TECHNICAL RESEARCH REPORT

Fair Allocation of Discrete Bandwidth Layers in Multicast Networks

by Saswati Sarkar, Leandros Tassiulas

T.R. 99-43

Fair Allocation of Discrete Bandwidth Layers in Multicast Networks

Saswati Sarkar and *Leandros Tassiulas*
Dept. of Electrical and Computer Engineering and Institute for Systems Research
University of Maryland, College Park, MD, USA
email addresses: swati@eng.umd.edu, leandros@isr.umd.edu

Abstract

We study fairness when receivers in a multicast network can not subscribe to fractional layers. This case arises when the source hierarchically encodes its signal and the hierarchical structure is predetermined. Unlike the case of the fractional layer allocation, which has been studied extensively in [29], bandwidth can be allocated in discrete chunks only. Fairness issues become vastly different. Computation of lexicographically optimal rate allocation becomes NP-hard in this case, while lexicographically optimal rate allocation is polynomial complexity computable when fractional layers can be allocated. Furthermore, maxmin fair rate vector may not exist in this case. We introduce a new notion of fairness, maximal fairness. We propose a polynomial complexity algorithm for computation of maximally fair rates allocated to various source-destination pairs. Even though, maximal fairness is a weaker notion of fairness, it coincides with lexicographic optimality and maxmin fairness, when maxmin fair rate allocation exists. So the algorithm for computing maximally fair rate allocation computes maxmin fair rate allocation, when the latter exists.

1 Introduction

Multicasting provides an efficient way of transmitting data from a sender to a group of receivers. A single source node or a group of source nodes sends identical messages simultaneously to multiple destination nodes. Single destination or unicast and broadcast to the entire network are special cases of multicast. Multicast applications include collaborative applications like audio or video teleconferencing, video-on-demand services, distributed databases, distribution of software, financial information, electronic newspapers, billing records, medical images, weather maps and experimental data, distributed interactive simulation (DIS) activities such as tank battle simulations. Many distributed systems such as the V System[12] and the Andrew

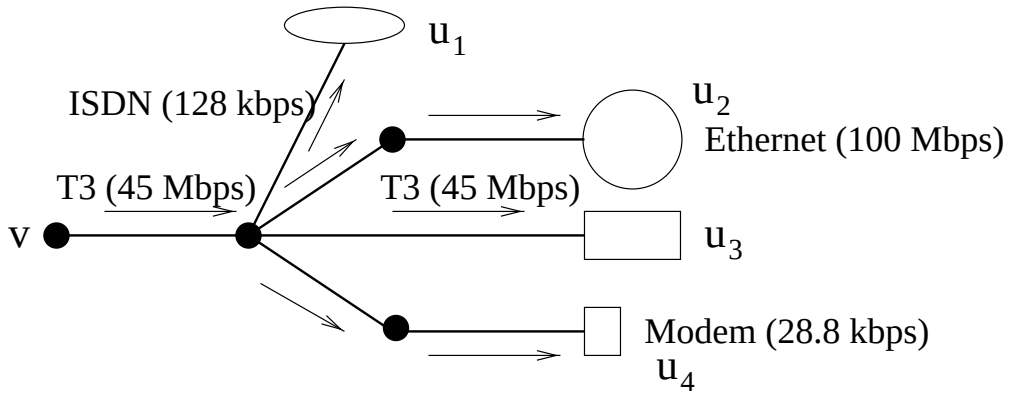

Figure 1:

distributed computing environment[27], popular protocol suites like Sun's broadcast RPC service[22] and IBMs NetBIOS[18] are using multicasting. Multicasting has been used primarily in the Internet, but future ATM networks are likely to deploy multicasting in a large scale, particularly in applications like broadcast video, video-conferencing, multiparty telephony and workgroup applications[11]. In general many multicast sessions simultaneously share the network resources. Ideally all sessions should have a fair share of bandwidth. This issue of *inter-session* fairness have been studied extensively in unicast networks. Multicasting poses some specific challenges in this regard. This is because of network heterogenity. A single session may have many destinations, and end systems can have widely varying bandwidth connectivities. On one hand there are fast ethernets (100 Mbps) and on the other hand there are slow modems (28.8 kbps). The paths to different destinations may have different bandwidth capacities, e.g., one may consist of multi-megabit links, such as, $T3$ (45 Mbps) and another may have a 128 kbps ISDN line (Refer to the network shown in figure 1 for an example). Every receiver would like to receive service at a rate commensurate with its capabilities and the capacity of the path leading to it from the source independent of the capabilities of the other receivers of the same session. This is the issue of *intra-session* fairness. Besides, like in unicast, there is the issue of *inter-session* fairness, that is fairness of members across multiple sessions. So multicasting poses the issue of *intra-session* fairness in addition to that of *inter-session* fairness.

As the Internet evolves to higher speed and larger size, the problems caused by heterogenity will only get worse. A single rate of transmission per session is likely to either overwhelm the slow receivers or starve the fast ones, in absence of additional provisions. There are three alternative approaches, simulcast, transcoding and layered transmission. Simulcast advocates that every source maintains multiple streams carrying the same information but transmitted at different rates and quality, targetted at receivers with different capabilities[9]. Depending upon the individual capabilities, the receivers are partitioned across groups and each group subscribes to one stream. The rate of the streams can be controlled to attain a fair share. However this is bandwidth inefficient as the same basic information is replicated across all streams. This contradicts the basic principle of multicast that messages need only be replicated at forking nodes. Besides, unless there are as many groups as the number of receivers, the problem of heterogenity remains to a limited extent. In transcoding, the source transmits at a rate matching the fastest of its receivers. The transmission rates are transcoded at the intermediate nodes to match the capabilities of slower receivers downstream[30]. At every link, the transmission rate of a session is equal to that of the fastest session receiver downstream of the link. Video gateways are generally used for transcoding[2]. However, in many situations, transcoding introduces unacceptable delay. As we discuss later, it introduces security hazards and is a cost-prohitive option. The last approach is to have a hierarchical or a layered transmission scheme. In this approach, a signal is encoded into a number of layers that can be incrementally combined to provide progressive refinement. The different layers of a multicast group are considered different multicast groups and receivers adapt to congestion by adding and dropping layers, where adding a layer is joining a multicast group and dropping a layer is leaving a group[14]. Again the number of layers of a session in a link is the maximum of the number of layers of the session receivers downstream. This layered transmission scheme have been used for both video[30] and audio[7] transmissions over the internet and has potentials for use in ATM networks as well[15]. Layered transmission schemes can be used to attain inter-session and intra-session fairness, in a bandwidth efficient manner, by having the receivers subscribe to a "fair" number of layers. We assume that the network transmits hierarchically encoded signals. As we discuss later, the layer bandwidths are often predetermined and can not be tuned according to the needs of the network. A receiver either receives a layer fully or does not receive the

layer, at all. It can not partially subscribe to a layer, unlike [29]. Effectively, the network can only allocate a discrete set of rates to the receivers, whereas a continuous set of rates can be allocated when receivers can subscribe to fractional layers[29]. We study fair allocation of rates under this additional constraint. As it turns out, fairness in a discrete set is vastly different from that in a continuous set.

Maxmin fairness[6] is a well accepted notion of fairness. We would define this notion more precisely later, but informally speaking, a rate allocation is maxmin fair, if no receiver can be allocated a higher rate without hurting another receiver having equal or lower rate. A maxmin fair rate allocation may not exist in a discrete set. However, a maxmin fair rate allocation always exists in a continuous set. Lexicographic optimality is another notion of fairness. Formal definition follows later, but informally speaking, a lexicographically optimal rate vector is one which maximizes its minimum component in a feasible set, subject to this maximization, it maximizes the second minimum, etc. Lexicographically optimal rate allocation exists in a discrete set, but as we prove later, its computation is an NP-hard problem. However lexicographically optimal rate allocation is identical to the maxmin fair rate allocation and is thus polynomial complexity computable in a continuous set[26][29]. We can compute a maximally fair rate allocation in a discrete set, instead. We introduce the concept of maximal fairness more formally later, but a rate allocation is maximally fair if no other rate allocation is "fairer" in some sense. That is, if a rate allocation is maximally fair, then to increase the rate of a receiver s, we must lower that of another receiver j to a value less than the new rate of s. If a rate allocation is maximally fair, then any other rate allocation will be "unfair" or "less fair" to some receiver. Maximally fair rate allocation is a weaker notion of fairness as compared to maxmin fairness and lexicographic optimality. But, maximal fairness has various desirable properties fairness properties,e.g., it coincides with maxmin fairness and lexicographic optimality when maxmin fair rate allocation exists. We discuss other desirable fairness properties of maximally fair allocations later. In a nutshell, maximal fairness is probably the best we can achieve in the discrete case in view of the nonexistence of maxmin fair rate allocation and computational complexity of lexicographically optimal rate allocation. We will present a polynomial complexity algorithm for computing maximally fair rate allocation in this paper. This algorithm yields a maxmin fair rate vector, if it exists. Our algorithm for computation of

maximally fair allocation do not assume any properties specific to internet or ATM. So it is applicable in a very general scenario. Keeping in mind ATM networks, we have incorporated minimum rate requirements and maximum rate constraints in our model.

We review related work briefly The problem of fair allocation of bandwidths to multicast sessions under the constraint that all receivers of the same session must receive service at the same rate has been studied in [31]. Intra-session fairness can not be achieved by a single rate of transmission per session on account of network heterogenity. [9] advocates simulcast, but that is bandwidth inefficient. There are two well known network protocols for layered transmission, RLM (Receiver-driven Layered Multicast)[24] and LVMR (Layered Video Multicast with Retransmissions)[19]. The goal of these approaches is to achieve improved intra-session fairness. However, as [20] points out, neither handles inter-session fairness very well, when there are multiple sessions competing for bandwidth. A scheme for fair allocation of layers for multi-session layered video multicast which strives to rectify this defect in RLM and LVMR has been proposed in [20]. The authors present empirical evidence that the scheme improves inter-session fairness for networks with multiple video sessions sharing only one link. They mention that if M video streams share a link and no stream has bandwidth constraint on other links or end systems, then the layer difference between any two streams is either 0 or 1 in the steady state. But typically, streams would have bandwidth constraint on other links as well. There is no experimental or analytical evidence that the scheme works well for more complex networks, with sessions sharing several links with each other. In absence of further mechanisms, like elaborate scheduling policies, it may not be possible to establish conclusively that the scheme attains fair allocation of rates as per some well defined notion of fairness, like maxmin fairness for example. Besides [20] does not make any effort towards the computation of the actual rates or the number of layers allocated to the receivers in an arbitrary network, under some well defined notion of fairness. Rubenstein *et. al.* also points out that maxmin fair rate allocations may not exist for discrete bandwidth layers[26]. But, [26] suggests a policy of coordinated random add and drop of the highest layer for various receivers, as a remedy. This attains long term rates close to the maxmin fair allocation. However, this oscillation is likely to produce perceptually annoying distortion. The resulting perceptual quality may even be worse than not subscribing to the highest layer at all. Besides, [26] observes that this

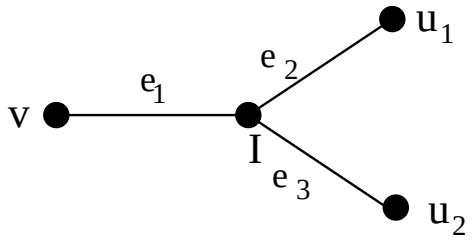

Figure 2:

random add and drop of highest layer generally leads to underutilization of link capacity. We recommend use of maximally fair rate allocations instead.

This report is organized as follows. Section 2 discusses the network scenarios under which the assumption of a discrete feasible set of rate allocations becomes a necessity. Section 3 describes the problem of maxmin fairness for multicast transmission and presents the mathematical framework used to model the problem. Section 4 develops the notion of fairness in the discrete case. Section 5 presents an algorithm for computation of the maximally fair rates. Section 6 identifies some directions for future research. This section concludes the paper.

2 Constraints on Signal Structure

We discuss the network scenarios under which the feasible set of rate allocations becomes discrete.

In multirate transmission, source transmits at a rate equal to the maximum of the rates allocated to its receivers. At forking points, video gateways may be used to transcode signal into a lower bit rate such that the rate in every link is equal to the maximum of the fair rates allocated to the session receivers downstream[30]. Output rates for video gateways can be fine tuned to match the required receiver rates, if rate adaptive videogateways are used[2]. However, transcoding places additional computational and administrative burden on the network. It may not be possible to know apriori which intermediate nodes need transcoding provisions, because this depends on fair rate allocations and fair rate allocations will change dynamically depending on the traffic conditions. So we may need to deploy transcoding gateways at all intermediate nodes which have a fanout greater than 1 (number of outgo-

ing links). This makes transcoding financially prohibitive. For transcoding, internal nodes must perform intensive computational tasks and may need to process arbitrary coding algorithms. This increases end to end delay. More importantly, transcoding can not be applied to a secure communication without entrusting the network with the encryption key. Depending on the security risks, this may be totally unacceptable. In these cases, network heterogenity can be countered through hierarchical encoding only.

We assume that hierarchical coding is used, i.e., the source encodes its signal into a number of layers that can be incrementally combined to provide progressive refinement in quality. Hierarchical coding was first suggested for packet voice transmission[7]. Subsequently, several hierarchical coding schemes have been proposed for video, e.g., [1], [16], [33], to name a few. Layer structure may be fixed and predetermined because of several reasons. Certain coding schemes are not amenable to dynamic layer bandwidths because of their inherent structure, e.g., perceptually weighted wavelets using hierarchical vector quantization with wavelet decomposition(WWHVQ)[35]. In certain video codes, substreams can be extracted to produce a specific range of resolutions only, e.g., 3D Subband Video Coding[33]. Certain coding schemes are particularly successful only when some apriori structure or hierarchy can be found in the problem[34], because certain computationally expensive optimizations, e.g., in quantizer thresholds, can be carried out offline. Some codecs, e.g., PVH codec of [23] are amenable to dynamic layer bandwidth adaptation, but the implementations still have a fixed layout strategy. In many cases, generating a layer requires a dedicated filter, and the number of filters employed by the source is fixed. Thus the number of layers is limited. Similarly, requirements for minimum packet size and limitations on packetization delay, may impose a lower bound on bandwidth assigned to a single layer and thus adversely affect layer granularity. Thus a perfect match between the layer bandwidths and desired receiver rates is not always feasible due to various reasons. The following approach is advocated in [29] in cases of fixed or partially adaptive signal hierarchy. Allocate to the receivers as many layers as permitted by the computed fair rate. If the total bandwidth consumed by the layers allocated to a receiver is strictly less than the fair computed rate, allocate one more layer to the receiver and let the network drop a certain portion of packets of the last layer at a forking point. For example, let u_1, u_2 be two receivers of a session with v_1 as the source in Figure 2. Let the fair rates for u_1 and u_2 be 2 and 1.5 units respectively. Let

each layer consume 1 unit of bandwidth. So both receivers will be allocated 2 layers. Intermediate node I should not replicate 50% of packets of layer 2. All packets of layer 2 are transmitted across e_2 and only 50% of layer 2 packets are transmitted across e_3. So 50% of layer 2 packets should not be replicated at node I. However, the idea of allocating an extra layer to a receiver, and letting the network drop a certain percentage of packets of the highest layer may not work always. This is because the network may not be well equipped to selectively replicate a certain percentage of packets of the highest layer or selectively drop a certain percentage of packets of the highest layer after replicating all the packets. Most of the current day routers follow random drop or drop tail policy, i.e., in event of congestion the routers drop any packet in the queue or the packet at the end of the queue, depending on the respective policy. So if the receivers oversubscribe if they have residual bandwidth (like u_2 in Figure 2 subscribing to 2 layers, when its fair share is 1.5 units), the network will be congested and the routers will indiscriminately drop packets of all the layers. An important characteristic of hierarchical encoding is that the layers are generally ordered in some manner. An enhancement layer yields useful information only when the base layer and lower enhancements layers have been successfully decoded. So loss of packets of all the layers will adversely affect signal quality. Its a good idea not to subscribe to the highest layer at all in this case and keep the network free of congestion. Besides, if the highest layer employs a differential coding scheme, then even a small percentage loss of packets may garble the entire information in the final layer. Another point to keep in mind is that generally the layers are coded so that the loss of entire layer may cause graceful degradation in quality. However loss of a certain proportion of the highest layer packets may produce perceptually annoying distortions in certain coding schemes. In all these cases, "partial" subscription to a layer is useless and receivers can only subscribe to layers fully. Besides, sometimes a source simply transmits each layer of its signal on a separate multicast group[24]. A receiver either subscribes to a group or it does not. It can not "partially" subscribe.

