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 the components of any feasible layer allocation vector can be either 0 or 1, 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 $P = NP$. ■ 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 5 (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:

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

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_j^1 is an integer for all j and hence γ_j^2 is an integer for all j.

$\gamma_j^2 \geq \gamma_j^1$, if $j \notin W$. Since $\gamma_j^1 \geq \iota_j$ from feasibility of $\vec{\gamma}^1$, $\gamma_j^2 \geq \iota_j$, if $j \notin W$. Now let $j \in W$. $\gamma_j^1 > \iota_{\chi(j)l}$ for some $l \in L_j$. Thus $\gamma_j^1 > \iota_{\chi(j)l} \geq \iota_j$. Since γ_j^1 and ι_j are integers, $\gamma_j^1 \geq \iota_j + 1$. It follows that $\gamma_j^2 \geq \iota_j$. 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 $\gamma_j^2 \leq \gamma_j^1$, if $j \neq s$ and $\gamma_s^2 = \gamma_s^1 + 1$,

$$\begin{aligned} \Gamma_{il}^2 &\leq \Gamma_{il}^1, \text{ if } s \notin m(i,l) \text{ and} && (2) \\ \Gamma_{\chi(s)l}^2 &\leq \max(\gamma_s^1 + 1, \Gamma_{\chi(s)l}^1) \\ &\leq \Gamma_{\chi(s)l}^1 + 1 && (3) \end{aligned}$$

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

$$\begin{aligned} b \sum_{i \in n(l)} \Gamma_{il}^2 &\leq b \sum_{i \in n(l)} \Gamma_{il}^1 + b \\ &\leq C_l \text{ (since } l \in Y) && (4) \end{aligned}$$

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 (2) $\sum_{i \in n(l)} \Gamma_{il}^2 \leq \sum_{i \in n(l)} \Gamma_{il}^1$. Again it follows that

$$\begin{aligned} b \sum_{i \in n(l)} \Gamma_{il}^2 &\leq b \sum_{i \in n(l)} \Gamma_{il}^1 \\ &\leq C_l \text{ (from the feasibility of } \vec{\gamma}^1) && (5) \end{aligned}$$

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 $\Gamma_{\chi(p)l}^2 = \Gamma_{\chi(p)l}^1 - 1$.

$$\begin{aligned}
\sum_{i \in n(l)} \Gamma_{il}^2 &= \sum_{i \in n(l) \setminus \{\chi(p), \chi(s)\}} \Gamma_{il}^2 + \Gamma_{\chi(s)l}^2 + \Gamma_{\chi(p)l}^2 \\
&\leq \sum_{i \in n(l) \setminus \{\chi(p), \chi(s)\}} \Gamma_{il}^1 + \Gamma_{\chi(s)l}^1 + 1 + \Gamma_{\chi(p)l}^1 - 1 \text{ (from (2) and (3))} \\
&= \sum_{i \in n(l)} \Gamma_{il}^1 \\
\text{Thus } b \sum_{i \in n(l)} \Gamma_{il}^2 &\leq b \sum_{i \in n(l)} \Gamma_{il}^1 \\
&\leq C_l \text{ (from the feasibility of } \vec{\gamma}^1) \qquad (6)
\end{aligned}$$

4) Let $l \notin L_s$.

$$\begin{aligned}
b \sum_{i \in n(l)} \Gamma_{il}^2 &\leq b \sum_{i \in n(l)} \Gamma_{il}^1 \text{ (from (2) and since } l \notin L_s) \\
&\leq C_l \text{ (from feasibility of } \vec{\gamma}^1) \qquad (7)
\end{aligned}$$

Thus from (4), (5), (6) and (7) $\vec{\gamma}^2$ 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

$$\gamma_s^2 \geq \gamma_s^1 + 1 \tag{8}$$

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

$$\begin{aligned} \Gamma^2_{\chi(s)l} &\geq \gamma_s^2 \\ &\geq \gamma_s^1 + 1 \text{ (by (8))} \\ &= \Gamma^1_{\chi(s)l} + 1 \text{ (by pseudo-bottleneck property (2))} \end{aligned} \tag{9}$$

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

$$\begin{aligned} \sum_{i \in n(l)} \Gamma^2_{il} &\geq \sum_{i \in n(l)} \Gamma^1_{il} + 1 \text{ (from (9))} \\ \text{Thus } b \sum_{i \in n(l)} \Gamma^2_{il} &> C_l \text{ (from pseudo-bottleneck property (1))} \end{aligned} \tag{10}$$

(10) 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 (9). 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), $\gamma_j^1 \leq \gamma_s^1 + 1$.

$$\begin{aligned} \gamma_j^2 &\leq \Gamma^2_{il} \text{ (since } j \in m(i,l)) \\ &< \Gamma^1_{il} \text{ (from choice of } i) \\ &= \gamma_j^1 \text{ (from choice of } j) \qquad (11) \\ &\leq \gamma_s^1 + 1 \\ &\leq \gamma_s^2 \text{ (from (8))} \qquad (12) \end{aligned}$$

From (11) $\gamma_j^2 \neq \gamma_j^1$. Thus $j \in \tau$. From (12) $\gamma_j^2 < \gamma_s^2$. 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 prove Theorem 3 using the following key steps. 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.

Lemma 8: $r_s(k+1) \geq r_s(k)$ if $k \geq 0$ for all virtual sessions s. Also, $\lambda_{il}(k+1) \geq \lambda_{il}(k)$ 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.

Proof of Lemma 8: If $s \notin S(k)$, $\omega_s(k+1) = r_s(k)$. Let $s \in S(k)$. Consider any link $l \in L_s$.

$$\begin{aligned}
\eta_{l\chi(s)}(k+1) &= \max(\eta_l(k+1), \lambda_{\chi(s),l}(k)) \\
&\geq r_s(k) \text{ (since } s \in m(\chi(s),l)\ \forall\, l \in L_s \text{ and } \lambda_{il}(k) = \max_{s \in m(i,l)} r_s(k)) \qquad (13) \\
\omega_s(k+1) &= b\lfloor \frac{\min_{l \in L_s} \eta_{l\chi(s)}(k+1)}{b} \rfloor \\
&\geq b\lfloor \frac{r_s(k)}{b} \rfloor \text{ (from (13))} \\
&= r_s(k) \text{ (by Lemma 9)}
\end{aligned}$$

$$\text{Thus } \omega_s(k+1) \geq r_s(k)\ \forall\ s \qquad (14)$$

Thus $r_s(k+1) \geq \omega_s(k+1) \geq r_s(k)$.

Since $\lambda_{il}(k+1) = \max_{j \in m(i,l)} r_j(k+1)$ and $r_j(k+1) \geq r_j(k)\ \forall j$, $\lambda_{il}(k+1) \geq \lambda_{il}(k)$. ∎

Lemma 9: Rates $r_s(k)$ and $\lambda_{il}(k)$ are multiples of b for all virtual sessions s, sessions i, link l and iterations k.

Proof of Lemma 9: $r_s(0) = \mu_s$, and μ_s is a multiple of b by definition. Let $r_s(k)$ be a multiple of b. $\omega_s(k+1) = r_s(k)$, if $s \notin S(k)$. $\omega_s(k+1) = b\lfloor \frac{\min_{l \in L_s} \eta_{ls}(k+1)}{b} \rfloor$, if $s \in S(k)$. Thus $\omega_s(k+1)$ is a multiple of b in both cases. $r_s(k+1) = \omega_s(k+1)$ or $r_s(k+1) = \omega_s(k+1) + b$. Thus $r_s(k+1)$ is a multiple of

b. The first part follows by induction.