So we assume that the source encodes its signal into a number of layers that can be incrementally combined to provide progressive refinement in quality. The bandwidths consumed by the layers are predetermined. A receiver can not be allotted a layer "partially". As a consequence the possible rates of a receiver form a discrete set. Unlike in [29], *any* rate vector

which satisfies the capacity constraints and the minimum and maximum rate constraints can not be allocated. Only a discrete subset of these rate vectors satisfying the capacity constraints and the minimum and maximum rate constraints can be allocated. We need to make a fair allocation of rates in this discrete set. Equivalently, we can consider the problem of fair allocation of layers, where every layer has a predetermined bandwidth.

3 Network Model

We consider an arbitrary topology network with N multicast sessions. A multicast session is identified by the triplet (n, v, U), n is a unique number assigned to the session, v is the source node of the session and U is the group of intended destination nodes (n has been incorporated to distinguish between different sessions with the same source destination pair). We assume that the traffic from node v is transported across a predefined multicast tree to nodes in U. The tree can be established during connection establishment phase if the network is connection oriented or can be established by some well known multicast routing protocol like DVMRP[13], MOSPF[21], CBT[5], PIM[14] and MIP[25] in an internet type network. The receivers may have minimum number of layers constraints. Also, some sources may not be able to transmit more than a certain number of layers because of the predetermined hierarchical structure. Some receivers may have a low processing ability. In that case, it is useless to allocate higher number of layers to that session. So layer allocations can have maximum number of layer constraints as well. These parameters are useful for ATM like scenarios, where session establishment is preceded by a negotiation stage and the network can be informed of these requirements during the negotiation stage. In a connectionless network, sessions can not have any such requirement as the network would never know of these, and would have to make layer allocation irrespective of any such requirement. Such a scenario can very well be accomodated in our model, by assuming minimum layer requirement as 0 and maximum number to be ∞ for for each receiver.

We call every source destination pair of a session a virtual session. For example, if a session n has source v and destination set U, where $U = \{u_1, \ldots, u_t\}$, then this session would correspond to t virtual sessions, (n, v, u_1), $\ldots, (n, v, u_t)$. The network shown in figure 1 has a single session, $(1, v, U)$,

with $U = \{u_1, \ldots, u_4\}$. This session corresponds to 4 virtual sessions, $(1, v, u_1)$, $(1, v, u_2)$, $(1, v, u_3)$, $(1, v, u_4)$. Our objective would be to achieve a fair layer allocation for the virtual sessions. To ensure fairness in a multirate network, we need to consider fair layer allocation for the virtual sessions separately, instead of those for the overall sessions. We assume that every virtual session (source-destination pair) has a minimum and a maximum layer requirement. Absence of these requirements can be incoporated by choosing minimum layer requirement as 0 and maximum layer requirement as ∞.

Informally speaking a layer allocation for the virtual session is feasible, if the number of layers for every virtual session is between the minimum and the maximum possible number of layers for the virtual session. Besides if session n corresponds to virtual sessions $m_{n1}, \ldots, m_{nt}$ in link l, then the maximum of the bandwidths allocated to the virtual sessions $m_{n1}, \ldots, m_{nt}$ is the bandwidth consumed by session n in link l. Total bandwidth consumed by all sessions traversing through link l can not exceed the capacity of link l. Formal definition follows.

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

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

Equivalently, a layer allocation vector $\vec{\gamma}$ is feasible if the corresponding

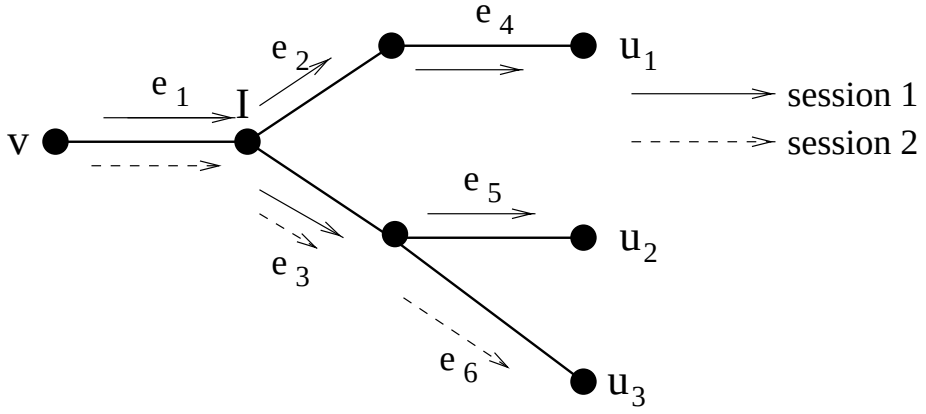

Figure 3:

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

1. r_i is a multiple of b, the layer bandwidth, for all virtual sessions i,
2. $\mu_i \leq r_i \leq \sigma_i \ \forall \ i$, where μ_i and σ_i are respectively the minimum and maximum rates of virtual session i, $\sigma_i \geq \mu_i \geq 0$, $\mu_i = b\iota_i$ and $\sigma_i = bp_i$,
3. Total bandwidths consumed by various sessions in a link should not exceed the link capacity.

$$\sum_{i \in n(l)} \lambda_{il} \leq C_l \qquad \text{(capacity condition)},$$

where λ_{il} denotes the rate allocated to session i on link l under rate allocation $\vec{r}$. It is the maximum of the rates allocated to the virtual sessions of session i traversing link l, i.e., $\lambda_{il} = \max_{j \in m(i,l)} r_j$.

Example 3.1: Refer to the network shown in Figure 3. There are 2 sessions, session 1, $(v, \{u_1, u_2\})$ and session 2, (v, u_3). There are 3 virtual sessions, (v, u_1), (v, u_2) and (v, u_3), named virtual sessions 1, 2, and 3 respectively. Session 1 corresponds to two virtual sessions, (v, u_1) and (v, u_2). Session 2 corresponds to one virtual session, (v, u_3). Every layer consumes 0.5 units of bandwidth. Virtual session (v, u_1) requires at least 8 layers, i.e., at least 4 units of bandwidth. It can not receive more than 10 layers. Its maximum

rate is 5 units. Virtual session (v, u_2) needs at least 2 layers and has no restriction on the maximum number of layers. It has a minimum rate of 1 unit and maximum rate of ∞ (no constraint on maximum rate). Virtual session (v, u_3) does not have a minimum number of layers requirement, i.e., can have a minimum rate of 0. It can receive at most 10 layers, i.e., has a maximum rate of 5 units. C_i denotes the capacity of edge e_i. $(C_1, \ldots, C_6) = (7, 4, 5, 4, 4, 6)$ units. $n(e_1) = n(e_3) = \{1, 2\}$, $n(e_2) = n(e_4) = n(e_5) = \{1\}$, $n(e_6) = \{2\}$. $m(1, e_1) = \{1, 2\}$, $m(2, e_1) = \{3\}$, $m(1, e_2) = \{1\}$, $m(1, e_3) = \{2\}$, $m(2, e_3) = \{3\}$, $m(1, e_4) = \{1\}$, $m(1, e_5) = \{2\}$, $m(2, e_6) = \{3\}$. A layer allocation vector $(\gamma_1, \gamma_2, \gamma_3)$ is feasible if $\gamma_1, \gamma_2, \gamma_3$ are integers,

$$\left.\begin{array}{lll} 8 \leq & \gamma_1 & \leq 10 \\ 2 \leq & \gamma_2 & \leq \infty \\ 0 \leq & \gamma_3 & \leq 10 \end{array}\right\} \text{Minimum and Maximum layer constraints} \qquad (1)$$

$$\left.\begin{array}{llllll} 0.5\left(\Gamma_{1e_1} + \Gamma_{2e_1}\right) & \leq & 7 & \Gamma_{1e_1} = \max(\gamma_1, \gamma_2) & \Gamma_{2e_1} = \gamma_3 & (\text{Link } e_1) \\ 0.5\Gamma_{1e_2} & \leq & 4 & \Gamma_{1e_2} = \gamma_1 & & (\text{Link } e_2) \\ 0.5\left(\Gamma_{1e_3} + \Gamma_{2e_3}\right) & \leq & 5 & \Gamma_{1e_3} = \gamma_2 & \Gamma_{2e_3} = \gamma_3 & (\text{Link } e_3) \\ 0.5\Gamma_{1e_4} & \leq & 4 & \Gamma_{1e_4} = \gamma_1 & & (\text{Link } e_4) \\ 0.5\Gamma_{1e_5} & \leq & 4 & \Gamma_{1e_5} = \gamma_2 & & (\text{Link } e_5) \\ 0.5\Gamma_{2e_6} & \leq & 6 & \Gamma_{2e_6} = \gamma_3 & & (\text{Link } e_6) \end{array}\right\} \text{Capacity constraints} \qquad (2)$$

If $\vec{\gamma}$ is a layer allocation vector, the corresponding rate allocation vector $\vec{r}^{\gamma}$ is defined as $r_i^{\gamma} = 0.5\gamma_i$.

A rate vector (r_1, r_2, r_3) is feasible if r_1, r_2 and r_3 are multiples of 0.5,

$$\left.\begin{array}{lll} 4 \leq & r_1 & \leq 5 \\ 1 \leq & r_2 & \leq \infty \\ 0 \leq & r_3 & \leq 5 \end{array}\right\} \text{Minimum and Maximum rate constraints}$$

$$\left.\begin{array}{llllll} \lambda_{1e_1} + \lambda_{2e_1} & \leq & 7 & \lambda_{1e_1} = \max(r_1, r_2) & \lambda_{2e_1} = r_3 & (\text{Link } e_1) \\ \lambda_{1e_2} & \leq & 4 & \lambda_{1e_2} = r_1 & & (\text{Link } e_2) \\ \lambda_{1e_3} + \lambda_{2e_3} & \leq & 5 & \lambda_{1e_3} = r_2 & \lambda_{2e_3} = r_3 & (\text{Link } e_3) \\ \lambda_{1e_4} & \leq & 4 & \lambda_{1e_4} = r_1 & & (\text{Link } e_4) \\ \lambda_{1e_5} & \leq & 4 & \lambda_{1e_5} = r_2 & & (\text{Link } e_5) \\ \lambda_{2e_6} & \leq & 6 & \lambda_{2e_6} = r_3 & & (\text{Link } e_6) \end{array}\right\} \text{Capacity constraints}$$

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

We would like to comment on our assumption that all multicast packets of the same session move along the same tree. Different multicast layers are perceived as different multicast groups and trees for different groups may be completely different in general. However, trees for different multicast layers of the same session will remain the same if source rooted trees are used, as all these layers (multicast groups) have the same source. Besides, trees for different multicast layers of the same session should not differ very much, as that would complicate reconstruction of information at the receiver. For example, if we consider video transmission, and different layer packets of the same session traverse along different multicast trees, then different layer packets for the same frame may arrive at a receiver at different times, and frame reconstruction will involve a lot of packet reordering. This may incur an unacceptable delay jitter. Thus it may be a good idea to use source rooted trees in this case. Many video coders make the same assumption, e.g., [8].

4 Fairness in Discrete Feasible set

To the best of our knowledge, fairness in a discrete feasible set have not been studied before. As it turns out, things become significantly different, when the feasible set is discrete. First we describe various useful notions of fairness in a discrete feasible set. Informally, a feasible layer allocation vector is maxmin fair if it is not possible to maintain feasibility and increase the number of layers of a virtual session without decreasing that of any other virtual session which has equal or lower number of layers. More formally, a feasible layer allocation vector $\vec{\gamma}^1$ is maxmin fair if it satisfies the following property with respect to any other feasible layer allocation vector $\vec{\gamma}^2$: if there exists i such that the ith component of $\vec{\gamma}^2$, γ_i^2 is strictly greater than that of

$\vec{\gamma}^1$, γ_i^1 ($\gamma_i^1 < \gamma_i^2$), then there exists j such that the jth component of $\vec{\gamma}^1$, γ_j^1 is less than or equal to the ith component of $\vec{\gamma}^1$, γ_i^1 ($\gamma_j^1 \leq \gamma_i^1$) and the jth component of $\vec{\gamma}^2$ (γ_j^2) is strictly less than the jth component of $\vec{\gamma}^1$ ($\gamma_j^2 < \gamma_j^1$). The components of $\vec{\gamma}^2$ are more unequal than those of $\vec{\gamma}^1$ in some sense.

Example 4.1: The maxmin fair layer allocation vector in the network of Example 4.1 is $(8, 5, 5)$. It is easy to check that this layer allocation vector is feasible. It is not possible to increase γ_1 above 8 because of the capacity constraint of link e_2. Any increase in γ_2 (γ_3) will cause a decrease in γ_3 (γ_2) because of the capacity constraint of link e_3 and $\gamma_2 = \gamma_3$ in the allocation $(8, 5, 5)$. Thus $(8, 5, 5)$ is the maxmin fair layer allocation vector.

Given a $M-$ dimensional vector $\vec{V}$, define its lexicographically ordered version $\hat{V}$ as follows: $\hat{v}_j = v_k$ for some k and $\hat{v}_1 \leq \hat{v}_2 \leq \ldots \hat{v}_M$. In other words, components of $\vec{V}$ are an ordered version of those of $\hat{V}$. A layer allocation vector $\vec{\gamma}^1$ is lexicographically greater than a layer allocation vector $\vec{\gamma}^2$ if there exists i such that $\hat{\gamma}_i^1 > \hat{\gamma}_i^2$ and $\hat{\gamma}_j^1 = \hat{\gamma}_j^2$ if $j \leq i$. Layer allocations $\vec{\gamma}^1$ and $\vec{\gamma}^2$ are lexicographically equal if $\hat{\gamma}^1 = \hat{\gamma}^2$. Layer allocation $\vec{\gamma}^1$ is lexicographically less than $\vec{\gamma}^2$, if $\vec{\gamma}^2$ is lexicographically greater than $\vec{\gamma}^1$. A layer allocation vector $\vec{\gamma}$ is lexicographically optimal if it is feasible and if every feasible layer allocation vector is lexicographically less than or equal to $\vec{\gamma}$. Informally, a layer allocation is lexicographically optimal, if its smallest component is the largest amongst the smallest components of all feasible layer allocations, subject to this it has the largest second smallest component and so on.

Example 4.2: A lexicographically optimal layer allocation vector in the network of Example 4.1 is $(8, 5, 5)$. The minimum component of every layer allocation vector must be less than or equal to 5, because of the capacity constraint of link e_3. If the minimum component is equal to 5, then the second minimum must also be equal to 5, because of the capacity constraint of link e_3. Capacity constraint of link e_2 forces the largest component to be 8.

Maxmin fairness and lexicographic optimality are not entirely equivalent, though they have been used interchangeably in many places. Certain references have defined maxmin fair vector as the lexicographic optimal one,

e.g.[10]. Our definition of maxmin fairness have been suggested in [6]. Going by this definition, maxmin fairness is stronger than lexicographic optimality. In general, if we consider finite dimensional vectors, with the feasibility set closed and bounded, a lexicographically optimal vector always exists, but a maxmin fair vector may not exist. However as Lemma 4 proves, if a maxmin fair vector exists, it is lexicographically optimal. For example, a maxmin fair layer(rate) allocation vector may not exist as we would discuss later. However if the feasibility set were continuous, as in [29], maxmin fair rate allocation vector always exists and is lexicographically optimal. In this case the definitions of maxmin fairness and lexicographic optimality may be used interchangeably. However, in our case, we need to distinguish between the two.

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

Example 4.3: A network has a single link and two sessions traversing through the link. Every layer consumes 1 unit of bandwith. The capacity of the link is 1 unit. So possible allocation vectors are $(0, 1)$, $(1, 0)$, $(0, 0)$. None of these vectors are maxmin fair. For each of these vectors, it is possible to maintain feasiblity and increase the number of layers of one session, possibly hurting another session, but the other session has higher number of layers. For example, consider the allocation $(0, 1)$. It is possible to increase the number of layers of session 1, by hurting session 2. However session 2 has higher number of layers than session 1 in this allocation. Both $(1, 0)$ and $(0, 1)$ are lexicographically optimal.

In view of the nonexistence of maxmin fair layer allocation, lexicographically optimal layer allocation is the best one can hope for in this case. However as NP-hardness lemma shows, computation of lexicographically optimal layer allocation is an NP-hard problem in this case.

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

We prove this lemma in the appendix. The idea behind the proof is that, given any arbitrary graph G, we can construct a network such that the

computation of a lexicographically optimal layer allocation in the network has the same complexity as the computation of the largest independent set in the graph G. It is well known that the latter is an instance of an $NP-$hard problem.

In view of this unfortunate result, we can consider a different notion of fairness, *maximal fairness.* We use the concept of *relative fairness* introduced in [28]. A layer allocation vector $\vec{\gamma}^1$ is *fairer* than another layer allocation vector $\vec{\gamma}^2$, if for every virtual session i which has higher number of layers under $\vec{\gamma}^2$ than under $\vec{\gamma}^1$, there is some other virtual session j whose number of layers was already no more than that of i under $\vec{\gamma}^1$, and has been decreased further by $\vec{\gamma}^2$. A more formal definition of relative fairness follows.
A layer allocation vector $\vec{\gamma}^1$ is *fairer* than another layer allocation vector $\vec{\gamma}^2$ if

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

A layer allocation $\vec{\gamma}$ is maximally fair if it is feasible and if no other feasible layer allocation is fairer than $\vec{\gamma}$. There can be more than one maximally fair layer allocation vectors. One or more of these maximally fair layer allocation vectors are lexicographically optimal (proved in Lemma 5 in appendix). In the network of Example 4.3, both the layer allocation vectors $(1, 0)$ and $(0, 1)$ are maximally fair and lexicographically optimal.

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

As we show later, maximally fair layer allocation exists and can be computed in polynomial complexity. One objective could be to allocate layers as per some maximally fair layer allocation. But, first we investigate, whether maximal fairness is a good notion of fairness. To this effect, we introduce the concept of pseudo-bottleneck links. This is analogous to the concept of bottleneck links for a continuous feasible set[29]. Consider a layer allocation vector $\vec{\gamma}$. A link l is said to be *pseudo-bottlenecked* with respect to a virtual

session k traversing l for layer allocation vector $\vec{\gamma}$, if the following conditions are met:

1. Capacity of the link is *almost* fully utilized, i.e., the difference between the capacity of the link and the sum of the rates allocated to the sessions traversing the link must be less than b, i.e., $b\sum_{i\in n(l)}\Gamma_{il} > C_l - b$.

2. The virtual session has the maximum number of layers amongst all virtual sessions of the same session traversing the link, i.e., $\gamma_k = \Gamma_{\chi(k)l}$, where $\chi(k)$ is the session of virtual session k.

3. If the number of layers assigned to any other virtual session j traversing link l is higher than that of virtual session k by two or more layers, then the number of layers assigned to virtual session j is less than or equal to the minimum number of layers required for some virtual session in $m(\chi(j), l)$ (set of virtual sessions traversing link l and belonging to the same session as virtual session j). Formalizing this, if $\gamma_j > \iota_{\chi(j)l}$, where $\iota_{il} = \max_{j\in m(i,l)}\iota_j$, then $\gamma_j \leq \gamma_k + 1$. Informally speaking had there been no minimum number of layers requirement, the number of layers assigned to any other virtual session j traversing link l would not exceed that of virtual session k by more than one.

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

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

Example 4.4: Consider the network of Example 3.1. Let every layer consume 1 unit of bandwidth each ($b = 1$). Ignore the maximum rate constraints. The rest of the constraints remain the same. Virtual sessions (v, u_1), (v, u_2) and

(v, u_3) are named virtual sessions $1, 2, 3$ respectively. Virtual sessions 1, 2 belong to session 1 and virtual session 3 belongs to session 2. Consider the layer allocation $(4, 2, 3)$. Link e_1 is pseudo-bottlenecked w.r.t. virtual sessions $1, 3$ and link e_3 is pseudo-bottlenecked w.r.t. virtual session 2. Now consider the layer allocation vector $(4, 4, 1)$. Virtual session 3 does not have a pseudo-bottleneck link. This is because it traverses through links e_1, e_3 and e_6. Virtual session 2 traversing through link e_3 has 3 more layers than virtual session 3 and does not have a minimum number of layers requirement. This violates pseudo-bottleneck condition (3). Total bandwidth consumed by the sessions traversing through link e_1 is 5 units in all but the capacity of the link is 7 units, 2 units more than the capacity utilized and $b = 1$. Similarly, utilized capacity of link e_6 is 1 unit, while actual capacity of e_6 is 6 units. This violates pseudo-bottleneck condition (1). However, link $e_2(e_3)$ is pseudo-bottlenecked w.r.t. virtual session 1(2.)

Lemma 2 (Pseudo-bottleneck lemma) *A feasible layer allocation vector is maximally fair iff every virtual session has a pseudo-bottleneck link.*

We prove this lemma in the appendix. This lemma shows that maximal fairness is a good notion of fairness. Note that by this lemma if a layer allocation vector is maximally fair, then the number of layers allocated to a virtual session s can be increased only at the expense of one or more virtual sessions which have at most one layer more than s. From the definition of a pseudo-bottleneck link, if there is no minimum number of layers requirement for any virtual session, every virtual session will be assigned at least $\lceil\left(\frac{C_l}{|n(l)|b} - (1 - |n(l)|^{-1})\right)\rceil$ layers for some link l on its path. This indicates that every virtual session is guaranted a bandwidth close to the fair share of capacity for at least one link in its path. For large C_l, a virtual session gets at least $\lceil\frac{C_l}{|n(l)|b}\rceil$ (approximately) layers for some link l in its path. In general, networks have large capacities. Thus maximally fair allocation is fair to every virtual session in some sense. In presence of minimum rate requirements, every virtual session s is guaranted $\lceil\left(\frac{C_l - \sum_{i\in\tau(l)}\mu_{il}}{|n(l)\setminus\tau(l)|b} - (1 - |n(l)\setminus\tau(l)|^{-1})\right)\rceil$ layers, for some link l on its path, where $\tau(l)$ is the set of sessions traversing link l with session rate on link l exceeding the rate of virtual session s ($\tau(l) \subset n(l)$). Intuitively, this means that virtual session s receives an almost fair share of the residual link l bandwidth, after distributing the minimum

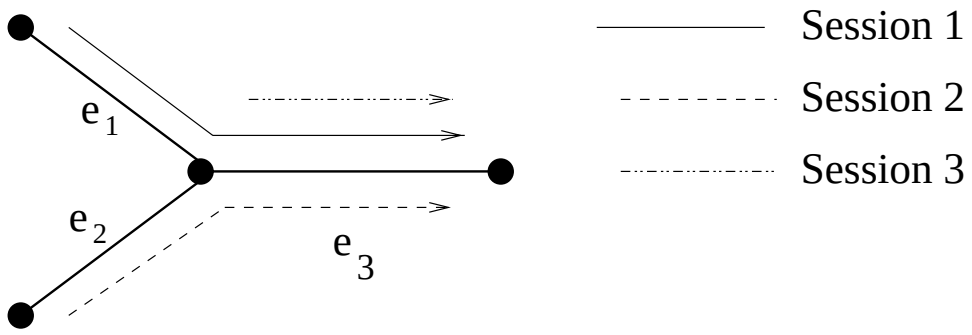

Figure 4:

rates to other sessions. This lemma also serves as a test for maximal fairness of a feasible layer allocation vector. We will use this lemma in proving the correctness of an algorithm for computation of a maximally fair layer allocation vector. There exists a similar result for maxmin fairness in a continuous feasible set which says that a rate allocation is maxmin fair if and only if every virtual session has a bottleneck link. The definitions of bottleneck links are similar in both cases.

Maximal fairness has other nice properties. As Lemma 5 shows, a lexicographically optimal layer allocation is maximally fair. We prove this lemma in the appendix. Also by definition, a maxmin fair layer allocation, (if one exists), is fairer than all other feasible layer allocations. Thus if a maxmin fair layer allocation exists, it is the only maximally fair layer allocation (This is true for any vector and any reference feasible set, not just layer allocations). So any algorithm for computation of a maximally fair layer allocation will yield a maxmin fair layer allocation, if one exists. This is interesting, in view of the observation that even if a maxmin fair rate allocation exists in our discrete feasible set, it may be different from the maxmin fair rate allocation in the continuous feasible set (i.e., feasible set with only the restrictions that rate allocations must satisfy capacity and the minimum rate constraints). Consider the following example.

Example 4.5: Consider a network with 3 links and 3 unicast sessions (unicast is a special case of multicast) shown in Figure 4. The links are $\{e_1, e_2, e_3\}$. The capacity of link e_i is 1.9 units, $i \in \{1, 2\}$ and 6 units if $i = 3$. The sessions

are $\{1, 2, 3\}$. Sessions and virtual sessions are the same since all sessions are unicast. Session 1 traverses through e_1, e_3, session 2 traverses through e_2, e_3, and session 3 traverses through e_3. Each session requires at least 0 layer, i.e., there is no minimum layer requirement. Every layer consumes 1 unit of bandwidth, i.e., $b = 1$. If we had allowed allocation of any rates, subject to the capacity and the minimum rate constraints, like in [29], the maxmin fair rate allocation would be $(1.9, 1.9, 2.2)$. Now let it be possible only to allocate layers (as we have assumed in this paper). Let every layer consume 1 unit of bandwidth. Maxmin fair layer and rate allocations exist in this example and are both equal to $(1, 1, 4)$.

This difference is because of the difference in the feasible set of rate vectors in the two cases. The feasible set of rate vectors for discrete bandwidth layers is a proper subset of the continuous feasible set of [29]. 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 the algorithm presented in [29] 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 it exists. Thus we have strong incentive to compute the maximally fair layer allocation. In the next section, we present a polynomial complexity algorithm for computation of a maximally fair layer allocation.

5 Algorithm for Computation of Maximally fair Layer Allocation

An algorithm that would obtain a maximally fair layer allocation is not entirely obvious. For example, let every layer consume b units of bandwidth and r_s^c be the maxmin fair bandwidth allocated to virtual session s if we can allocate any rate subject to the capacity and the minimum rate constraints. Maxmin fair rates can be computed using an algorithm presented in [29] in this case. It is tempting to think that a maximally fair rate allocation will allocate γ_s layers to the sth virtual session where $\gamma_s \in \{\lfloor \frac{r_s^c}{b} \rfloor, \lceil \frac{r_s^c}{b} \rceil\}$. The following counterexample shows that it is not always so.

Example 5.1: Consider the network of Example 4.5, shown in Figure 4.

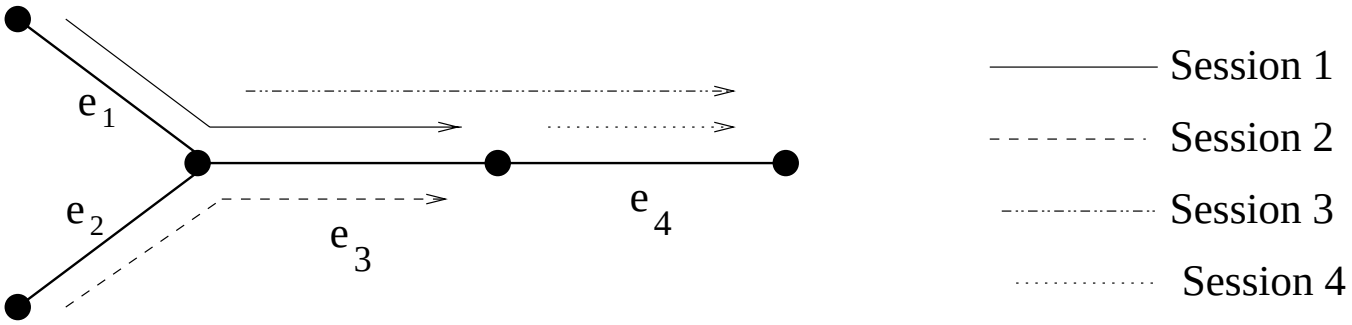

Figure 5:

If we had allowed allocation of any rates, subject to capacity and minimum rate constraints, like in [29], the maxmin fair rate allocation would be $(1.9, 1.9, 2.2)$. Now let it be possible only to allocate layers (as we have assumed in this paper). Let every layer consume 1 unit of bandwidth. Consider the layer allocation scheme which allocates γ_i layers to session i, where $\gamma_i = \lceil r_i^c \rceil$ or $\gamma_i = \lfloor r_i^c \rfloor$ subject to feasibility. Since $r_i^c \leq 1.9$, for $i < 3$, $\gamma_i = 1$ $i < 3$ and either $\gamma_3 = \lfloor r_3^c \rfloor = 2$ or $\gamma_3 = \lceil r_3^c \rceil = 3$. However $(1, 1, 4)$ is a feasible layer allocation and is fairer than both $(1, 1, 2)$ and $(1, 1, 3)$. Maxmin fair layer allocation exists in this example and is equal to $(1, 1, 4)$.

So the intuitive algorithm does not yield a maximally fair layer allocation. An intuitively appealing way to fix this flaw is as follows: obtain the maxmin fair rates assuming the feasible set to be continuous (as [29] assumes). Initially allocate $\lfloor \frac{r_i^c}{b} \rfloor$ layers where r_i^c is the fair rate of virtual session i to virtual session i, $\forall i$. Now try to add a layer to virtual session $1, \ldots, M$, if there are M virtual sessions in some predetermined order (could be increasing order of the layers allocated) without decreasing the number of layers allocated to any virtual session any time. Repeat this process as long as the number of layers allocated to a virtual session i can not be increased without decreasing that allocated to some other virtual session j, for all i. It appears that this algorithm will always yield a maximally fair layer allocation. However that is not quite true. Consider the following counter example.

Example 5.2: Consider a modified version of the network of Example 4.5, shown in Figure 5. There is one additional edge, e_4, with capacity 8 units.

There is also one additional session, session 4. Session 4 traverses through e_4 only. Session 3 traverses through e_3 and e_4. The rest remains the same. The maxmin fair rate allocation, assuming that the feasible set is continuous is $(1.9, 1.9, 2.2, 5.8)$. Let every layer consume 1 unit of bandwidth. The layer allocation algorithm suggested above would either yield a layer allocation vector $(1, 1, 3, 5)$ or $(1, 1, 2, 6)$ depending on the order in which sessions are chosen for allocation of additional layers. However $(1, 1, 4, 4)$ is a feasible layer allocation vector which is fairer than both $(1, 1, 3, 5)$ and $(1, 1, 2, 6)$. In fact maxmin fair layer allocation exists and is equal to $(1, 1, 4, 4)$ in this case. Thus neither $(1, 1, 3, 5)$ nor $(1, 1, 2, 6)$ is a maximally fair layer allocation vector.

The counter examples indicate that it is possible that all maximally fair vectors have $\gamma_i < \lfloor \frac{r_i^c}{b} \rfloor$ or $\gamma_i > \lceil \frac{r_i^c}{b} \rceil$ for some virtual session i. We would like to mention that if the maxmin fair rate allocations are computed with capacity C_l of link l replaced by $b\lfloor \frac{C_l}{b} \rfloor$, then the algorithm mentioned above may yield a maximally fair layer allocation if one tries to add a layer to the virtual sessions in some particular order. But there is no obvious way to determine this order, particularly for multicast networks. For example, 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. Consider the following counter example.