Since $\lambda_{il}(k) = \max_{j \in m(i,l)} r_j(k)$ and $r_j(k)$ is a multiple of $b\ \forall j$, so is $\lambda_{il}(k)$. ∎

Lemma 10: 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 10 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. Consider the link $l_{\min}$ which has the minimum link control parameter in iteration k, among all links carrying unsaturated virtual sessions. Next, consider the session $i_{\min}$ which has the minimum session link rate among all sessions traversing link $l_{\min}$ and are unsaturated at the end of iteration $k-1$. We show that any unsaturated virtual session s of $i_{\min}$ traversing $l_{\min}$ has $\omega_s(k)$ strictly less than this minimum link control parameter.

Proof of Lemma 10: 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 $l_{\min}$.

$$\omega_j(k) \geq \eta_{l_{\min}}(k)\ \forall j \in S(k-1) \tag{15}$$

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 $U_{l_{\min}}(k-1) \neq \phi$. Thus $\eta_{l_{\min}}(k)$ is the maximum θ which satisfies $F_{l_{\min}}(k-1) + \sum_{i \in U_{l_{\min}}(k-1)} \max(\theta, \lambda_{il_{\min}}(k-1)) \leq C_{l_{\min}}$ and observe that

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

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

$$\eta_{l_{\min}}(k) \geq \min_{i \in U_{l_{\min}}(k-1)} \lambda_{il_{\min}}(k-1) \tag{16}$$

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

$$\Omega_{il}(k) \leq \max(\eta_l(k), \lambda_{il}(k-1)),\ \forall\, i \in n(l) \tag{17}$$

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 (17) holds.

Let $i_{\min} = \arg\min_{i \in U_{l_{\min}}(k-1)} \lambda_{il_{\min}}(k-1)$. $i_{\min}$ is well defined as $U_{l_{\min}}(k-1) \neq \phi$. 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 $i_{\min} \in U_{l_{\min}}(k-1)$.

$$
\begin{aligned}
\eta_{i_{\min}l_{\min}}(k) &= \max(\eta_{l_{\min}}(k), \lambda_{i_{\min}l_{\min}}(k-1)) \\
&= \eta_{l_{\min}}(k) \text{ (from (16) and the definition of } i_{\min}). \qquad (18)\\
&\leq \eta_l(k)\ \forall\ l \in L_s,\ \text{(since } s \in m(i_{\min}, l_{\min}) \cap S(k-1), \\
&\qquad \text{and from the definition of } l_{\min}) \\
&\leq \eta_{i_{\min}l}(k)\ \forall\ l \in L_s \qquad (19)\\
\text{Thus } \min_{l \in L_s} \eta_{i_{\min}l}(k) &= \eta_{i_{\min}l_{\min}}(k) \text{ (from } l_{\min} \in L_s \text{ and (19))} \\
\text{Thus } \omega_s(k) &= b\lfloor \frac{\eta_{i_{\min}l_{\min}}(k)}{b} \rfloor \text{ (since } s \in m(i_{\min}, l_{\min}) \cap S(k-1), \\
&\qquad \omega_s(k) = b\lfloor \frac{\min_{l \in L_s} \eta_{i_{\min}l}(k)}{b} \rfloor) \\
&= b\lfloor \frac{\eta_{l_{\min}}(k)}{b} \rfloor \text{ (from (18))} \\
&\leq \eta_{l_{\min}}(k) \qquad (20)\\
\text{Thus } \omega_s(k) &= \eta_{l_{\min}}(k) \text{ (from (15), (20) and since } s \in S(k-1)) \\
&= \max(\eta_{l_{\min}}(k), \lambda_{\chi(s)l_{\min}}(k-1)) \qquad (21)\\
&\qquad \text{(from (18) and since } \chi(s) = i_{\min}) \\
&\geq \Omega_{\chi(s)l_{\min}}(k) \text{ (from (17))} \\
\text{Also } \omega_s(k) &\leq \Omega_{\chi(s)l_{\min}}(k) \\
\text{Thus } \omega_s(k) &= \Omega_{\chi(s)l_{\min}}(k) \qquad (22)
\end{aligned}
$$

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

$$\begin{aligned} \Omega_{il_{\min}}(k) &\geq \lambda_{il_{\min}}(k-1) \text{ (from (14))} \\ &= \max(\eta_{l_{\min}}(k), \lambda_{il_{\min}}(k-1)) \\ \text{Thus } \Omega_{il_{\min}}(k) &= \max(\eta_{l_{\min}}(k), \lambda_{il_{\min}}(k-1)) \text{ (from (17))} \end{aligned} \tag{23}$$

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 $i \in U_{l_{\min}}(k-1)$). The reasoning is exactly the same as that behind (21). $\Omega_{il_{\min}}(k) = \max_{j \in m(i,l_{\min})} \omega_j(k) \geq \max(\eta_{l_{\min}}(k), \lambda_{il_{\min}}(k-1))$, since $m(i, l_{\min}) \cap S(k-1) \neq \phi$. From (17),

$$\Omega_{il_{\min}}(k) = \max(\eta_{l_{\min}}(k), \lambda_{il_{\min}}(k-1)) \tag{24}$$

From (23) and (24)

$$\Omega_{il_{\min}}(k) = \max(\eta_{l_{\min}}(k), \lambda_{il_{\min}}(k-1)) \; \forall \; i \in U_{l_{\min}}(k-1) \tag{25}$$

If $i \in n(l_{\min}) \setminus U_{l_{\min}}(k-1)$, $j \notin S(k-1)$, $\forall j \in m(i, l_{\min})$ 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

$$\begin{aligned} \sum_{i \in n(l_{\min}) \setminus U_{l_{\min}}(k-1)} \Omega_{il_{\min}}(k) &= \sum_{i \in n(l_{\min}) \setminus U_{l_{\min}}(k-1)} \lambda_{il_{\min}}(k-1) \\ &= F_{l_{\min}}(k-1) \end{aligned} \tag{26}$$

$$\begin{aligned} \sum_{i \in n(l_{\min})} \Omega_{il_{\min}}(k) &= \sum_{i \in n(l_{\min}) \setminus U_{l_{\min}}(k-1)} \Omega_{il_{\min}}(k) + \sum_{i \in U_{l_{\min}}(k-1)} \Omega_{il}(k) \\ &\qquad \text{(since } U_{l_{\min}}(k-1) \subseteq n(l_{\min})\text{)} \\ &= F_{l_{\min}}(k-1) + \sum_{i \in U_{l_{\min}}(k-1)} \max(\eta_{l_{\min}}(k), \lambda_{il_{\min}}(k-1)) \\ &\qquad \text{(from (25) and (26))} \\ &= C_l \quad \text{(since } U_{l_{\min}}(k-1) \neq \phi\text{)} \end{aligned} \tag{27}$$

Thus from (22) and (27) $s \in \Lambda(k)$, 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 11: If $\Lambda(k) = \phi$, there exists a virtual session $s \in S(k-1)$ such that

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

Lemma 11 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 11: Let $\Lambda(k) = \phi$. From Lemma 10, $\omega_s(k) < \min_{l \in \cup_{p \in S(k-1)} L_p} \eta_l(k)$, for some $s \in S(k-1)$. It follows that $\omega_s(k) < \min_{l \in L_s} \eta_l(k)$.