Example 5.3: Consider the network shown in Figure 6. There are 7 edges, $e_1, \ldots, e_7$ and 6 sessions, sessions $1, \ldots, 6$. Session 1 and 2 are multicast and the rest are unicast. Session 1 (source: v_1, destinations: u_1, u_2) traverses edges e_1, e_2, e_3, and consists of two virtual sessions, virtual sessions 1 and 2. Virtual session 1 (source: v_1, destination: u_1) traverses edges e_1, e_2. Virtual session 2 (source: v_1, destination: u_2) traverses edges e_1, e_3. Session 2 (source: v_1, destinations: u_3, u_4) traverses edges e_1, e_4, e_5 and consists of two virtual sessions, virtual sessions 3 and 4. Virtual session 3 (source: v_1, destination: u_3) traverses edges e_1, e_4. Virtual session 4 (source: v_1, destination: u_4) traverses e_1, e_5. Sessions $3, 4, 5, 6$, traverse 1 edge each, e_3, e_4, e_6, e_7 respectively and consist of 1 virtual session each, virtual sessions $5, 6, 7, 8$ respectively, traversing the same edges as the respective sessions. Sessions 7 (source: v_2, destination: u_5), and 8 (source: v_2, destination: u_6), traverse 2 edges each, e_3, e_6 and e_4, e_7 respectively and consist of 1 virtual session each,

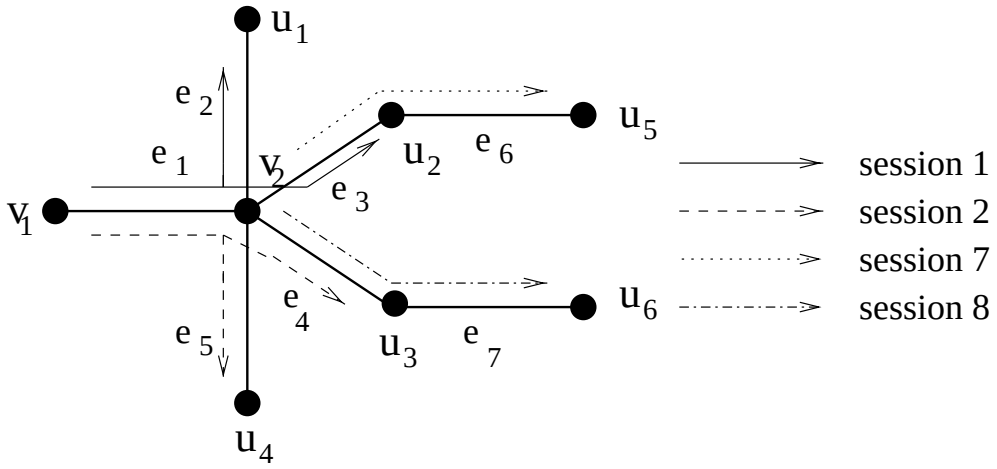

Figure 6: Sessions $1, 2, 7, 8$ have been shown in the figure. Sessions $3, 4, 5, 6$ span one link each, e_3, e_4, e_6, e_7 respectively.

virtual sessions $9, 10$ respectively, traversing the same edges as the respective sessions.

$$C_l = \begin{cases} 5 & l = e_1 \\ 3 & l \in \{e_2, e_5\} \\ 8 & l \in \{e_3, e_4\} \\ 1 & l \in \{e_6, e_7\}. \end{cases}$$

Every layer consumes 1 unit of bandwidth ($b = 1$). Note that $C_l = b\lfloor \frac{C_l}{b} \rfloor$ for every link l. The maxmin fair rate allocation assuming the feasible set to be continuous (i.e., without the restriction that virtual session rates are integers in this case) is given by $\vec{r}$, where

$$r_i = \begin{cases} 2.5 & i \in \{1, \ldots, 4\} \\ 5 & i \in \{5, 6\} \\ 0.5 & \text{otherwise.} \end{cases}$$

Going by this algorithm, we initially allocate γ_i layers to virtual session i, where

$$\gamma_i = \begin{cases} 2 & i \in \{1, \ldots, 4\} \\ 5 & i \in \{5, 6\} \\ 0 & \text{otherwise.} \end{cases}$$

Virtual sessions $7, \ldots, 10$ are allocated the minimum number of layers. We can increment the layers of either virtual sessions $7, 8$ or $7, 10$, or $8, 9$ or

$9, 10$, but not those of any bigger combination amongst $7, \ldots, 10$, because of feasibility. Let us select virtual sessions $9, 10$ arbitrarily amongst those assigned the minimum number of layers. Now $(3, 2, 2, 2, 5, 5, 0, 0, 1, 1)$ or $(2, 2, 2, 3, 5, 5, 0, 0, 1, 1)$ are the possible output layer allocations depending on whether we choose virtual session 1 or 4 for further incrementation at the next stage. Neither is maximally fair. A layer allocation $(3, 3, 2, 2, 4, 5, 0, 0, 1, 1)$ is feasible and fairer than $(3, 2, 2, 2, 5, 5, 0, 0, 1, 1)$. A layer allocation $(2, 2, 3, 3, 5, 4, 0, 0, 1, 1)$ is feasible and fairer than $(2, 2, 2, 3, 5, 5, 0, 0, 1, 1)$. Thus incrementation in increasing order of layer allocation may lead to layer allocations which are not maximally fair.

If we had incremented the number of layers of virtual sessions $7, 8$ instead of $9, 10$, in Example 5.3, then we would have either $(3, 3, 2, 2, 5, 5, 1, 1, 0, 0)$ or $(2, 2, 3, 3, 5, 5, 1, 1, 0, 0)$, both maximally fair layer allocations, but it is not easy to know ahead the right order. Trying all possible orders will yield an exponential complexity algorithm in the worst case. However there exists a polynomial complexity algorithm for computation of a maximally fair layer allocation.

Now we present an algorithm for computation of a maximally fair layer allocation vector. $n(l)$, $m(i, l)$, $\chi(s)$ and μ_{il} are as defined in pages 10, 10, 17 and 17 respectively. We introduce some additional terminologies below.

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

A virtual session is *saturated* under a rate vector if it traverses a link in which the session rate is equal to its bandwidth and if the difference between the bandwidth consumed in the link and the capacity of the link is less than b units.

A session is *saturated* on a link l if all the virtual sessions of the session traversing the link l are saturated.

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

link, if there were no bandwith constraints on other links, and feasible rate allocations need only satisfy the capacity constraints.

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

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

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

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

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

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

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

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

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

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

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

The algorithm follows.

1. $k = 0$ $\eta_l(0) = 0$, $F_l(0) = 0$, $S(0) = \{1, \ldots, M\}$, $\Xi_l(0) = n(l)$ $\forall$ link l, $r_j(0) = \mu_j$, $\forall j \in S(0)$, $\lambda_{il}(0) = \max_{j \in m(i,l)} r_j(0)$.

2. $k \rightarrow k + 1$

3. For every link l in the network compute the link control parameter. If $\Xi_l(k-1) \neq \phi$, then $\eta_l(k)$ is the maximum possible θ, which satisfies the equation, $F_l(k-1)+\sum_{i \in \Xi_l(k-1)} \max(\theta, \lambda_{il}(k-1)) = C_l$ else $\eta_l(k) = \eta_l(k-1)$. For all sessions i passing through link l, $\eta_{il}(k) = \max(\eta_l(k), \lambda_{il}(k-1))$.

4. Compute $\omega_s(k)$ for all virtual sessions $s \in S$, where $\omega_s(k) = b\lfloor \frac{\min_{l \in L_s} \eta_{\chi(s)l}(k)}{b} \rfloor$, if $s \in S(k-1)$, else $\omega_s(k) = r_s(k-1)$.

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

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

7. If $\Lambda(k) \neq \phi$, compute the rates allocated to the virtual sessions after the kth iteration, via, $r_s(k) = \omega_s(k)$, $\forall s$ and go to step (9).

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

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

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

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

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

10. Compute the set of virtual sessions unsaturated after the kth iteration, $S(k) = S(k-1) \setminus \{s : \exists\ l \in L_s, \text{ s.t. } \sum_{i \in n(l)} \lambda_{il}(k) > C_l - b \text{ and } r_s(k) = \lambda_{\chi(s),l}(k)\}$.

11. If $S(k) = \phi$, i.e., all virtual sessions are saturated, compute the layer allocation vector $\vec{\gamma}$ via $\gamma_j = \frac{r_j(k)}{b}$ and the algorithm terminates, else go to the next step.

12. For every link l, compute the set of unsaturated sessions passing through link l at the end of the kth iteration: $\Xi_l(k) = \{n : n \in \{1, \ldots, N\}, m(n,l) \cap S(k) \neq \phi\}$ (N is the number of sessions).

13. For every link l, for which $\Xi_l(k) \neq \phi$, compute the bandwidth consumed by the saturated sessions passing through link l, $F_l(k) = \sum_{i \in n(l) \setminus \Xi_l(k)} \lambda_{il}(k)$.

14. Go to step (2).

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

Initially all the link control parameters are assigned zero values. The rates of the virtual sessions are initialized to the respective minimum rates. All sessions and virtual sessions are unsaturated. Next the algorithm computes the link control parameters as per step (3). If there are no minimum rate requirements, the link control parameter for link l at the first iteration

is the capacity of the link per session traversing the link. The next step is to compute the session link control parameters for every session at every link. Session link control parameter for a session traversing a link is the maximum of the link control parameter and the previous iteration session link rate. If a virtual session traversing link l, had no bandwidth constraint on other links, then it is assigned a rate equal to the greatest multiple of b not exceeding its session link control parameter. On account of the bandwidth constraint in other links, the virtual session gets a rate equal to the greatest multiple of b not exceeding the minimum of its session link control parameters on its path. This ensures that virtual session rates are multiples of b. If no virtual session is saturated, $(\Lambda(k) = \phi)$, then try to find a virtual session s which satisfies the properties mentioned in step (8). At least one such virtual session exists in this case(Lemma 14). Increment the rate of such a virtual session by b. This is done because otherwise, the algorithm can continue forever. This is because in the next iteration, the same link control parameter will be computed and the process repeats again and again. A session is saturated if all its virtual sessions are saturated. The bandwidth consumed by the saturated sessions, if any, are computed. This bandwidth is subtracted from the link capacity, and the link control parameters are recomputed at the beginning of every iteration as per step (3) and the process continues. A new iteration starts if there are still unsaturated virtual sessions. It turns out that either the rate of at least one virtual session increases by b units or the number of unsaturated virtual session decreases by at least one, at the end of every iteration (proof of Theorem 2). Neither of these two can continue indefinitely. So the algorithm terminates in finite number of iterations. ($|\mathcal{L}|M$ iterations, $\mathcal{L}$ is the set of all links, M is the number of virtual sessions). Upon termination we have a maximally fair rate vector and the corresponding layer allocation is maximally fair as well. An example illustrating the operation of the algorithm follows .

Example 5.4: Consider the network of Example 4.4. $L_1 = \{e_1, e_2, e_4\}$, $L_2 = \{e_1, e_3, e_5\}$, $L_3 = \{e_1, e_3, e_6\}$. Link control parameters are as follows. $\eta_{e_1}(1) = 3$, $\eta_{e_2}(1) = 4$, $\eta_{e_3}(1) = 2.5$, $\eta_{e_4}(1) = 4$, $\eta_{e_5}(1) = 4$, $\eta_{e_6}(1) = 6$. Now, the session link control parameters are as follows. $\eta_{1e_1}(1) = 4$, $\eta_{2e_1}(1) = 3$, $\eta_{1e_2}(1) = 4$, $\eta_{1e_3}(1) = 2.5$, $\eta_{2e_3}(1) = 2.5$, $\eta_{1e_4}(1) = 4$, $\eta_{1e_5}(1) = 4$, $\eta_{2e_6}(1) = 6$. Computing the $\omega_s(1)$s as per step 4, we have $\omega_1(1) = 4$, $\omega_2(1) = 2$, $\omega_3(1) = 2$. Observe that virtual session 1 is saturated w.r.t. $\vec{\omega}(1)$. So $\vec{r}(1) = \vec{\omega}(1)$. Vir-

tual session 2 and 3 are not saturated w.r.t. $\vec{r}(1)$. $S(1) = \{2, 3\}$. $\Xi_{e_1}(1) = \Xi_{e_3}(1) = \{1, 2\}$, $\Xi_{e_5}(1) = \{1\}$, $\Xi_{e_6}(1) = \{2\}$, $\Xi_l(1) = \phi$, if $l \in \{e_2, e_4\}$. If $l \in \{e_2, e_4\}$, $F_{e_l}(1) = 4$, and $F_{e_l}(1) = 0$ otherwise. Computations for the next iteration are as follows. $\eta_{e_1}(2) = 3$, $\eta_{e_3}(2) = 2.5$, $\eta_{e_5}(2) = 4$, $\eta_{e_6}(2) = 6$. $\eta_l(2) = \eta_l(1)$ for the rest of the links. The session link control parameters are as follows. $\eta_{1e_1}(2) = 4$, $\eta_{2e_1}(2) = 3$, $\eta_{1e_2}(2) = 4$, $\eta_{1e_3}(2) = 2.5$, $\eta_{2e_3}(2) = 2.5$, $\eta_{1e_4}(2) = 4$, $\eta_{1e_5}(2) = 4$, $\eta_{2e_6}(2) = 6$. $\omega_2(2) = \omega_3(2) = 2$. $\omega_1(1) = r_1(1) = 4$. No new virtual session is saturated w.r.t. $\vec{\omega}(2)$. Both virtual sessions 2 and 3 satisfy the conditions for incrementation in step (8). We choose virtual session 2 for incrementation arbitrarily. It follows that $r_1(2) = 4, r_2(2) = 3, r_3(2) = 2$. This saturates both virtual sessions 2 and 3. All virtual sessions are saturated and the algorithm terminates.

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

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

1. *maximally fair*

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

We prove this theorem formally in the appendix. We give the gist of the proof here. We first show that the rate allocation at the end of every iteration is feasible. We show that if a virtual session saturates in an iteration, then it has a pseudo-bottleneck link in all subsequent iterations. Maximal fairness of the final rate allocation vector and hence the output layer allocation vector follows from the pseudo-bottleneck lemma (Lemma 2) since the algorithm terminates only when all virtual sessions saturate. The last part of the theorem follows from the observation made in Section 4 that any algorithm which outputs a maximally fair layer allocation, outputs a maxmin fair layer allocation, if one such exists. The following theorem proves that the algorithm terminates in a finite number of iterations.

Theorem 2 (Finite-Termination Theorem) *The algorithm terminates in at most* $M + |\mathcal{L}|M$ *number of iterations, where* $\mathcal{L}$ *is the set of links and* M *is the number of virtual sessions.*

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

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

The source can transmit each layer on a separate multicast group. Once the fair layer allocation is computed, the receivers should subscribe to the appropriate multicast groups. If no receiver subscribes to a particular layer, then the source should not transmit it.

6 Conclusion and Discussion

We would like to point out that the choice of an unsaturated virtual session s for possible incrementaion of rate is crucial for attaining maximal fairness in the end. Step (8) of the algorithm requires this virtual session to be any unsaturated virtual session s for which $\omega_s(k) < \min_{l \in L_s} \eta_l(k)$ and $\Omega_{il}(k) \geq b\lfloor \frac{\eta_l(k)}{b} \rfloor$, $\forall\ i \in \Xi_l(k-1)$, if $l \in L_s$ and $\lfloor \frac{\eta_l(k)}{b} \rfloor = \min_{l \in l_s} \lfloor \frac{\eta_l(k)}{b} \rfloor$. However, if the virtual session s is chosen completely adhoc, i.e., even if it does not satisfy the criteria mentioned above, then the output layer allocation may not be maximally fair. For example, choosing any unsaturated virtual session s, for which $\omega_s(k) < \min_{l \in L_s} \eta_l(k)$, may not attain a maximally fair layer allocation in the end. Consider the following example.

Example 6.1: Consider the network shown in Figure 7. It has 3 links, e_1, e_2, e_3 and 4 sessions, sessions $1, 2, 3, 4$. All sessions are unicast and do not have any minimum number of layers requirement. Sessions and virtual sessions are the same in this case. Session 1 traverses though link e_1, session 2 traverses through links e_1 and e_2, session 3 traverses through links e_2 and e_3 and session 4 traverses through link e_3. Every layer consumes unit bandwidth.

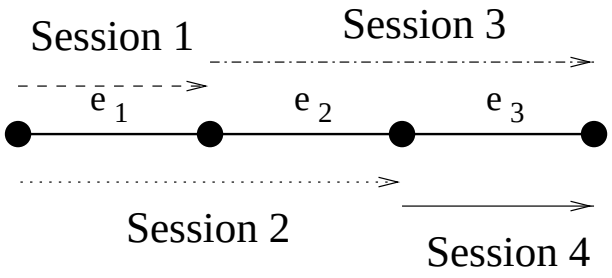

Figure 7:

$$C_{e_i} = \begin{cases} 1 & i = 1 \\ 4 & i = 2 \\ 6.2 & i = 3 \end{cases}$$

$$\eta_{e_i}(1) = \begin{cases} 0.5 & i = 1 \\ 2 & i = 2 \\ 3.1 & i = 3 \end{cases}$$

$$\omega_i(1) = \begin{cases} 0 & i \in \{1, 2\} \\ 2 & i = 3 \\ 3 & i = 4 \end{cases}$$

Observe that the difference between bandwidth consumed and capacity is at least 1 unit for every link. Thus $\Lambda(1) = \phi$. $\omega_j(1) < \min_{l \in L_j} \eta_l(1)$ for sessions $1, 2, 4$. Only sessions 1 and 2 satisfy the other criterion for incrementation though. However, if we choose to increment the rate of session 4 by 1 instead of either session 1 or 2, we obtain a rate vector $\vec{r}(1) = (0, 0, 2, 4)$. Sessions 3 and 4 are both saturated now. So $S(1) = \{1, 2\}$. Consider iteration 2. $\eta_{e_1}(2) = 0.5$, $\eta_{e_2}(2) = 2$. $\omega_s(2) = 1$, $s \in \{1, 2\}$. The difference between the bandwidth consumed and capacity is again at least 1 unit for both links e_1 and e_2. Thus $\Lambda(2) = \phi$. Both sessions 1 and 2 satisfy the criteria for incrementation. Incrementing the rate of session 1 (chosen arbitrarily amongst sessions $1, 2$) by 1 unit saturates sessions 1 and 2 and the algorithm terminates with rate allocation $(1, 0, 2, 4)$. The layer allocation vector

is $(1,0,2,4)$ as well. Now, $(1,0,3,3)$ is a feasible layer allocation vector fairer than $(1,0,2,4)$. Thus the output of the algorithm $(1,0,2,4)$ is not a maximally fair layer allocation vector. This is because of the choice of session 4 as a candidate for rate incrementation in iteration 1, despite the fact that $\lfloor \eta_{e_3}(1) \rfloor = \min_{l \in L_4} \lfloor \eta_{e_3}(1) \rfloor$ and $\Omega_{3e_3}(1) = \omega_3(1) < \lfloor \eta_{e_3}(1) \rfloor = 3$.

We would like to mention that a layer obtained from an available bandwidth can be looked upon as a function or "utility" of a bandwidth. This utility function is a stair case function because a layer essentially corresponds to a range of bandwidth, the range extending from the minimum bandwidth required to attain the layer to that required to attain the next higher layer. Thus this function is not strictly increasing. An algorithm for computation of rates for attaining maxmin fair utilities in unicast networks is proposed in [10]. We consider multicast networks here. More importantly, the algorithm attains maxmin fair utiliies for strictly increasing utility functions only. We show that the operation of this algorithm is ambiguous for non strictly increasing utility functions in general and staircase utility functions in particular, even for unicast networks. The algorithm computes the available capacity for every link, where available capacity is the difference between the actual capacity and the bandwidths consumed by the saturated sessions. The available bandwidth is distributed amongst the sessions so as to equalize the utilities (number of layers in this case) obtained by them. Every session gets a bandwidth equal to the minimum on its path. The sessions with minimum bandwidth amongst all unsaturated sessions are considered saturated. Now consider Example 4.3. Let the utility of a bandwidth or the number of layers equivalent to a given bandwidth be the floor of the bandwidth divided by the bandwidth consumed by every layer, which is 1 unit in this case. Thus if x units of bandwidth is assigned to a session, it gets $\lfloor x \rfloor$ layers. The capacity of the link will be utilized and both the sessions will get equal number of layers only if they are assigned α and $1-\alpha$ units respectively, where $0 < \alpha < 1$. If $\alpha \leq 0.5$, session 1 is saturated. If $\alpha \geq 0.5$, session 2 is saturated. The algorithm does not specify the value of α in this case. For $\alpha = 0.5$, both sessions saturate and the algorithm terminates with a layer allocation $(0,0)$. Wlog $\alpha < 0.5$. Session 1 is saturated with 0 layers. The available capacity is $1-\alpha$ units. It has to be assigned to session 2, which still gets $\lfloor 1-\alpha \rfloor = 0$ layers. Session 2 now saturates. So for all possible executions of this algorithm, the layer allocation is $(0,0)$. This clearly is not a maximally fair layer

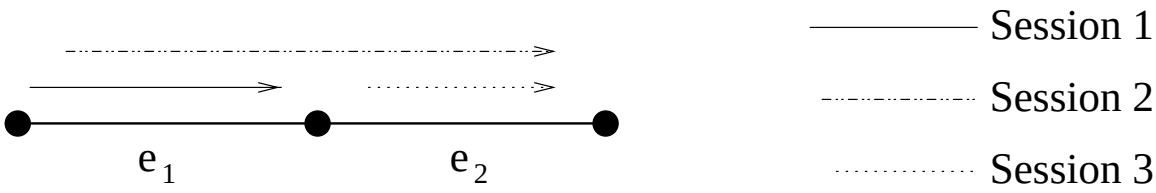

Figure 8:

allocation. So this approach does not obtain maximally fair layer allocation always. It remains to be seen whether the algorithm in [10] can be modified to attain maximally fair rates for non strictly increasing utility functions. No modification will attain maxmin fair layer allocation in all cases, since maxmin fair layer allocation does not exist always (Examples 4.3). So [10] does not apply to our case.

A lexicographically optimal layer allocation is the best one can hope for, when the feasible set is discrete. However, computation of a lexicographically optimal layer allocation for discrete bandwidth layers is a NP-hard problem. Heuristics for this computation is a topic of future research. It is interesting to note that every maximally fair layer allocation is an output of the algorithm for computation of a maximally fair layer allocation we proposed. A particular output in a specific instance depends upon the choice of the virtual session for possible incrementation of rate in step (8), amongst those virtual sessions s which satisfy the conditions of step (8) at the kth iteration, for various ks. In general, in any iteration, there can be more than one virtual sessions which satisfy these properties. Consider the following example.

Example 6.2: Refer to the network shown in Figure 8. The network has two links, e_1 and e_2 and 3 unicast sessions. Session 1 traverses link e_1, session 2 traverses both links and session 3 traverses link e_2. Both links have unit capacity and every layer consumes 1 unit of bandwidth. No session has any minimum number of layer requirement. $\eta_{e_1}(1) = \eta_{e_2}(1) = 0.5$. $\eta_{ie_1}(1) = 0.5$ for sessions $i \in \{1, 2\}$ and $\eta_{ie_2}(1) = 0.5$ for sessions $i \in \{2, 3\}$. It follows that $\omega_s(1) = 0$, $s \in \{1, 2, 3\}$. None of the sessions (sessions and virtual sessions are the same in this case) are saturated. All the sessions satisfy the properties for incrementation in step (8). If session 2 is chosen for incrementation, $\vec{r}(1) = (0, 1, 0)$ and all the sessions saturate. Thus the final rate allocation vector and the layer allocation vector are both $(0, 1, 0)$. If session 1 is chosen,

$\vec{r}(2) = (1,0,0)$. Sessions 1 and 2 saturate. $\eta_{e_2}(2) = 1$. $\eta_{3e_2}(2) = 1$. $\omega_3(2) = 1$. Session 3 saturates. The final rate allocation vector and layer allocation vector are both $(1,0,1)$. The only possible maximally fair layer allocation vectors are $(1,0,1)$ and $(0,1,0)$. Both of these are possible outputs of the algorithm. Note that $(1,0,1)$ is lexicographically greater than $(0,1,0)$ and turns out to be lexicographically optimal. Thus a lexicographically optimal layer allocation vector is a possible output of this algorithm.

In general by Lemma 5, a lexicographically optimal layer allocation vector is maximally fair and hence a possible output of the algorithm. It suggests that heuristics for making a "good" choice of the virtual sessions as candidates for rate incrementation may serve as good heuristics for yielding a lexicographic optimal layer allocation. If such heuristics fail to yield a lexicographic optimal layer allocation, they would at least yield maximally fair layer allocations. A study of such heuristics is a topic of future research. It may also be a good idea to run the algorithm for computation of a maximally fair vector a few times (it has polynomial complexity), each time with a different choice for candidates for layer incrementation and to choose finally that layer allocation which is lexicographically largest amongst the available ones.

An interesting question is whether every maximally fair layer allocation is provably close to a lexicographically optimal layer allocation, w.r.t. some distance definition. To investigate this, we define the concept of *lexicographic difference*. Given any two layer allocation vectors $\vec{\gamma}^1$ and $\vec{\gamma}^2$, the lexicographic difference between $\vec{\gamma}^1$ and $\vec{\gamma}^2$ is $\hat{\gamma}^1_i - \hat{\gamma}^2_i$, where $\hat{\gamma}^k$ is the lexicographic ordered version of $\vec{\gamma}^k$ and $i = \arg\min_{\hat{\gamma}^1_j \neq \hat{\gamma}^2_j} j$, i.e., the lexicographic difference between two layer allocations is the difference between the smallest components which differ in their lexicographically ordered versions. Lexicographic difference between two layer allocations $\vec{\gamma}^1$ and $\vec{\gamma}^2$ is positive, zero, or negative depending on whether $\vec{\gamma}^1$ is lexicographically greater than, equal to, less than $\vec{\gamma}^2$ respectively. Unfortunately, given any integer k, one can construct a network where the lexicographic difference between a lexicographically optimal layer allocation and one particular maximally fair layer allocation is k units. Consider the following example.

Example 6.3: Consider the network shown in Figure 9. It consists of $k+1$

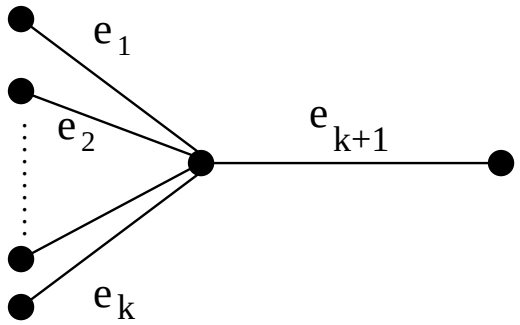

Figure 9:

links, $e_1, \ldots, e_{k+1}$ and $2k+1$ unicast sessions, session $1, \ldots, 2k+1$. Session m traverses through edges $e_{\lceil \frac{m}{2} \rceil}$ and e_{k+1} for odd m, $m \leq 2k-1$. Session m traverses through edge $e_{\frac{m}{2}}$ for even m. Session $2k+1$ traverses through edge e_{k+1}. Edges $1, \ldots, k$ have unit capacity and edge $k+1$ has $2k$ units of capacity. Every layer consumes 1 unit of bandwidth. The lexicographic optimal layer allocation vector $\vec{\gamma}^O$ is as follows:

$$\gamma_i^O = \begin{cases} 0 & i \text{ odd}, i \leq 2k-1 \\ 1 & i \text{ even}, \\ 2k & i = 2k+1. \end{cases}$$

The layer allocation vector $\vec{\gamma}^1$ described below is a maximally fair layer allocation.

$$\gamma_i^1 = \begin{cases} 1 & i \text{ odd}, i \leq 2k-1 \\ 0 & i \text{ even} \\ k & i = 2k+1. \end{cases}$$

Observe that the lexicographically ordered version of $\vec{\gamma}^O$, $\hat{\gamma}^0$ is as follows:

$$\hat{\gamma}_i^O = \begin{cases} 0 & i \leq k \\ 1 & k+1 \leq i \leq 2k \\ 2k & i = 2k+1 \end{cases}$$

and the lexicographically ordered version of $\vec{\gamma}^1$, $\hat{\gamma}^1$ is as follows:

$$\hat{\gamma}_i^1 = \begin{cases} 0 & i \leq k \\ 1 & k+1 \leq i \leq 2k \\ k & i = 2k+1. \end{cases}$$

$\hat{\gamma}_i^O = \hat{\gamma}_i^1$, if $i \neq 2k+1$. It follows that the lexicographic difference between $\vec{\gamma}^O$ and $\vec{\gamma}^1$ is $\hat{\gamma}_{2k+1}^O - \hat{\gamma}_{2k+1}^1 = k$ units. The algorithm for generation of maximally fair layer allocation may result in $\vec{\gamma}^1$, if it chooses sessions $1, 3, \ldots, 2k-1$ for incrementation in step (8). These are apparently "bad" choices. The choice of sessions $2, 4, \ldots, 2k$ for incrementation in step (8) are "good" choices. However there is no way of knowing which are "good" and which are "bad" choices in general. Heuristics like preferring sessions which span fewer number of links, for incrementation may be used. Also, randomized choice of candidates for rate incrementation amongst those which satisfy the condition of step (8) may be useful. In this example, all bad choices, $1, 3, \ldots, 2k-1$ will be made with very low probability. So expected lexicographic difference of the output maximally fair layer allocation from the lexicographic optimal, will be around $k/2$ units. The research problem of developing a polynomial complexity algorithm for generation of a maximally fair layer allocation which is within lexicographic difference of k units from a lexicographic optimal layer allocation, for some constant k, is essentially open. It is not even known whether there can exist one such algorithm.

We have so far assumed that every layer of every source consumes the same bandwidth, i.e., b units. This assumption have been made elsewhere, as well, e.g., while simulating the RLM internet protocol, [24] assumes that every layer consumes 32 kb/s bandwidth. However, this assumption does not hold in all coding schemes. There may be more complicated situations where the hierarchical signal structure of some sessions are flexible while the signal structure is predetermined for some others, i.e., possible service rates of receivers of some sessions form a continuous set while those of receivers of some other sessions form a discrete set. Also, it is possible that layer bandwidths can be fine tuned in certain ranges, while the granulartity is coarse in other ranges. Fairness in these scenarios becomes more technical, and is beyond the scope of the current paper. This is a topic of future research. The simple case we have studied here gives the essential intuition, though.

Computation of the maxmin fair rates and a maximally fair allocation of layers in a distributed manner is a topic of future research. The criteria for determination of rates uses information along L_s mainly. The only place where the algorithm uses global information is that $r_s(k) = \omega_s(k) + b$, for at most one virtual session, s. However, $r_s(k) = \omega_s(k) + b$ can hold for multiple virtual sessions, as long as they satisfy the criteria of step (8), subject to

feasibility and the algorithm will still output a maximally fair layer allocation. This feature of the algorithm ($r_s(k) = \omega_s(k)+b$ for at most one virtual session s) is not crucial to the proof of maximal fairness of the output and is a matter of convenience. This may facilitate the developement of a distributed algorithm for computation of a maximally fair layer allocation. However, the details for the distributed implementation need to be worked out.

We have also not addressed the issue of developing the detailed protocol for informing the source, how many layers it should transmit and informing the receivers which groups they should subscribe to. Developing such protocols would be interesting from an implementation point of view.