$$\begin{aligned} \min_{l \in \cup_{p \in S(k-1)} L_p} \eta_l(k) &> \omega_s(k) \\ &= b\lfloor \frac{\min_{l \in L_s} \eta_{\chi(s)l}(k)}{b} \rfloor \text{ (since } s \in S(k-1)) \\ &\geq b\lfloor \frac{\min_{l \in L_s} \eta_l(k)}{b} \rfloor \\ \text{Thus } \lfloor \frac{\min_{l \in \cup_{p \in S(k-1)} L_p} \eta_l(k)}{b} \rfloor &\geq \lfloor \frac{\min_{l \in L_s} \eta_l(k)}{b} \rfloor \\ \lfloor \frac{\min_{l \in \cup_{p \in S(k-1)} L_p} \eta_l(k)}{b} \rfloor &\leq \lfloor \frac{\min_{l \in L_s} \eta_l(k)}{b} \rfloor \text{ since } s \in S(k-1), L_s \subseteq \cup_{p \in S(k-1)} L_p \\ \text{Thus } \lfloor \frac{\min_{l \in \cup_{p \in S(k-1)} L_p} \eta_l(k)}{b} \rfloor &= \lfloor \frac{\min_{l \in L_s} \eta_l(k)}{b} \rfloor \end{aligned} \tag{28}$$

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 (28)

$$\lfloor \frac{\eta_l(k)}{b} \rfloor = \lfloor \frac{\min_{l_1 \in \cup_{p \in S(k-1)} L_p} \eta_{l_1}(k)}{b} \rfloor \tag{29}$$

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

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

$$\geq \quad b\lfloor\frac{\min_{l_1\in\cup_{p\in S(k-1)}L_p}\eta_{l_1}(k)}{b}\rfloor \text{ since } j\in S(k-1), L_j\subseteq\cup_{p\in S(k-1)}L_p$$
$$= \quad b\lfloor\frac{\eta_l(k)}{b}\rfloor \text{ (from (29))}. \tag{30}$$

If session $i \in U_l(k-1)$, there exists virtual session j s.t. $j \in m(i,l) \cap S(k-1)$. Thus from (30) and the fact that $\Omega_{il}(k) = \max_{j\in m(i,l)} \omega_j(k)$, it follows that $\Omega_{il}(k) \geq b\lfloor\frac{\eta_l(k)}{b}\rfloor \ \forall\, i \in U_l(k-1)$. 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 12: Let $T = \{k : r_s(k) = \omega_s(k) + b, \text{ for some } s\}$. 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 12 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 12: 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 among 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 9). Thus $\Omega_{\chi(s(k))l(k)}(k) \leq b\lfloor \frac{\eta_{\chi(s(k))l(k)}(k)}{b} \rfloor$. It follows that $\omega_{s(k)}(k) = \Omega_{\chi(s(k))l(k)}(k)$.

$$
\begin{aligned}
\lambda_{\chi(s)l(k)}(k) &\geq r_{s(k)}(k) \\
&= \omega_{s(k)}(k) + b \\
&= \Omega_{\chi(s(k))l(k)}(k) + b \\
&\geq \lambda_{\chi(s(k))l(k)}(k-1) + b \text{ (from (14))} \qquad (31) \\
\lambda_{il(k)}(k) &\geq \lambda_{il(k)}(k-1)\ \forall\, i \in n(l(k)) \text{(from Lemma 8)} \qquad (32) \\
\sum_{i\in n(l(k))} \lambda_{il(k)}(k) &\geq b + \sum_{i\in n(l(k))} \lambda_{il(k)}(k-1) \text{ (from (31) and (32))} \qquad (33) \\
\Theta_{l(k)}(k) &= C_{l(k)} - \sum_{i\in n(l(k))} \lambda_{il(k)}(k) \\
&\leq C_{l(k)} - \sum_{i\in n(l(k))} \lambda_{il(k)}(k-1) - b \text{ (from (33))} \\
&= \Theta_{l(k)}(k-1) - b \qquad (34)
\end{aligned}
$$

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 $\Omega_{il(k)}(k) \geq b\lfloor \frac{\eta_{l(k)}(k)}{b} \rfloor\ \forall i \in U_{l(k)}(k-1)$. From (14), $\Omega_{il(k)}(k) \geq \lambda_{il(k)}(k-1)$. Thus

$$
\begin{aligned}
\Omega_{il(k)}(k) &\geq \max(b\lfloor \frac{\eta_{l(k)}(k)}{b} \rfloor, \lambda_{il(k)}(k-1)),\ \forall i \in U_{l(k)}(k-1) \qquad (35) \\
\Theta_{l(k)}(k) &= C_{l(k)} - \sum_{i\in n(l(k))} \lambda_{il(k)}(k) \\
&= C_{l(k)} - \left(F_l(k-1) + \sum_{i\in U_{l(k)}(k-1)} \lambda_{il(k)}(k) \right) \\
&\leq C_{l(k)} - \left(F_l(k-1) + \sum_{i\in U_{l(k)}(k-1)} \Omega_{il(k)}(k) \right) \text{ (since } \lambda_{il(k)}(k) \geq \Omega_{il(k)}(k)) \\
&\leq C_{l(k)} - \left(F_l(k-1) + \sum_{i\in U_{l(k)}(k-1)} \max(b\lfloor \frac{\eta_{l(k)}(k)}{b} \rfloor, \lambda_{il(k)}(k-1)) \right) \text{ (from (35))}
\end{aligned}
$$

$$
\begin{aligned}
&< C_{l(k)} - \left(F_l(k-1) + \sum_{i \in U_{l(k)}(k-1)} \max(\eta_{l(k)}(k) - b, \lambda_{il(k)}(k-1)) \right) \\
&\leq |U_{l(k)}(k-1)|b + C_{l(k)} - \left(F_l(k-1) + \sum_{i \in U_{l(k)}(k-1)} \max(\eta_{l(k)}(k), \lambda_{il(k)}(k-1)) \right) \\
&= |U_{l(k)}(k-1)|b \\
&\leq Mb
\end{aligned} \tag{36}
$$

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

$$
\begin{aligned}
\lfloor \frac{\eta_{l(k)}(k)}{b} \rfloor &\leq \lfloor \frac{\eta_{\chi(s)l(k)}(k)}{b} \rfloor \\
&= \frac{\omega_s(k)}{b} \\
&< \frac{\min_{l \in L_{s(k)}} \eta_l(k)}{b} \text{ since } r_s(k) = \omega_s(k) + b
\end{aligned} \tag{37}
$$

$$
\begin{aligned}
\text{We know that } \lfloor \frac{\eta_{l(k)}(k)}{b} \rfloor &\geq \lfloor \frac{\min_{l \in L_{s(k)}} \eta_l(k)}{b} \rfloor \text{ (since } l(k) \in L_{s(k)}) \\
\text{If } \lfloor \frac{\eta_{l(k)}(k)}{b} \rfloor &> \lfloor \frac{\min_{l \in L_{s(k)}} \eta_l(k)}{b} \rfloor \\
\text{then } \lfloor \frac{\eta_{l(k)}(k)}{b} \rfloor &\geq \frac{\min_{l \in L_{s(k)}} \eta_l(k)}{b}
\end{aligned} \tag{38}
$$

But (38) contradicts (37). 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 12 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 11 and the algorithm. Thus (1) holds in this case. ∎

We prove Lemmas 6 and 7 which we used in the proof of Theorem 2. We assume that the set of feasible rate vectors is nonempty.

A. *Proof of Lemma 6*

We first state and prove Lemma 13 which we subsequently use in the proof of Lemma 6. At first, the rate allocated to a virtual session is the minimum of the session link parameters of its session in its path, and the computation of the session link parameters ensures feasibility. Next, the rate of one virtual session may be incremented by b in step (8) of the algorithm. Lemma 13 shows that this increase does not violate feasibility.

Lemma 13: If $\Lambda(k) = \phi$, for all virtual sessions $j \in S(k-1)$,

$$\Big(\max(\Omega_{\chi(j)l}(k), \omega_j(k) + b)\Big) + \sum_{i \in n(l), i \neq \chi(j)} \Omega_{il}(k) \leq C_l, \; \forall\, l \in L_j$$

Proof of Lemma 13: Consider any virtual session $j \in S(k-1)$ and a link $l \in L_j$. Since $j \notin \Lambda(k)$, either $\omega_j(k) < \Omega_{\chi(j)l}(k)$, or $\sum_{i \in n(l)} \Omega_{il}(k) \leq C_l - b$. In the first case, $\omega_j(k) + b \leq \Omega_{\chi(j)l}(k)$, since $\omega_j(k)$ and $\Omega_{\chi(j)l}(k)$ are both multiples of b from the proof of Lemma 9. Thus $\max(\Omega_{\chi(j)l}(k), \omega_j(k) + b)) = \Omega_{\chi(j)l}(k)$. Thus $\Big(\max(\Omega_{\chi(j)l}(k), \omega_j(k) + b)\Big) + \sum_{i \in n(l), i \neq \chi(j)} \Omega_{il}(k) \leq \sum_{i \in n(l)} \Omega_{il}(k)$. If $j \notin S(k-1)$, $\omega_j(k) = r_j(k-1) \leq \max(\eta_l(k), \lambda_{\chi(j)l}(k-1))$. If $j \in S(k-1)$, $\omega_j(k) \leq \max(\eta_l(k), \lambda_{\chi(j)l}(k-1))$ from the algorithm. It follows that $\Omega_{il}(k) \leq \max(\eta_l(k), \lambda_{il}(k-1))$. If $i \notin U_l(k-1)$, $\Omega_{il}(k) = \lambda_{il}(k-1)$ since $\omega_j(k) = r_j(k-1), \forall j \in m(\chi(j), l)$. It follows that $\sum_{i \in n(l)} \Omega_{il}(k) \leq \sum_{i \in n(l) \setminus U_l(k-1)} \lambda_{il}(k-1) + \sum_{i \in \setminus U_l(k-1)} \max(\eta_l(k), \lambda_{il}(k-1)) = C_l$. Hence $\Big(\max(\Omega_{\chi(j)l}(k), \omega_j(k) + b)\Big) + \sum_{i \in n(l), i \neq \chi(j)} \Omega_{il}(k) \leq C_l$

in the first case. In the second case,

$$\begin{aligned}\Big(\max(\Omega_{\chi(j)l}(k), \omega_j(k)+b)\Big) + \sum_{i\in n(l), i\neq\chi(j)} \Omega_{il}(k) &\leq b + \sum_{i\in n(l)} \Omega_{il}(k) \text{ (since } \omega_j(k) \leq \Omega_{il}(k)) \\ &\leq C_l \text{ from assumption.}\end{aligned}$$

The result holds in both cases. ∎

Proof of Lemma 6: We prove by induction. $r_s(0) = \mu_s$, for all virtual sessions s. μ_s is a multiple of b by assumption. Thus $\vec{r}(0)$ satisfies the minimum rate requirements. $\lambda_{il}(0) = \mu_{il}$, for all sessions $i \in n(l)$ and all links l. Since the set of feasible rate allocation vectors is nonempty, $\sum_{i\in n(l)} \mu_{il} \leq C_l$, for all link l. Thus $\vec{r}(0)$ is a feasible rate vector.

Let $\vec{r}(k)$ be feasible. Let the algorithm not terminate in k iterations, i.e., $S(k) \neq \phi$. We will prove that $\vec{r}(k+1)$ is feasible. $r_j(k+1)$ is a multiple of b for all virtual sessions j. $r_s(k+1) \geq r_s(k) \geq \mu_s$ for all virtual sessions s. The first inequality follows from Lemma 8 and the last from the feasibility of $\vec{r}(k)$. Thus $\vec{r}(k+1)$ satisfies the minimum rate requirements. If $j \in S(k)$, and virtual session j traverses through link l, then $\omega_j(k+1) \leq \max(\eta_l(k+1), \lambda_{\chi(j),l}(k))$. If $j \notin S(k)$, and virtual session j traverses through link l, then $\omega_j(k+1) = r_j(k) \leq \max(\eta_l(k+1), \lambda_{\chi(j),l}(k))$. It follows that

$$\Omega_{il}(k+1) \leq \max(\eta_l(k+1), \lambda_{il}(k)),\ \forall\, i \in n(l) \tag{39}$$

If $i \in n(l) \setminus U_l(k)$, $\omega_j(k+1) = r_j(k), \forall\, j \in m(i,l)$. Thus $\Omega_{il}(k+1) = \lambda_{il}(k)$ if $i \in n(l) \setminus U_l(k)$.

$$\begin{aligned}\sum_{i\in n(l)} \Omega_{il}(k+1) &= \sum_{i\in n(l)\setminus U_l(k)} \Omega_{il}(k+1) + \sum_{i\in U_l(k)} \Omega_{il}(k+1) \\ &\leq \sum_{i\in n(l)\setminus U_l(k)} \lambda_{il}(k) + \sum_{i\in U_l(k)} \max(\eta_l(k+1), \lambda_{il}(k)) \\ &= F_l(k) + \sum_{i\in U_l(k)} \max(\eta_l(k+1), \lambda_{il}(k)) \\ &\leq C_l \end{aligned} \tag{40}$$

If $r_j(k+1) = \omega_j(k+1)$, for all virtual sessions j, traversing through link l ($\chi(j) \in n(l)$), then $\lambda_{il}(k+1) = \Omega_{il}(k+1), \forall\, i \in n(l)$. Thus $\vec{r}(k+1)$ satisfies the capacity condition from (40). If $r_j(k+1) \neq \omega_j(k+1)$, for one or more virtual sessions j traversing through link l, then $r_s(k+1) = \omega_s(k+1) + b$, for some virtual session s traversing through link l and $r_j(k+1) = \omega_j(k+1), j \neq s$. Thus $\lambda_{il}(k+1) = \Omega_{il}(k+1)$, $i \neq \chi(s)$ and $\lambda_{\chi(s)l}(k+1) = \max(\omega_s(k+1) + b, \Omega_{\chi(s)l}(k+1))$.

$$\begin{aligned}\sum_{i \in n(l)} \lambda_{il}(k+1) &= \sum_{i \in n(l), i \neq \chi(s)} \Omega_{il}(k+1) + \max(\omega_s(k+1) + b, \Omega_{\chi(s)l}(k+1)) \\ &\leq C_l \text{ (from Lemma 13, since } r_s(k+1) = \omega_s(k+1) + b, s \in S(k), \text{ and } \Lambda(k+1) = \phi).\end{aligned}$$

Thus $\vec{r}(k+1)$ satisfies the capacity condition in this case as well. ∎

B. *Proof of Lemma 7*

We first present a roadmap for this proof. We first state and prove Lemma 14. We use this lemma in proving Lemma 15 which we state and prove next. We prove Lemma 7 using Lemmas 8,9 and 15.