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

A Properties of Maxmin Fair Vector

In this section we prove some general properties of a maximally fair vector. This leads to some interesting properties of a maxmin fair vector, if it exists. We consider arbitrary K-dimensional vectors for some integer K and consider an arbitrary feasible set. The definitions of maximal fairness, lexicographic optimality and maxmin fairness remain the same as those used before, only the reference feasible set of vectors is arbitrary.

Proposition 1 *Consider two $M-$dimensional vectors $\vec{A}$ and $\vec{B}$. If $\vec{A}$ is fairer than $\vec{B}$ then there exists a component s such that $a_s = \min_{j\in\tau} a_j$ and $a_s > b_s$, where $\tau = \{j : a_j \neq b_j\}$, a_j and b_j are the jth components of $\vec{A}$ and $\vec{B}$ respectively.*

Remark: This proposition has been proved in [28]. We prove it for completeness. We will use this result later.

Proof of Proposition 1: Let $\vec{A}$ be fairer than $\vec{B}$. Thus $\vec{A} \neq \vec{B}$. $\tau \neq \phi$. Let there exist no s such that $a_s = \min_{j \in \tau} a_j$ and $a_s > b_s$, i.e., $a_j < b_j$, $\forall j$ such that $a_j = \min_{p \in \tau} a_p$ ($a_j \neq b_j$, $\forall j \in \tau$). Consider one such j. Since $\vec{A}$ is fairer than $\vec{B}$, and $a_j < b_j$, there exists a u such that $a_u > b_u$ and $a_u \leq a_j$ Since $a_u \neq b_u$, $u \in \tau$. Thus $a_u \geq a_j$. It follows that $a_u = a_j$. Thus $a_u = \min_{p \in \tau} a_p$ and $a_u > b_u$. This contradicts the fact that $a_j < b_j$, $\forall j$ such that $a_j = \min_{p \in \tau} a_p$. Thus if $\vec{A}$ is fairer than $\vec{B}$, there exists a component s such that $a_s = \min_{j \in \tau} a_j$ and $a_s > b_s$. □

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

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

Lemma 4 *If $\vec{A}$ is maxmin fair, then it is lexicographically optimal amongst all feasible vectors.*

Proof of Lemma 4: If $\vec{A}$ is maxmin fair, then by definition of maxmin fairness and maximal fairness, $\vec{A}$ is fairer than any other feasible vector. Lexicographical optimality follows from Lemma 3. □

Lemma 5 *If $\vec{A}$ is lexicographically optimal in a feasible set, then it is maximally fair in the same feasible set.*

Proof of Lemma 5: Let $\vec{A}$ not be maximally fair in a feasible set. Thus there exists another feasible vector $\vec{B}$, fairer than $\vec{A}$. It follows that $\vec{B}$ is lexicographically greater than $\vec{A}$ (by Lemma 3). This contradicts the lexicographic optimality of $\vec{A}$. □

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

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

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

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

Every edge has unit capacity. No session has any minimum number of layers requirement. Let $\{i_1, \ldots, i_p, k_1, \ldots, k_t\}$ constitute the adjacency set of j and $i_1 < i_2 < \ldots < i_p < j < k_1 < k_2 < \ldots < k_t$. The path of session j consists of nodes $v_{i_1j1}, v_{i_1j2}, v_{i_2j1}, v_{i_2j2}, \ldots, v_{i_pj1}, v_{i_pj2}, v_{jk_11}, v_{jk_12}, \ldots, v_{jk_t1}, v_{jk_t2}$ 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 polunomial function of the input graph G size).

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

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

We will prove that layer allocation $\vec{\gamma}^S$ is feasible iff S is an independent set in G. By construction, two sessions i, j share a link iff vertices i and j are

adjacent in G. (Wlog $i < j$.) This follows because if i, j share a link then there exists a l such that $n(l) = \{i, j\}$ and this means that $l = (v_{ij1}, v_{ij2})$. This implies that there exists an edge between vertices i, j in G. If i and j are adjacent in G, then by construction, both sessions i, j traverse through link l, where $l = (v_{ij1}, v_{ij2})$.

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

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

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

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

mum independent set has been generated in polynomial time. Generation of maximum independent set of a graph G is a known NP-hard problem. Thus $P = NP$. □

B Proof of Pseudo-Bottleneck Lemma

Proof of Lemma 2 (Pseudo-Bottleneck Lemma): Let a virtual session s not have a pseudo-bottleneck link under a feasible layer allocation vector $\vec{\gamma}^1$. We will show that it can not be a maximally fair layer allocation vector. Let

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

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

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

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

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

Consider a link $l \in L_s$. Since $\gamma^2_j \leq \gamma^1_j$, if $j \neq s$ and $\gamma^2_s = \gamma^1_s + 1$,

$$\begin{aligned} \Gamma^2_{il} &\leq \Gamma^1_{il}, \text{ if } i \neq \chi(s) \text{ and} && (3) \\ \Gamma^2_{\chi(s)l} &\leq \max(\gamma^1_s + 1, \Gamma^1_{\chi(s)l}) \\ &\leq \Gamma^1_{\chi(s)l} + 1 && (4) \end{aligned}$$

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

$$\begin{aligned} b \sum_{i \in n(l)} \Gamma^2_{il} &\leq b \sum_{i \in n(l)} \Gamma^1_{il} + b \\ &\leq C_l \text{ (since } l \in Y) \end{aligned} \tag{5}$$

2. Let $l \in Z$. Since γ^1_j is an integer for all j, so is Γ^1_{il} for all sessions i and link l. Thus $\gamma^1_s < \Gamma^1_{\chi(s)l}$, means that $\gamma^1_s + 1 \leq \Gamma^1_{\chi(s)l}$. Thus $\Gamma^2_{\chi(s)l} \leq \max(\gamma^1_s + 1, \Gamma^1_{\chi(s)l}) = \Gamma^1_{\chi(s)l}$. Thus from (3) $\sum_{i \in n(l)} \Gamma^2_{il} \leq \sum_{i \in n(l)} \Gamma^1_{il}$. Again it follows that

$$\begin{aligned} b \sum_{i \in n(l)} \Gamma^2_{il} &\leq b \sum_{i \in n(l)} \Gamma^1_{il} \\ &\leq C_l \text{ (from the feasibility of } \vec{\gamma}^1) \end{aligned} \tag{6}$$

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 $\chi(p) \in n(l)$, $\gamma^1_p > \gamma^1_s + 1$ and $\gamma^1_p > \iota_{\chi(p)l}$. Consider all virtual sessions $j \in \chi(p)$ such that $\gamma^1_j = \Gamma^1_{\chi(p)l}$. It follows that $\gamma^1_j \geq \gamma^1_p$. Thus $\gamma^1_j > \iota_{\chi(j)l}$, $\gamma^1_j > \gamma^1_s + 1$. Hence $j \in W$ if $\gamma^1_j = \Gamma^1_{\chi(p)l}$, $j \in \chi(p)$. Now $\gamma^2_j = \gamma^1_j - 1$ for all $j \in W$. It follows that $\Gamma^2_{\chi(p)l} = \Gamma^1_{\chi(p)l} - 1$.

$$\begin{aligned} \sum_{i \in n(l)} \Gamma^2_{il} &= \sum_{i \in n(l) \setminus \{\chi(p), \chi(s)\}} \Gamma^2_{il} + \Gamma^2_{\chi(s)l} + \Gamma^2_{\chi(p)l} \\ &\leq \sum_{i \in n(l) \setminus \{\chi(p), \chi(s)\}} \Gamma^1_{il} + \Gamma^1_{\chi(s)l} + 1 + \Gamma^1_{\chi(p)l} - 1 \text{ (from (3) and (4))} \\ &= \sum_{i \in n(l)} \Gamma^1_{il} \end{aligned}$$

$$\begin{aligned} \text{Thus } b \sum_{i \in n(l)} \Gamma^2_{il} &\leq b \sum_{i \in n(l)} \Gamma^1_{il} \\ &\leq C_l \text{ (from the feasibility of } \vec{\gamma}^1) \end{aligned} \tag{7}$$

4. Let $l \notin L_s$.

$$\begin{aligned} b \sum_{i \in n(l)} \Gamma^2_{il} &\leq b \sum_{i \in n(l)} \Gamma^1_{il} \text{ (from (3) and since } \chi(s) \notin n(l)) \\ &\leq C_l \text{ (from feasibility of } \vec{\gamma}^1) \end{aligned} \tag{8}$$

Thus from (5), (6), (7) and (8) $\vec{\gamma}^2$ 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 in the appendix. Since γ_s^1 and γ_s^2 are both integers, $\gamma_s^2 > \gamma_s^1$ means that

$$\gamma_s^2 \geq \gamma_s^1 + 1 \tag{9}$$

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

$$\begin{aligned} \Gamma^2_{\chi(s)l} &\geq \gamma_s^2 \\ &\geq \gamma_s^1 + 1 \text{ (by (9))} \\ &= \Gamma^1_{\chi(s)l} + 1 \text{ (by pseudo-bottleneck property (2))} \end{aligned} \tag{10}$$

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

$$\begin{aligned} \sum_{i\in n(l)} \Gamma^2_{il} &\geq \sum_{i\in n(l)} \Gamma^1_{il} + 1 \text{ (from (10))} \\ \text{Thus } b \sum_{i\in n(l)} \Gamma^2_{il} &> C_l \text{ (from pseudo-bottleneck property (1))} \end{aligned} \tag{11}$$

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

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

$$
\begin{aligned}
&= \gamma_j^1 \text{ (from choice of } j) && (12)\\
&\leq \gamma_s^1 + 1 \\
&\leq \gamma_s^2 \text{ (from (9))} && (13)
\end{aligned}
$$

From (12) $\gamma_j^2 \neq \gamma_j^1$. Thus $j \in \tau$. From (13) $\gamma_j^2 < \gamma_s^2$. 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. □

C Proof of Maximal-Fairness Theorem

We assume that the set of feasible rate vectors is nonempty.

Lemma 6 *$r_s(k)$ and $\lambda_{il}(k)$ are multiples of b for all virtual sessions s, sessions i, link l and iterations k.*

Proof of Lemma 6: $r_s(0) = \mu_s$. 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 7 *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 7: 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 6. 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 $i \notin \Xi_l(k-1)$, $\Omega_{il}(k) = \lambda_{il}(k-1)$ since $\omega_j(k) = r_j(k-1)$, $\forall j \in m(\chi(j), l)$. It follows that $\sum_{i \in n(l)} \Omega_{il}(k) \leq \sum_{i \in n(l) \setminus \Xi_l(k-1)} \lambda_{il}(k-1) + \sum_{i \in \setminus \Xi_l(k-1)} \max(\eta_l(k), \lambda_{il}(k-1)) = C_l$. 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.

Lemma 8 *$r_s(k+1) \geq r_s(k)$ if $k \geq 0$ for all virtual sessions s. $\lambda_{il}(k+1) \geq \lambda_{il}(k)$ if $k \geq 0$ for all sessions i and links l*

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

$$\begin{aligned}\eta_{l\chi(s)}(k+1) &= \max(\eta_l(k+1), \lambda_{\chi(s),l}(k)) \\ &\geq r_s(k) \text{ (since } s \in m(\chi(s), l) \ \forall\ l \in L_s \text{ and } \lambda_{il}(k) = \max_{s \in m(i,l)} r_s(k)) \quad (14)\\ \omega_s(k+1) &= b\lfloor \frac{\min_{l \in L_s} \eta_{l\chi(s)}(k+1)}{b} \rfloor \\ &\geq b\lfloor \frac{r_s(k)}{b} \rfloor \text{ (from (14))} \\ &= r_s(k) \text{ (by Lemma 6)}\end{aligned}$$

$$\text{Thus } \omega_s(k+1) \geq r_s(k) \ \forall\ s \quad (15)$$

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 *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 9: Let $k = 1$. The algorithm can not terminate in 0 iterations. $S(0) \neq \phi$, $F_l(0) = 0$. If $\Xi_l(0) = \phi$, then $\eta_l(1) = \eta_l(0) = 0$, and the lemma holds for link l and iteration 1. Let $\Xi_l(0) \neq \phi$. Since the feasible

set of rate vectors is nonempty, $\sum_{i\in\Xi_l(0)}\mu_{il} \leq C_l$ 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 $\sum_{i\in\Xi_l(0)}\max(\theta, \lambda_{il}(0)) \leq C_l$. 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 $l \in L_j$, $r_j(k-1) \leq \lambda_{\chi(j),l}(k-1) \leq \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{\chi(j),l}(k-1))$.

$$\begin{aligned} r_j(k) &\leq \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{\chi(j),l}(k-1)) \text{ if } j \notin S(k-1), l \in L_j \quad (16)\\ &\leq \max(\eta_l(k), \lambda_{\chi(j),l}(k-1)) \text{ if } j \notin S(k-1), l \in L_j \quad (17)\end{aligned}$$

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

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

$$r_j(k) \leq \max(\eta_l(k), \lambda_{\chi(j),l}(k-1)) \text{ if } j \in S(k-1), l \in L_j \quad (18)$$

$$\lambda_{il}(k) \leq \max(\eta_l(k), \lambda_{il}(k-1)),\ \forall\ i \in n(l) \quad \text{(from (17) and (18))} \quad (19)$$

$$\sum_{i\in\Xi_l(k)}\max(\eta_l(k), \lambda_{il}(k)) \leq \sum_{i\in\Xi_l(k)}\max(\eta_l(k), \lambda_{il}(k-1)) \text{ (from (19))} \quad (20)$$

$$\begin{aligned} F_l(k) &= F_l(k-1) + \sum_{i\in\Xi_l(k-1)\setminus\Xi_l(k)}\lambda_{il}(k)\\ &\leq F_l(k-1) + \sum_{i\in\Xi_l(k-1)\setminus\Xi_l(k)}\max(\eta_l(k), \lambda_{il}(k-1)) \text{ (from (19))} \quad (21)\end{aligned}$$

$$\begin{aligned} F_l(k) + \sum_{i\in\Xi_l(k)}\max(\eta_l(k), \lambda_{il}(k)) &\leq F_l(k-1) + \sum_{i\in\Xi_l(k-1)\setminus\Xi_l(k)}\max(\eta_l(k), \lambda_{il}(k-1)) +\\ &\quad \sum_{i\in\Xi_l(k)}\max(\eta_l(k), \lambda_{il}(k-1)) \text{ (from (20) and (21))}\end{aligned}$$

$$= F_l(k-1) + \sum_{i \in \Xi_l(k-1)} \max(\eta_l(k), \lambda_{il}(k-1)) \quad (22)$$

(since $\Xi_l(k) \subseteq \Xi_l(k-1)$ as $S(k) \subseteq S(k-1)$)

$$= C_l \quad (23)$$

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 6) and $\omega_s(k) < \min_{l_1 \in L_s} \eta_{l_1}(k)$. Thus

$$\begin{aligned} \omega_s(k) &\leq b\lfloor \frac{\min_{l_1 \in L_s} \eta_{l_1}(k)}{b} \rfloor \text{ (proof of Lemma 6)} \\ &< b\lfloor \frac{\eta_l(k)}{b} \rfloor \text{ by hypothesis.} \\ \text{Thus } \omega_s(k) + b &\leq b\lfloor \frac{\eta_l(k)}{b} \rfloor \text{ (by Lemma 6)} \\ r_s(k) &= \omega_s(k) + b \\ &\leq b\lfloor \frac{\eta_l(k)}{b} \rfloor \\ &\leq \max(b\lfloor \frac{\eta_l(k)}{b} \rfloor, \lambda_{\chi(s)l}(k-1)) \quad (24) \end{aligned}$$

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 $j \neq s$.

$$\begin{aligned} \omega_j(k) &\leq b\lfloor \frac{\eta_{\chi(j)l}(k)}{b} \rfloor \ \forall\ j \\ &= \max(b\lfloor \frac{\eta_l(k)}{b} \rfloor, \lambda_{\chi(j)l}(k-1)) \text{ (by Lemma 6)} \\ \text{Thus } r_j(k) &\leq \max(b\lfloor \frac{\eta_l(k)}{b} \rfloor, \lambda_{\chi(j)l}(k-1)) \ \forall\ j \in S(k-1), j \neq s \quad (25) \\ \lambda_{il}(k) &\leq \max(b\lfloor \frac{\eta_l(k)}{b} \rfloor, \lambda_{il}(k-1)) \ \forall\ i \in n(l) \quad (26) \end{aligned}$$

(from (16), (24) and (25))

Using (26) it can be proved that

$$F_l(k) + \sum_{i \in \Xi_l(k)} \max(b\lfloor \frac{\eta_l(k)}{b} \rfloor, \lambda_{il}(k)) \ \leq \ F_l(k-1) +$$

$$\sum_{i\in\Xi_l(k-1)} \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{il}(k-1)) \qquad (27)$$

$$\leq F_l(k-1) + \sum_{i\in\Xi_l(k-1)} \max(\eta_l(k), \lambda_{il}(k-1))$$

$$\text{(since } \eta_l(k) \geq b\lfloor\frac{\eta_l(k)}{b}\rfloor\text{)}$$

$$\leq C_l \text{ (by induction hypothesis)} \qquad (28)$$

The proof for (27) is exactly similar to the one for (22) using (19).

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 $r_s(k) \leq \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{\chi(s)l}(k-1))$

$$r_j(k) = \omega_j(k), j \neq s$$

$$\leq \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{il}(k-1)) \text{ if } j \in S(k-1) \qquad (29)$$

$$\lambda_{il}(k) \leq \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{il}(k-1)) \ \forall i \in n(l) \qquad (30)$$

(from (16), assumption and (29))

Now using (30) it can be proved that

$$F_l(k) + \sum_{i\in\Xi_l(k)} \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{il}(k)) \leq C_l \qquad (31)$$

The proof is exactly similar to the one for (28) using (26).

Now let

$$r_s(k) > \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{il}(k-1)) \qquad (32)$$

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

$$\lambda_{il}(k) \geq \Omega_{il}(k)$$

$$\geq b\lfloor\frac{\eta_l(k)}{b}\rfloor \text{ (from the algorithm)}$$

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

$$\begin{aligned} \text{Thus } \max(b\lfloor \frac{\eta_l(k)}{b} \rfloor, \lambda_{il}(k)) &= \lambda_{il}(k) \\ &= \Omega_{il}(k) \qquad (33) \\ &\quad (\text{if } i \neq \chi(s), \text{ since then } r_j(k) = \omega_j(k)\ \forall j \in m(i,l)) \end{aligned}$$

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

$$\begin{aligned} \omega_s(k) &\leq \max(b\lfloor \frac{\eta_l(k)}{b} \rfloor, \lambda_{\chi(s)l}(k-1)) \text{ (since } l \in L_s) \\ r_s(k) &\leq \max(b\lfloor \frac{\eta_l(k)}{b} \rfloor, \lambda_{\chi(s)l}(k-1)) + b \qquad (34) \end{aligned}$$

From (32), (34) and since $r_s(k)$ is a multiple of b, by Lemma 6, $r_s(k) = \max(b\lfloor \frac{\eta_l(k)}{b} \rfloor, \lambda_{\chi(s)l}(k-1)) + b$.

$$\begin{aligned} \text{If } \chi(j) \in n(l), \omega_j(k) &\leq \max(b\lfloor \frac{\eta_l(k)}{b} \rfloor, \lambda_{\chi(j)l}(k-1)), j \in S(k-1) \qquad (35) \\ \omega_j(k) &= r_j(k-1),\ j \notin S(k-1) \\ &\leq \max(b\lfloor \frac{\eta_l(k)}{b} \rfloor, \lambda_{\chi(j)l}(k-1)),\ j \notin S(k-1) \qquad (36) \\ \Omega_{\chi(s)l}(k) &\leq \max(b\lfloor \frac{\eta_l(k)}{b} \rfloor, \lambda_{\chi(s)l}(k-1)) \\ &\qquad (\text{from (35) and (36)}) \\ \text{Thus } \max(\Omega_{\chi(s)l}(k), r_s(k)) &= \max(b\lfloor \frac{\eta_l(k)}{b} \rfloor, \lambda_{\chi(s)l}(k-1)) + b \qquad (37) \\ r_j(k) &= \omega_j(k), j \neq s \\ r_s(k) &= \omega_s(k) + b \\ \text{Thus } \lambda_{\chi(s)l}(k) &= \max(\Omega_{\chi(s)l}(k), r_s(k)) \\ &= \max(b\lfloor \frac{\eta_l(k)}{b} \rfloor, \lambda_{\chi(s)l}(k-1)) + b \text{ (from (37))} \qquad (38) \\ \max(b\lfloor \frac{\eta_l(k)}{b} \rfloor, \lambda_{\chi(s)l}(k)) &= \max(b\lfloor \frac{\eta_l(k)}{b} \rfloor, \lambda_{\chi(s)l}(k-1)) + b \text{ (from (38))} \\ &= \max(\Omega_{\chi(s)l}(k), r_s(k)) \text{ (from (37))} \qquad (39) \end{aligned}$$

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

$$\begin{aligned}
F_l(k) + \sum_{i\in\Xi_l(k)} \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{il}(k)) &= F_l(k-1) + \sum_{i\in\Xi_l(k-1)\setminus\Xi_l(k)} \lambda_{il}(k) + \\
&\quad \sum_{i\in\Xi_l(k)} \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{il}(k)) \\
&\quad \text{(since } \Xi_l(k) \subseteq \Xi_l(k-1)\text{)} \\
&\leq F_l(k-1) + \sum_{i\in\Xi_l(k-1)\setminus\Xi_l(k)} \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{il}(k)) + \\
&\quad \sum_{i\in\Xi_l(k)} \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{il}(k)) \\
&= F_l(k-1) + \sum_{i\in\Xi_l(k-1)} \max(b\lfloor\frac{\eta_l(k)}{b}\rfloor, \lambda_{il}(k)) \\
&\quad \text{(since } \Xi_l(k-1) \supseteq \Xi_l(k)\text{)} \\
&= F_l(k-1) + \sum_{i\in\Xi_l(k-1), i\neq\chi(s)} \Omega_{il}(k) \\
&\quad + \max(\Omega_{\chi(s)l}(k), r_s(k)) \\
&\quad \text{(since } \chi(s) \in \Xi_l(k-1)\text{) (from (33) and (39))} \\
&= \sum_{i\in n(l)\setminus\Xi_l(k-1)} \Omega_{il}(k) + \sum_{i\in\Xi_l(k-1), i\neq\chi(s)} \Omega_{il}(k) \\
&\quad + \max(\Omega_{\chi(s)l}(k), r_s(k)) \\
&\quad \text{(since } \omega_j(k) = r_j(k-1) \forall j \in n(l) \setminus S(k-1)\text{)} \\
&= \sum_{i\in n(l), i\neq\chi(s)} \Omega_{il}(k) + \max(\Omega_{\chi(s)l}(k), w_s(k) + b) \\
&\leq C_l) \qquad (40)
\end{aligned}$$

(40) follows since from Lemma 7, 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 (23), (28), (31) and (40), $\theta = b\lfloor\frac{\eta_l(k)}{b}\rfloor$ satisfies the inequality $F_l(k) + \sum_{i\in\Xi_l(k)} \max(\theta, \lambda_{il}(k))$

$\leq C_l$. 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 10 *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 $k \geq 0$.*

Proof of Lemma 10: $r_s(0) = \mu_s \leq \mu_{\chi(s)l}\ \forall l \in L_s$ 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+1th 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 9, $\lfloor\frac{\eta_l(k)}{b}\rfloor \leq \lfloor\frac{\eta_l(k+1)}{b}\rfloor$. 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 9, $\lceil\frac{\eta_l(k)}{b}\rceil \leq \lceil\frac{\eta_l(k+1)}{b}\rceil$. 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 6), $\omega_s(k+1) \leq b\lfloor\frac{\eta_l(k+1)}{b}\rfloor$, 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. □

Lemma 11 *The rate allocation $\vec{r}(k)$ at the end of the kth iteration is feasible, $k \geq 0$.*

Proof of Lemma 11: 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

$$\Omega_{il}(k+1) \leq \max(\eta_l(k+1), \lambda_{il}(k)), \ \forall \ i \in n(l) \tag{41}$$

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

$$\begin{aligned}
\sum_{i \in n(l)} \Omega_{il}(k+1) &= \sum_{i \in n(l) \setminus \Xi_l(k)} \Omega_{il}(k+1) + \sum_{i \in \Xi_l(k)} \Omega_{il}(k+1) \\
&\leq \sum_{i \in n(l) \setminus \Xi_l(k)} \lambda_{il}(k) + \sum_{i \in \Xi_l(k)} \max(\eta_l(k+1), \lambda_{il}(k)) \\
&= F_l(k) + \sum_{i \in \Xi_l(k)} \max(\eta_l(k+1), \lambda_{il}(k)) \\
&\leq C_l
\end{aligned} \tag{42}$$

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 (42). 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 $\lambda_{\chi(s)l}(k+1) = \max(\omega_s(k+1) + b, \Omega_{\chi(s)l}(k+1))$.

$$\sum_{i \in n(l)} \lambda_{il}(k+1) = \sum_{i \in n(l), i \neq \chi(s)} \Omega_{il}(k+1) + \max(\omega_s(k+1) + b, \Omega_{\chi(s)l}(k+1))$$

$$\leq \quad C_l \text{ (from Lemma 7, since } r_s(k+1) = \omega_s(k+1) + b, s \in S(k), \text{ and } \Lambda(k+1) =$$

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

Lemma 12 *If $s \notin S(k)$, $k \geq 0$, there exists a link $l \in L_s$ such that $\sum_{i \in n(l)} \lambda_{il}(k) > C_l - b$, $r_s(k) = \lambda_{\chi(s)l}(k)$ and for any virtual session j traversing the link ($\chi(j) \in n(l)$), if $r_j(k) > \mu_{\chi(j)l}$ then $r_j(k) \leq r_s(k) + b$.*

Proof of Lemma 12: 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 $l \in L_s$,

$$\sum_{i \in n(l)} \lambda_{il}(t+1) \quad > \quad C_l - b \tag{43}$$

$$r_s(t+1) \quad = \quad \lambda_{\chi(s)l}(t+1) \tag{44}$$

$$\begin{aligned} \sum_{i \in n(l)} \lambda_{il}(k) \quad &\geq \quad \sum_{i \in n(l)} \lambda_{il}(t+1) \text{ (from Lemma 8 and since } t+1 \leq k) \\ &> \quad C_l - b \text{ (from (43))} \end{aligned} \tag{45}$$

(45) proves the first part of the lemma.

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 6).

$$\begin{aligned} C_l \quad &< \quad b + \lambda_{ul}(t+1) + \sum_{i \in n(l), i \neq u} \lambda_{il}(t+1) \text{ (from (43))} \\ &\leq \quad \lambda_{ul}(k) + \sum_{i \in n(l), i \neq u} \lambda_{il}(k) \\ &= \quad \sum_{i \in n(l)} \lambda_{il}(k) \end{aligned}$$

Thus $\vec{r}(k)$ is infeasible. This contradicts Lemma 11.

$$\text{Thus } \lambda_{il}(k) = \lambda_{il}(t+1), \forall\ i \in n(l). \tag{46}$$

Since $t+1 \leq k$, $r_s(t+1) \leq r_s(k)$ (Lemma 8). From (44) and (46), $r_s(k) \geq \lambda_{\chi(s)l}(k)$. Clearly from definition $r_s(k) \leq \lambda_{\chi(s)l}(k)$. Thus $r_s(k) = \lambda_{\chi(s)l}(k)$. This proves the second part of the lemma.

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

$$\begin{aligned}
C_l &< \sum_{i \in n(l) \setminus \{\chi(s)\}} \lambda_{il}(t+1) + (\lambda_{\chi(s)l}(t+1) + b) \quad \text{(from (43))} \\
&= \sum_{i \in n(l) \setminus \Xi_l(t)} \lambda_{il}(t+1) + \sum_{i \in \Xi_l(t), i \neq \chi(s)} \lambda_{il}(t+1) + \\
&\quad (b + \lambda_{\chi(s)l}(t+1)) \text{ (since } s \in S(t), \chi(s) \in \Xi_l(t)) \\
&\leq F_l(t) + \sum_{i \in \Xi_l(t), i \neq \chi(s)} \max(\eta_l(t+1), \lambda_{il}(t)) + \\
&\quad (b + \lambda_{\chi(s)l}(t)) \\
C_l &\geq F_l(t) + \sum_{i \in \Xi_l(t), i \neq \chi(s)} \max(\eta_l(t+1), \lambda_{il}(t)) + \\
&\quad \max(\eta_l(t+1), \lambda_{\chi(s)l}(t)) \\
\text{Thus } \lambda_{\chi(s)l}(t) + b &> \max(\eta_l(t+1), \lambda_{\chi(s)l}(t)) \\
&\geq \eta_l(t+1) \\
\text{Thus } r_s(t+1) + b &> \eta_l(t+1) \text{ (from (44))} \\
r_s(t+1) + b &\geq b\lceil \frac{\eta_l(t+1)}{b} \rceil \text{ (by Lemma 6)} \\
&\geq r_j(t+1) \text{ if } \chi(j) \in n(l), r_j(t+1) > \mu_{\chi(j)l} \text{ (by Lemma 10)} \\
\text{Thus } r_s(t+1) + b &\geq \lambda_{il}(t+1) \text{ if } i \in n(l), \lambda_{il}(t+1) > \mu_{il} \qquad (47)
\end{aligned}$$

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 6 $\omega_j(t+1) \leq b\lfloor \frac{\min_{l_1 \in L_j} \eta_{l_1}(t+1)}{b} \rfloor$. 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

$$\lfloor \frac{\eta_l(t+1)}{b} \rfloor = \lfloor \frac{\min_{l_1 \in L_j} \eta_{l_1}(t+1)}{b} \rfloor \qquad (48)$$

Since s traverses l and $s \in S(t)$, $\chi(s) \in \Xi_l(t)$. $l \in L_j$. Thus from (48), 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 $\lambda_{\chi(s)l}(t+1) \geq \Omega_{\chi(s)l}(t+1)$,

$$\begin{aligned}
\lambda_{\chi(s)l}(t+1) &\geq b\lfloor\frac{\eta_l(t+1)}{b}\rfloor \\
r_s(t+1) &\geq b\lfloor\frac{\eta_l(t+1)}{b}\rfloor \text{ (from (44))} \\
\text{Thus } r_s(t+1)+b &\geq b\lceil\frac{\eta_l(t+1)}{b}\rceil \\
&\geq r_k(t+1) \text{ if } \chi(k) \in n(l), r_k(t+1) > \mu_{\chi(k)l} \text{ (by Lemma 10)} \\
\text{Thus } r_s(t+1)+b &\geq \lambda_{il}(t+1) \text{ if } i \in n(l), \lambda_{il}(t+1) > \mu_{il} \qquad (49)
\end{aligned}$$

$r_s(t+1) \leq r_s(k)$ (Lemma 8). Thus from (46), (47) and (49), in both cases, $\lambda_{il}(k) \leq r_s(k) + b$, if $i \in n(l)$ and $\lambda_{il}(k) > \mu_{il}$. If $r_j(k) > r_s(k) + b$ and $\chi(j) \in n(l)$, then $\lambda_{\chi(j)l}(k) > r_s(k) + b$ and thus $\lambda_{\chi(j)l}(k) \leq \mu_{\chi(j)l}$. It follows that $r_j(k) \leq \mu_{\chi(j)l}$. The result follows. □

Proof of Theorem 1 (Maximal Fairness Theorem): Let the algorithm terminate in k iterations. Since the rate allocation $\vec{r}(k)$ is feasible by Lemma 11, the corresponding layer allocation vector, $\vec{\gamma}$ is feasible as well. Since $S(k) = \phi$, there does not exist a virtual session s in $S(k)$. Thus from Lemma 12 for every virtual session s there exists a link which satisfies the properties of a pseudo-bottleneck link w.r.t. virtual session s for rate vector $\vec{r}(k)$ and corresponding layer allocation vector $\vec{\gamma}$. Thus $\vec{\gamma}$ is a maximally fair layer allocation vector by Lemma 2. Thus (1) follows.