Lemma 14: If $k \geq 1$ and the algorithm has not terminated in $k-1$ iterations, $\lfloor \frac{\eta_l(k)}{b} \rfloor \geq \lfloor \frac{\eta_l(k-1)}{b} \rfloor$.

Proof of Lemma 14: Let $k = 1$. The algorithm can not terminate in 0 iterations. $S(0) \neq \phi$, $F_l(0) = 0$. If $U_l(0) = \phi$, then $\eta_l(1) = \eta_l(0) = 0$, and the lemma holds for link l and iteration 1. Let $U_l(0) \neq \phi$. Since the feasible set of rate vectors is nonempty, $\sum_{i \in U_l(0)} \mu_{il} \leq C_l$ where $\mu_{il} = \max_{j \in m(i,l)} \mu_j$. $\mu_{il} \geq 0$, for every session i and link l. $r_j(0) = \mu_j$. Thus $\lambda_{il}(0) = \mu_{il}$. Thus $\theta = 0$ satisfies the inequality $\sum_{i \in U_l(0)} \max(\theta, \lambda_{il}(0)) \leq C_l$. Clearly $\eta_l(1)$ is the maximum possible θ which satisfies the above inequality. Hence $\eta_l(1) \geq 0 = \eta_l(0)$. The equality follows from the initialization of $\eta_l(0)$. Thus the lemma holds for $k = 1$.

Let the lemma hold for iterations $1, \ldots, k$ and $S(k) \neq \phi$, i.e., $\lfloor \frac{\eta_l(1)}{b} \rfloor \leq \ldots \leq \lfloor \frac{\eta_l(k)}{b} \rfloor$. Thus there exists $\eta_l(p), p = 1, \ldots, k$. We will show that the lemma holds for the $k+1$th iteration. If $j \notin S(k-1)$, $r_j(k) = \omega_j(k) = r_j(k-1)$. If $l \in L_j$, $r_j(k-1) \leq \lambda_{\chi(j),l}(k-1) \leq \max(b \lfloor \frac{\eta_l(k)}{b} \rfloor, \lambda_{\chi(j),l}(k-1))$.

$$\begin{aligned} r_j(k) &\leq \max(b \lfloor \frac{\eta_l(k)}{b} \rfloor, \lambda_{\chi(j),l}(k-1)) \text{ if } j \notin S(k-1), l \in L_j && (41) \\ &\leq \max(\eta_l(k), \lambda_{\chi(j),l}(k-1)) \text{ if } j \notin S(k-1), l \in L_j && (42)\end{aligned}$$

$r_j(k) \geq \omega_j(k)$ for all virtual sessions j. Consider a link l, s.t. $U_l(k-1) \neq \phi$.

1) Let $r_j(k) = \omega_j(k)$ for all virtual sessions j traversing through link l. If $j \in S(k-1)$, and virtual session j traverses through link l, then $r_j(k) = \omega_j(k) \leq \eta_{\chi(j)l}(k)$. $\eta_{\chi(j)l}(k) = \max(\eta_l(k), \lambda_{il}(k-1)), \forall j \in m(i,l)$. Thus if $r_j(k) = \omega_j(k)$, for all virtual sessions j traversing through l

$$r_j(k) \leq \max(\eta_l(k), \lambda_{\chi(j),l}(k-1)) \text{ if } j \in S(k-1), l \in L_j \tag{43}$$

$$\lambda_{il}(k) \leq \max(\eta_l(k), \lambda_{il}(k-1)),\ \forall\ i \in n(l) \qquad \text{(from (42) and (43))} \tag{44}$$

$$\begin{aligned}
\sum_{i \in U_l(k)} \max(\eta_l(k), \lambda_{il}(k)) &\leq \sum_{i \in U_l(k)} \max(\eta_l(k), \lambda_{il}(k-1)) \text{ (from (44))} && (45)\\
F_l(k) &= F_l(k-1) + \sum_{i \in U_l(k-1) \setminus U_l(k)} \lambda_{il}(k) \\
&\leq F_l(k-1) + \\
&\qquad \sum_{i \in U_l(k-1) \setminus U_l(k)} \max(\eta_l(k), \lambda_{il}(k-1)) \text{ (from (44))} && (46)\\
F_l(k) + \sum_{i \in U_l(k)} \max(\eta_l(k), \lambda_{il}(k)) &\leq F_l(k-1) + \sum_{i \in U_l(k-1) \setminus U_l(k)} \max(\eta_l(k), \lambda_{il}(k-1)) + \\
&\quad \sum_{i \in U_l(k)} \max(\eta_l(k), \lambda_{il}(k-1)) \text{ (from (45) and (46))} \\
&= F_l(k-1) + \sum_{i \in U_l(k-1)} \max(\eta_l(k), \lambda_{il}(k-1)) && (47)\\
&\quad \text{(since } U_l(k) \subseteq U_l(k-1) \text{ as } S(k) \subseteq S(k-1)\text{)} \\
&= C_l && (48)
\end{aligned}$$

2) Now let there exist a virtual session s traversing through link l such that $r_s(k) > \omega_s(k)$, and $\lfloor \frac{\min_{l_1 \in L_s} \eta_{l_1}(k)}{b} \rfloor < \lfloor \frac{\eta_l(k)}{b} \rfloor$. Since $r_s(k) > \omega_s(k)$, $r_s(k) = \omega_s(k) + b$. $\omega_s(k)$ is divisible by b (proof of Lemma 9) and $\omega_s(k) < \min_{l_1 \in L_s} \eta_{l_1}(k)$. Thus

$$\begin{aligned}
\omega_s(k) &\leq b\lfloor \frac{\min_{l_1 \in L_s} \eta_{l_1}(k)}{b} \rfloor \text{ (proof of Lemma 9)} \\
&< b\lfloor \frac{\eta_l(k)}{b} \rfloor \text{ by hypothesis.} \\
\text{Thus } \omega_s(k) + b &\leq b\lfloor \frac{\eta_l(k)}{b} \rfloor \text{ (by Lemma 9)}
\end{aligned}$$

$$\begin{aligned} r_s(k) &= \omega_s(k) + b \\ &\leq b\lfloor\frac{\eta_l(k)}{b}\rfloor \\ &\leq \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{\chi(s)l}(k-1)) \end{aligned} \tag{49}$$

Since there exists at most one virtual session j such that $r_j(k) > \omega_j(k)$, $r_j(k) = \omega_j(k)$, $j \neq s$. Consider a virtual session j traversing through link l. Let $j \in S(k-1)$ and $j \neq s$.

$$\begin{aligned} \omega_j(k) &\leq b\lfloor\frac{\eta_{\chi(j)l}(k)}{b}\rfloor \ \forall\, j \\ &= \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{\chi(j)l}(k-1)) \text{ (by Lemma 9)} \\ \text{Thus } r_j(k) &\leq \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{\chi(j)l}(k-1)) \ \forall\, j \in S(k-1), j \neq s \qquad (50) \\ \lambda_{il}(k) &\leq \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{il}(k-1)) \ \forall\, i \in n(l) \qquad (51) \end{aligned}$$

(from (41), (49) and (50))

Using (51) it can be proved that

$$\begin{aligned} F_l(k) + \sum_{i \in U_l(k)} \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{il}(k)) &\leq F_l(k-1) + \\ &\quad \sum_{i \in U_l(k-1)} \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{il}(k-1)) \qquad (52) \\ &\leq F_l(k-1) + \sum_{i \in U_l(k-1)} \max(\eta_l(k), \lambda_{il}(k-1)) \\ &\quad \text{(since } \eta_l(k) \geq b\lfloor\frac{\eta_l(k)}{b}\rfloor) \\ &\leq C_l \text{ (by induction hypothesis)} \qquad (53) \end{aligned}$$

The proof for (52) is exactly similar to the one for (47) using (44).

3) Now let there exist a virtual session s traversing link l such that $r_s(k) > \omega_s(k)$ and $\lfloor\frac{\min_{l_1 \in L_s} \eta_{l_1}(k)}{b}\rfloor = \lfloor\frac{\eta_l(k)}{b}\rfloor$. As before, $r_s(k) = \omega_s(k) + b$. First let $r_s(k) \leq \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{\chi(s)l}(k-1))$

$$\begin{aligned} r_j(k) &= \omega_j(k), j \neq s \\ &\leq \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{il}(k-1)) \text{ if } j \in S(k-1) \end{aligned} \tag{54}$$

$$\lambda_{il}(k) \leq \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{il}(k-1)) \; \forall i \in n(l) \tag{55}$$

(from (41), assumption and (54))

Now using (55) it can be proved that

$$F_l(k) + \sum_{i \in U_l(k)} \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{il}(k)) \leq C_l \tag{56}$$

The proof is exactly similar to the one for (53) using (51).

Now let

$$r_s(k) > \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{il}(k-1)) \tag{57}$$

Observe that $\forall i \in U_l(k-1)$,

$$\begin{aligned} \lambda_{il}(k) &\geq \Omega_{il}(k) \\ &\geq b\lfloor\frac{\eta_l(k)}{b}\rfloor \text{ (from the algorithm)} \end{aligned}$$

The last step follows from the algorithm since $r_s(k) = \omega_s(k) + b$, $l \in l_s$ and $\lfloor\frac{\min_{l_1 \in L_s} \eta_{l_1}(k)}{b}\rfloor = \lfloor\frac{\eta_l(k)}{b}\rfloor$.

$$\begin{aligned} \text{Thus } \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{il}(k)) &= \lambda_{il}(k) \\ &= \Omega_{il}(k) \qquad (58) \\ &\quad \text{(if } i \neq \chi(s), \text{ since then } r_j(k) = \omega_j(k) \; \forall j \in m(i,l)) \end{aligned}$$

Now $r_s(k) = \omega_s(k) + b$. Thus $s \in S(k-1)$.

$$\begin{aligned} \omega_s(k) &\leq \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{\chi(s)l}(k-1)) \text{ (since } l \in L_s) \\ r_s(k) &\leq \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{\chi(s)l}(k-1)) + b \qquad (59) \end{aligned}$$

From (57), (59) and since $r_s(k)$ is a multiple of b, by Lemma 9, $r_s(k) = \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{\chi(s)l}(k-1)) + b$.

$$\text{If } \chi(j) \in n(l), \omega_j(k) \leq \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{\chi(j)l}(k-1)), j \in S(k-1) \tag{60}$$

$$\begin{aligned}
\omega_j(k) &= r_j(k-1),\ j \notin S(k-1) \\
&\le \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{\chi(j)l}(k-1)), j \notin S(k-1) \qquad (61) \\
\Omega_{\chi(s)l}(k) &\le \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{\chi(s)l}(k-1)) \\
&\qquad \text{(from (60) and (61))} \\
\text{Thus } \max(\Omega_{\chi(s)l}(k), r_s(k)) &= \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{\chi(s)l}(k-1)) + b \qquad (62) \\
r_j(k) &= \omega_j(k), j \neq s \\
r_s(k) &= \omega_s(k) + b \\
\text{Thus } \lambda_{\chi(s)l}(k) &= \max(\Omega_{\chi(s)l}(k), r_s(k)) \\
&= \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{\chi(s)l}(k-1)) + b \text{ (from (62))} \qquad (63) \\
\max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{\chi(s)l}(k)) &= \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{\chi(s)l}(k-1)) + b \text{ (from (63))} \\
&= \max(\Omega_{\chi(s)l}(k), r_s(k)) \text{ (from (62))} \qquad (64)
\end{aligned}$$

Thus if $r_s(k) > \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{il}(k-1))$,

$$\begin{aligned}
& F_l(k) + \sum_{i \in U_l(k)} \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{il}(k)) \\
= \ & F_l(k-1) + \sum_{i \in U_l(k-1)\setminus U_l(k)} \lambda_{il}(k) + \sum_{i \in U_l(k)} \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{il}(k)) \\
& \qquad \text{(since } U_l(k) \subseteq U_l(k-1)\text{)} \\
\le \ & F_l(k-1) + \sum_{i \in U_l(k-1)\setminus U_l(k)} \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{il}(k)) + \sum_{i \in U_l(k)} \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{il}(k)) \\
= \ & F_l(k-1) + \sum_{i \in U_l(k-1)} \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{il}(k)) \\
& \qquad \text{(since } U_l(k-1) \supseteq U_l(k)\text{)} \\
= \ & F_l(k-1) + \sum_{i \in U_l(k-1), i \neq \chi(s)} \Omega_{il}(k) + \max(\Omega_{\chi(s)l}(k), r_s(k)) \\
& \quad \text{(since } \chi(s) \in U_l(k-1)\text{) (from (58) and (64))} \\
= \ & \sum_{i \in n(l)\setminus U_l(k-1)} \Omega_{il}(k) + \sum_{i \in U_l(k-1), i \neq \chi(s)} \Omega_{il}(k) + \max(\Omega_{\chi(s)l}(k), r_s(k))
\end{aligned}$$

$$
\begin{aligned}
& \quad \text{(since } \omega_j(k) = r_j(k-1) \forall j \in n(l) \setminus S(k-1)) \\
= & \quad \sum_{i \in n(l), i \neq \chi(s)} \Omega_{il}(k) + \max(\Omega_{\chi(s)l}(k), w_s(k) + b) \\
\leq & \quad C_l
\end{aligned} \tag{65}
$$

(65) follows since from Lemma 13, since $r_s(k) = w_s(k) + b, s \in S(k-1)$ and $\Lambda(k) = \phi$.

Since virtual session s traverses link l, $\lfloor \frac{\min_{l_1 \in L_s} \eta_{l_1}(k)}{b} \rfloor \leq \lfloor \frac{\eta_l(k)}{b} \rfloor$. Thus (2) and (3) are the only possibilities, when $r_s(k) > \omega_s(k)$ for some virtual session s traversing through link l. (1) covers the other case, i.e., when $r_j(k) = \omega_j(k)$ for all virtual sessions traversing the link l. From (48), (53), (56) and (65), $\theta = b\lfloor \frac{\eta_l(k)}{b} \rfloor$ satisfies the inequality $F_l(k) + \sum_{i \in U_l(k)} \max(\theta, \lambda_{il}(k)) \leq C_l$. Clearly $\eta_l(k+1)$ is the maximum possible value of θ which satisfies the inequality and hence there exists a $\eta_l(k+1)$ and $\eta_l(k+1) \geq b\lfloor \frac{\eta_l(k)}{b} \rfloor$. Thus $\lfloor \frac{\eta_l(k+1)}{b} \rfloor \geq \lfloor \frac{\eta_l(k)}{b} \rfloor$. Hence the result follows from induction. ∎

Lemma 15: If $l \in L_s$ and $r_s(k) > \mu_{\chi(s)l}$, then $r_s(k) \leq b\lceil \frac{\eta_l(k)}{b} \rceil$ for all virtual sessions s and all iterations $k \geq 0$.

Lemma 15 states that the rate allocated to a virtual session s in iteration k can exceed the link control parameter (or rather a suitably discretized version of the link control parameter) only because of the minimum rate requirements of its session in its path.

Proof of Lemma 15: $r_s(0) = \mu_s \leq \mu_{\chi(s)l}$ $\forall l \in L_s$ and all virtual sessions s. Thus the lemma is trivially true for $k = 0$. Let the lemma hold for iterations $0, 1, \ldots, k$, $k \geq 0$. We will prove that the lemma holds for the $k+1$th iteration. Let $l \in L_s$. Let $r_s(k+1) > \mu_{\chi(s)l}$. If $s \notin S(k)$, $r_s(k+1) = \omega_s(k+1) = r_s(k)$. Thus $r_s(k) > \mu_{\chi(s)l}$. By induction hypothesis, $r_s(k) \leq b\lceil \frac{\eta_l(k)}{b} \rceil$. Thus $r_s(k+1) = r_s(k) \leq b\lceil \frac{\eta_l(k)}{b} \rceil$. By Lemma 14, $\lfloor \frac{\eta_l(k)}{b} \rfloor \leq \lfloor \frac{\eta_l(k+1)}{b} \rfloor$. It follows that $\lceil \frac{\eta_l(k)}{b} \rceil \leq \lceil \frac{\eta_l(k+1)}{b} \rceil$. Thus $r_s(k+1) \leq b\lceil \frac{\eta_l(k+1)}{b} \rceil$.

Now let $s \in S(k)$. Let $r_s(k+1) = \omega_s(k+1)$. We need to show that $\omega_s(k+1) \leq b\lceil \frac{\eta_l(k+1)}{b} \rceil$. Since $s \in S(k)$, $\omega_s(k+1) \leq \max(\eta_l(k+1), \lambda_{\chi(s)l}(k))$.

1) Let $\omega_s(k+1) \leq \lambda_{\chi(s)l}(k)$. If $\lambda_{\chi(s)l}(k) > \mu_{\chi(s)l}$, there exists $j \in m(\chi(s), l)$, such that $r_j(k) = \lambda_{\chi(s)l}(k) > \mu_{\chi(s)l}$. By induction hypothesis, $r_j(k) \leq b\lceil \frac{\eta_l(k)}{b} \rceil$. As argued before, using Lemma 14,

$\lceil \frac{\eta_l(k)}{b} \rceil \leq \lceil \frac{\eta_l(k+1)}{b} \rceil$. Thus $\omega_s(k+1) \leq \lambda_{\chi(s)l}(k) = r_j(k) \leq b\lceil \frac{\eta_l(k+1)}{b} \rceil$.

2) Let $\omega_s(k+1) \leq \eta_l(k+1)$. $\eta_l(k+1) \leq b\lceil \frac{\eta_l(k+1)}{b} \rceil$. Hence $\omega_s(k+1) \leq b\lceil \frac{\eta_l(k+1)}{b} \rceil$.

Now let $r_s(k+1) = \omega_s(k+1) + b$. (For $s \in S(k)$, the only possibilities are $r_s(k+1) = \omega_s(k+1)$ and $r_s(k+1) = \omega_s(k+1) + b$.) Thus $\omega_s(k+1) < \min_{l_1 \in L_s} \eta_{l_1}(k+1)$. Since $l \in L_s$ $\min_{l_1 \in L_s} \eta_{l_1}(k+1) \leq \eta_l(k+1)$. Thus $\omega_s(k+1) < \eta_l(k+1)$. Since $\omega_s(k+1)$ is a multiple of b (proof of Lemma 9), $\omega_s(k+1) \leq b\lfloor \frac{\eta_l(k+1)}{b} \rfloor$, if $\eta_l(k+1)$ is not a multiple of b and $\omega_s(k+1) \leq \eta_l(k+1) - b$, if $\eta_l(k+1)$ is a multiple of b. If $\eta_l(k+1)$ is not a multiple of b, then $\lfloor \frac{\eta_l(k+1)}{b} \rfloor = \lceil \frac{\eta_l(k+1)}{b} \rceil - 1$ and thus $\omega_s(k+1) + b \leq b\lceil \frac{\eta_l(k+1)}{b} \rceil$. If $\eta_l(k+1)$ is a multiple of b, then clearly $\omega_s(k+1) + b \leq \eta_l(k+1) = b\lceil \frac{\eta_l(k+1)}{b} \rceil$. Since $r_s(k+1) = \omega_s(k+1) + b$, the result follows for the $k+1$th iteration. The lemma follows by induction.

∎

We discuss the key steps in proving Lemma 7. Let a virtual session s be saturated with respect to rate allocation $\vec{r}(k)$. Then consider the first iteration t, $t \leq k$ in which s joined the set of saturated virtual sessions. From the definition of saturation there exists a link l in L_s which satisfies the first two conditions of a pseudo-bottleneck link with respect to $\vec{r}(t)$. Since the rates of virtual sessions do not decrease in subsequent iterations (Lemma 8), link l satisfies the first two conditions of a pseudo-bottleneck link with respect to $\vec{r}(k)$ as well. The main challenge is to show that link l satisfies the third condition of a pseudo-bottleneck link. The formal proof follows.

Proof of Lemma 7: Since $S(0) = \{1, \ldots, M\}$, $s \in S(0)$ for all virtual sessions s. Thus the lemma holds by vacuity for $k = 0$. Consider $k > 0$. Let $s \notin S(k)$. There exists t such that $s \in S(t) \setminus S(t+1)$, $t + 1 \leq k$. $s \in S(t) \setminus S(t+1)$ implies that for some link $l \in L_s$,

$$\sum_{i \in n(l)} \lambda_{il}(t+1) > C_l - b \tag{66}$$

$$r_s(t+1) = \lambda_{\chi(s)l}(t+1) \tag{67}$$

$$\sum_{i \in n(l)} \lambda_{il}(k) \geq \sum_{i \in n(l)} \lambda_{il}(t+1) \text{ (from Lemma 8 and since } t+1 \leq k)$$

$$> \quad C_l - b \text{ (from (66))} \tag{68}$$

From (68), link l satisfies the first condition of a pseudo-bottleneck link with respect to $\vec{r}(k)$.

From Lemma 8, $\lambda_{il}(t+1) \leq \lambda_{il}(k)$, since $t+1 \leq k$. Let there exist $u \in n(l)$ such that $\lambda_{ul}(t+1) < \lambda_{ul}(k)$. Then $\lambda_{ul}(k) \geq \lambda_{ul}(t+1) + b$ (Lemma 9).

$$\begin{aligned} C_l &< b + \lambda_{ul}(t+1) + \sum_{i \in n(l), i \neq u} \lambda_{il}(t+1) \text{ (from (66))} \\ &\leq \lambda_{ul}(k) + \sum_{i \in n(l), i \neq u} \lambda_{il}(k) \\ &= \sum_{i \in n(l)} \lambda_{il}(k) \end{aligned}$$

Thus $\vec{r}(k)$ is infeasible. This contradicts Lemma 6.

$$\text{Thus } \lambda_{il}(k) = \lambda_{il}(t+1), \forall\, i \in n(l). \tag{69}$$

Since $t+1 \leq k$, $r_s(t+1) \leq r_s(k)$ (Lemma 8). From (67) and (69), $r_s(k) \geq \lambda_{\chi(s)l}(k)$. Clearly from definition $r_s(k) \leq \lambda_{\chi(s)l}(k)$. Thus $r_s(k) = \lambda_{\chi(s)l}(k)$. Thus link l satisfies the second condition of a pseudo-bottleneck link with respect to $\vec{r}(k)$.

Now we show that link l satisfies the third condition of a pseudo-bottleneck link with respect to $\vec{r}(k)$. We will show that

$$r_s(t+1) + b \geq b\lceil \frac{\eta_l(t+1)}{b} \rceil \tag{70}$$

$$\begin{aligned} r_s(k) + b &\geq r_s(t+1) + b \text{ (from Lemma 8)} \\ &\geq b\lceil \frac{\eta_l(t+1)}{b} \rceil \text{ (from 70)} \\ &\geq r_j(t+1) \text{ if } \chi(j) \in n(l), r_j(t+1) > \mu_{\chi(j)l} \text{ (from Lemma 15)} \\ \text{Thus } r_s(k) + b &\geq \lambda_{il}(t+1) \text{ if } i \in n(l), \lambda_{il}(t+1) > \mu_{il} \\ &= \lambda_{il}(k) \text{ if } i \in n(l), \lambda_{il}(t+1) > \mu_{il} \text{ (from (69))} \\ &\geq r_j(k) \text{ if } \chi(j) \in n(l), r_j(t+1) > \mu_{\chi(j)l} \end{aligned}$$

The result follows.

Now we show (70). The intuition behind the result is that virtual session s saturates because of the bandwidth constraints in its pseudo-bottleneck link l, and thus the rate allocated to s must be lower bounded by the link control parameter within an additive factor of b. We consider two cases separately for this proof.

(Case 1): Let $\lambda_{il}(t+1) \leq \max(\eta_l(t+1), \lambda_{il}(t)), \forall i \in n(l)$. If $i \in n(l) \setminus U_l(t)$, $\lambda_{il}(t+1) = \lambda_{il}(t)$ and $F_l(t) = \sum_{i \in n(l) \setminus U_l(t)} \lambda_{il}(t) = \sum_{i \in n(l) \setminus U_l(t)} \lambda_{il}(t+1)$.

$$\begin{aligned}
C_l &< \sum_{i \in n(l) \setminus \{\chi(s)\}} \lambda_{il}(t+1) + (\lambda_{\chi(s)l}(t+1) + b) \qquad \text{(from (66))} \\
&= \sum_{i \in n(l) \setminus U_l(t)} \lambda_{il}(t+1) + \sum_{i \in U_l(t), i \neq \chi(s)} \lambda_{il}(t+1) + \\
&\quad (b + \lambda_{\chi(s)l}(t+1)) \text{ (since } s \in S(t), \chi(s) \in U_l(t)) \\
&\leq F_l(t) + \sum_{i \in U_l(t), i \neq \chi(s)} \max(\eta_l(t+1), \lambda_{il}(t)) + \\
&\quad (b + \lambda_{\chi(s)l}(t)) \\
C_l &\geq F_l(t) + \sum_{i \in U_l(t), i \neq \chi(s)} \max(\eta_l(t+1), \lambda_{il}(t)) + \\
&\quad \max(\eta_l(t+1), \lambda_{\chi(s)l}(t)) \\
\text{Thus } \lambda_{\chi(s)l}(t) + b &> \max(\eta_l(t+1), \lambda_{\chi(s)l}(t)) \\
&\geq \eta_l(t+1) \\
\text{Thus } r_s(t+1) + b &> \eta_l(t+1) \text{ (from (67))} \\
r_s(t+1) + b &\geq b\lceil \frac{\eta_l(t+1)}{b} \rceil \text{ (by Lemma 9)}
\end{aligned}$$

Thus (70) holds in this case.

(Case 2): Let there exist $i \in n(l)$ such that $\lambda_{il}(t+1) > \max(\eta_l(t+1), \lambda_{il}(t))$. Thus $r_j(t+1) > \max(\eta_l(t+1), \lambda_{il}(t))$ for at least one $j \in m(i,l)$. Observe that $\omega_u(t+1) \leq \max(\eta_l(t+1), \lambda_{il}(t))$, if $u \in S(t) \cap m(i,l)$. $\omega_u(t+1) = r_u(t) \leq \max(\eta_l(t+1), \lambda_{il}(t))$, if $u \in m(i,l) \setminus S(t)$. It follows that $\omega_u(t+1) \leq \max(\eta_l(t+1), \lambda_{il}(t))$, $\forall u \in m(i,l)$. Thus $r_j(t+1) > \omega_j(t+1)$. It follows that

$r_j(t+1) = \omega_j(t+1) + b$. This means that $\omega_j(t+1) < \min_{l_1 \in L_j} \eta_l(t+1)$. From Lemma 9 $\omega_j(t+1) \leq b\lfloor\frac{\min_{l_1 \in L_j} \eta_{l_1}(t+1)}{b}\rfloor$. Thus $r_j(t+1) \leq b\lfloor\frac{\min_{l_1 \in L_j} \eta_{l_1}(t+1)}{b}\rfloor + b$. Since $r_j(t+1) > \eta_l(t+1)$, $\eta_l(t+1) < b\lfloor\frac{\min_{l_1 \in L_j} \eta_{l_1}(t+1)}{b}\rfloor + b$. Since $l \in L_j$, $\lfloor\frac{\min_{l_1 \in L_j} \eta_{l_1}(t+1)}{b}\rfloor \leq \lfloor\frac{\eta_l(t+1)}{b}\rfloor \leq \frac{\eta_l(t+1)}{b} < \lfloor\frac{\min_{l_1 \in L_j} \eta_{l_1}(t+1)}{b}\rfloor + 1$. Thus

$$\lfloor\frac{\eta_l(t+1)}{b}\rfloor = \lfloor\frac{\min_{l_1 \in L_j} \eta_{l_1}(t+1)}{b}\rfloor \tag{71}$$

Since s traverses l and $s \in S(t)$, $\chi(s) \in U_l(t)$. $l \in L_j$. Thus from (71), and the fact that $r_j(t) = \omega_j(t)+b$, $\Omega_{\chi(s)l}(t+1) \geq b\lfloor\frac{\eta_l(t+1)}{b}\rfloor$. Since $\lambda_{\chi(s)l}(t+1) \geq \Omega_{\chi(s)l}(t+1)$,

$$\begin{aligned}
\lambda_{\chi(s)l}(t+1) &\geq b\lfloor\frac{\eta_l(t+1)}{b}\rfloor \\
r_s(t+1) &\geq b\lfloor\frac{\eta_l(t+1)}{b}\rfloor \text{ (from (67))} \\
\text{Thus } r_s(t+1) + b &\geq b\lceil\frac{\eta_l(t+1)}{b}\rceil
\end{aligned}$$

Thus (70) holds in this case as well. ∎

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

TABLE I

SUMMARY OF SYMBOLS USED THROUGHOUT THE PAPER

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

TABLE II

SUMMARY OF SYMBOLS USED IN THE RATE COMPUTATION ALGORITHM