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

D Proof of Finite-Termination Theorem

The last theorem ensures that the algorithm terminates in a finite number of iterations. We prove it using the following lemmas.

Lemma 13 *If* $\Lambda(k) = \phi$, $\omega_s(k) < \min_{l \in \cup_{p \in S(k-1)} L_p} \eta_l(k)$, *for some* $s \in S(k-1)$.

Proof of Lemma 13: 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 $l_{\min}$.

$$\omega_j(k) \geq \eta_{l_{\min}}(k) \ \forall j \in S(k-1) \tag{50}$$

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

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

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

$$\eta_{l_{\min}}(k) \geq \min_{i \in \Xi_{l_{\min}}(k-1)} \lambda_{il_{\min}}(k-1) \tag{51}$$

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

$$\begin{aligned}
\eta_{i_{\min} l_{\min}}(k) &= \max(\eta_{l_{\min}}(k), \lambda_{i_{\min} l_{\min}}(k-1)) \\
&= \eta_{l_{\min}}(k) \text{ (from (51) and the definition of } i_{\min}). \quad (52) \\
&\leq \eta_l(k) \ \forall \ l \in L_s, \text{ (since } s \in m(i_{\min}, l_{\min}) \cap S(k-1), \\
&\qquad \text{and from the definition of } l_{\min}) \\
&\leq \eta_{i_{\min} l}(k) \ \forall \ l \in L_s \quad (53) \\
\text{Thus } \min_{l \in L_s} \eta_{i_{\min} l}(k) &= \eta_{i_{\min} l_{\min}}(k) \text{ (from } l_{\min} \in L_s \text{ and (53))} \\
\text{Thus } \omega_s(k) &= b \lfloor \frac{\eta_{i_{\min} l_{\min}}(k)}{b} \rfloor \text{ (since } s \in m(i_{\min}, l_{\min}) \cap S(k-1),
\end{aligned}$$

$$\omega_s(k) = b\lfloor \frac{\min_{l \in L_s} \eta_{i_{\min} l}(k)}{b} \rfloor)$$

$$\begin{aligned}
&= b\lfloor \frac{\eta_{l_{\min}}(k)}{b} \rfloor \text{ (from (52))} \\
&\leq \eta_{l_{\min}}(k) \qquad (54) \\
\text{Thus } \omega_s(k) &= \eta_{l_{\min}}(k) \text{ (from (50), (54) and since } s \in S(k-1)) \\
&= \max(\eta_{l_{\min}}(k), \lambda_{\chi(s) l_{\min}}(k-1)) \qquad (55) \\
&\qquad \text{(from (52) and since } \chi(s) = i_{\min}) \\
&\geq \Omega_{\chi(s) l_{\min}}(k) \text{ (from (41))} \\
\text{Also } \omega_s(k) &\leq \Omega_{\chi(s) l_{\min}}(k) \\
\text{Thus } \omega_s(k) &= \Omega_{\chi(s) l_{\min}}(k) \qquad (56)
\end{aligned}$$

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

$$\begin{aligned}
\Omega_{i l_{\min}}(k) &\geq \lambda_{i l_{\min}}(k-1) \text{ (from (15))} \\
&= \max(\eta_{l_{\min}}(k), \lambda_{i l_{\min}}(k-1)) \\
\text{Thus } \Omega_{i l_{\min}}(k) &= \max(\eta_{l_{\min}}(k), \lambda_{i l_{\min}}(k-1)) \text{ (from (41))} \qquad (57)
\end{aligned}$$

Now let $\max(\eta_{l_{\min}}(k), \lambda_{i l_{\min}}(k-1)) = \eta_{l_{\min}}(k)$. $\omega_j(k) = \max(\eta_{l_{\min}}(k), \lambda_{i l_{\min}}(k-1)),\ \forall j\ \in m(i, l_{\min}) \cap S(k-1)$ ($m(i, l_{\min}) \cap S(k-1) \neq \phi$ as $i \in \Xi_{l_{\min}}(k-1)$). The reasoning is exactly the same as that behind (55). $\Omega_{i l_{\min}}(k) = \max_{j \in m(i, l_{\min})} \omega_j(k) \geq \max(\eta_{l_{\min}}(k), \lambda_{i l_{\min}}(k-1))$, since $m(i, l_{\min}) \cap S(k-1) \neq \phi$. From (41),

$$\Omega_{i l_{\min}}(k) = \max(\eta_{l_{\min}}(k), \lambda_{i l_{\min}}(k-1)) \qquad (58)$$

From (57) and (58)

$$\Omega_{i l_{\min}}(k) = \max(\eta_{l_{\min}}(k), \lambda_{i l_{\min}}(k-1))\ \forall\ i \in \Xi_{l_{\min}}(k-1) \qquad (59)$$

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

$$\begin{aligned}
\sum_{i \in n(l_{\min}) \setminus \Xi_{l_{\min}}(k-1)} \Omega_{i l_{\min}}(k) &= \sum_{i \in n(l_{\min}) \setminus \Xi_{l_{\min}}(k-1)} \lambda_{i l_{\min}}(k-1) \\
&= F_{l_{\min}}(k-1) \qquad (60) \\
\sum_{i \in n(l_{\min})} \Omega_{i l_{\min}}(k) &= \sum_{i \in n(l_{\min}) \setminus \Xi_{l_{\min}}(k-1)} \Omega_{i l_{\min}}(k) + \sum_{i \in \Xi_{l_{\min}}(k-1)} \Omega_{il}(k)
\end{aligned}$$

$$\begin{aligned} &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (\text{since } \Xi_{l_{\min}}(k-1) \subseteq n(l_{\min})) \\ &= F_{l_{\min}}(k-1) + \sum_{i \in \Xi_{l_{\min}}(k-1)} \max(\eta_{l_{\min}}(k), \lambda_{il_{\min}}(k-1)) \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (\text{from (59) and (60)}) \\ &= C_l \qquad (\text{since } \Xi_{l_{\min}}(k-1) \neq \phi) \end{aligned} \tag{61}$$

Thus from (56) and (61) $s \in \Lambda(k)$, 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 14 *If $\Lambda(k) = \phi$, there exists a virtual session $s \in S(k-1)$ such that*

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

Proof of Lemma 14: Let $\Lambda(k) = \phi$. From Lemma 13, $\omega_s(k) < \min_{l \in \cup_{p \in S(k-1)} L_p} \eta_l(k)$, for some $s \in S(k-1)$. It follows that $\omega_s(k) < \min_{l \in L_s} \eta_l(k)$.

$$\begin{aligned} \min_{l \in \cup_{p \in S(k-1)} L_p} \eta_l(k) &> \omega_s(k) \\ &= b\lfloor \frac{\min_{l \in L_s} \eta_{\chi(s)l}(k)}{b} \rfloor \ (\text{since } s \in S(k-1)) \\ &\geq b\lfloor \frac{\min_{l \in L_s} \eta_l(k)}{b} \rfloor \\ \text{Thus } \lfloor \frac{\min_{l \in \cup_{p \in S(k-1)} L_p} \eta_l(k)}{b} \rfloor &\geq \lfloor \frac{\min_{l \in L_s} \eta_l(k)}{b} \rfloor \\ \lfloor \frac{\min_{l \in \cup_{p \in S(k-1)} L_p} \eta_l(k)}{b} \rfloor &\leq \lfloor \frac{\min_{l \in L_s} \eta_l(k)}{b} \rfloor \text{ since } s \in S(k-1), L_s \subseteq \cup_{p \in S(k-1)} L_p \\ \text{Thus } \lfloor \frac{\min_{l \in \cup_{p \in S(k-1)} L_p} \eta_l(k)}{b} \rfloor &= \lfloor \frac{\min_{l \in L_s} \eta_l(k)}{b} \rfloor \end{aligned} \tag{62}$$

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 (62)

$$\lfloor \frac{\eta_l(k)}{b} \rfloor = \lfloor \frac{\min_{l_1 \in \cup_{p \in S(k-1)} L_p} \eta_{l_1}(k)}{b} \rfloor \tag{63}$$

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

$$\begin{aligned}
\omega_j(k) &= b\lfloor \frac{\min_{l_1 \in L_j} \eta_{\chi(j)l_1}(k)}{b} \rfloor \text{ (since } j \in S(k-1)) \\
&\geq b\lfloor \frac{\min_{l_1 \in L_j} \eta_{l_1}(k)}{b} \rfloor \\
&\geq b\lfloor \frac{\min_{l_1 \in \cup_{p \in S(k-1)} L_p} \eta_{l_1}(k)}{b} \rfloor \text{ since } j \in S(k-1), L_j \subseteq \cup_{p \in S(k-1)} L_p \\
&= b\lfloor \frac{\eta_l(k)}{b} \rfloor \text{ (from (63)).} \qquad (64)
\end{aligned}$$

If session $i \in \Xi_l(k-1)$, there exists virtual session j s.t. $j \in m(i,l) \cap S(k-1)$. Thus from (64) and the fact that $\Omega_{il}(k) = \max_{j \in m(i,l)} \omega_j(k)$, it follows that $\Omega_{il}(k) \geq b\lfloor \frac{\eta_l(k)}{b} \rfloor \ \forall \ i \in \Xi_l(k-1)$. This holds for all links l for which $\lfloor \frac{\eta_l(k)}{b} \rfloor = \min_{l_1 \in L_s} \lfloor \frac{\eta_{l_1}(k)}{b} \rfloor$. The result follows. □

Lemma 15 *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.*

Proof of Lemma 15: 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 $s(k)$. $s(k) \in S(k-1)$. There exists a link $l \in L_{s(k)}$ such that $\omega_s(k) = b\lfloor \frac{\eta_{\chi s(k) l}(k)}{b} \rfloor$. We call such a link, $l(k)$. If more than one link satisfies the above property, $l(k)$ is chosen amongst them arbitrarily. Elements in T define a sequence of links $l(k)$. We show that this sequence has at most $|\mathcal{L}|M$ elements.

Let $\Theta_l(k)$ denote the residual capacity of link l, at the end of the kth iteration, i.e., $\Theta_l(k) = C_l - \sum_{i \in n(l)} \lambda_{il}(k)$. We will show that $\Theta_{l(k)}(k) - \Theta_{l(k)}(k-1) \geq b$. 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 iteraion 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 $j \in m(\chi(s), l(k))\backslash S(k-1), \omega_j(k) = r_j(k-1) \leq \lambda_{\chi(s(k))l(k)}(k-1) \leq \max(b\lfloor\frac{\eta_{l(k)}(k)}{b}\rfloor, \lambda_{\chi(s(k))l(k)}(k)(k-1)) = b\lfloor\frac{\eta_{\chi(s(k))l(k)}(k)}{b}\rfloor$. The last equality follows from the divisibility of $\lambda_{\chi(s(k))l(k)}(k)(k-1)$ by b (Lemma 6). Thus $\Omega_{\chi(s(k))l(k)}(k) \leq b\lfloor\frac{\eta_{\chi(s(k))l(k)}(k)}{b}\rfloor$. It follows that $\omega_{s(k)}(k) = \Omega_{\chi(s(k))l(k)}(k)$.

$$\begin{aligned}
\lambda_{\chi(s)l(k)}(k) &\geq r_{s(k)}(k) \\
&= \omega_{s(k)}(k) + b \\
&= \Omega_{\chi(s(k))l(k)}(k) + b \\
&\geq \lambda_{\chi(s(k))l(k)}(k-1) + b \text{ (from (15))} && (65)\\
\lambda_{il(k)}(k) &\geq \lambda_{il(k)}(k-1)\ \forall\ i \in n(l(k)) \text{(from Lemma 8)} && (66)\\
\sum_{i\in n(l(k))} \lambda_{il(k)}(k) &\geq b + \sum_{i\in n(l(k))} \lambda_{il(k)}(k-1) \text{ (from (65) and (66))} && (67)\\
\Theta_{l(k)}(k) &= C_{l(k)} - \sum_{i\in n(l(k))} \lambda_{il(k)}(k) \\
&\leq C_{l(k)} - \sum_{i\in n(l(k))} \lambda_{il(k)}(k-1) - b \text{ (from (67))} \\
&= \Theta_{l(k)}(k-1) - b && (68)
\end{aligned}$$

Now we show that $\Theta_{l(k)}(k) < Mb$. We will show that $\lfloor\frac{\eta_{l(k)}(k)}{b}\rfloor = \min_{l\in L_{s(k)}}\lfloor\frac{\eta_{l(k)}(k)}{b}\rfloor$. Since $r_s(k) = \omega_s(k) + b$, it would follow that $\Omega_{il(k)}(k) \geq b\lfloor\frac{\eta_{l(k)}(k)}{b}\rfloor\ \forall i \in \Xi_{l(k)}(k-1)$. From (15) $\Omega_{il(k)}(k) \geq \lambda_{il(k)}(k-1)$. Thus

$$\begin{aligned}
\Omega_{il(k)}(k) &\geq \max(b\lfloor\frac{\eta_{l(k)}(k)}{b}\rfloor, \lambda_{il(k)}(k-1)),\ \forall i \in \Xi_{l(k)}(k-1) && (69)\\
\Theta_{l(k)}(k) &= C_{l(k)} - \sum_{i\in n(l(k))} \lambda_{il(k)}(k) \\
&= C_{l(k)} - \left(F_l(k-1) + \sum_{i\in\Xi_{l(k)}(k-1)} \lambda_{il(k)}(k)\right) \\
&\leq C_{l(k)} - \left(F_l(k-1) + \sum_{i\in\Xi_{l(k)}(k-1)} \Omega_{il(k)}(k)\right) \text{ (since } \lambda_{il(k)}(k) \geq \Omega_{il(k)}(k)) \\
&\leq C_{l(k)} - \left(F_l(k-1) + \sum_{i\in\Xi_{l(k)}(k-1)} \max(b\lfloor\frac{\eta_{l(k)}(k)}{b}\rfloor, \lambda_{il(k)}(k-1))\right) \text{ (from (69))}
\end{aligned}$$

$$
\begin{aligned}
&< C_{l(k)} - \left(F_l(k-1) + \sum_{i \in \Xi_{l(k)}(k-1)} \max(\eta_{l(k)}(k) - b, \lambda_{il(k)}(k-1)) \right) \\
&\leq |\Xi_{l(k)}(k-1)|b + C_{l(k)} - \left(F_l(k-1) + \sum_{i \in \Xi_{l(k)}(k-1)} \max(\eta_{l(k)}(k), \lambda_{il(k)}(k-1)) \right) \\
&= |\Xi_{l(k)}(k-1)|b \\
&\leq Mb \qquad (70)
\end{aligned}
$$

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

$$
\begin{aligned}
\lfloor \frac{\eta_{l(k)}(k)}{b} \rfloor &\leq \lfloor \frac{\eta_{\chi(s)l(k)}(k)}{b} \rfloor \\
&= \frac{\omega_s(k)}{b} \\
&< \frac{\min_{l \in L_{s(k)}} \eta_l(k)}{b} \text{ since } r_s(k) = \omega_s(k) + b \qquad (71) \\
\text{We know that } \lfloor \frac{\eta_{l(k)}(k)}{b} \rfloor &\geq \lfloor \frac{\min_{l \in L_{s(k)}} \eta_l(k)}{b} \rfloor \text{ (since } l(k) \in L_{s(k)}) \\
\text{If } \lfloor \frac{\eta_{l(k)}(k)}{b} \rfloor &> \lfloor \frac{\min_{l \in L_{s(k)}} \eta_l(k)}{b} \rfloor \\
\text{then } \lfloor \frac{\eta_{l(k)}(k)}{b} \rfloor &\geq \frac{\min_{l \in L_{s(k)}} \eta_l(k)}{b} \qquad (72)
\end{aligned}
$$

But (72) contradicts (71). 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 2 (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 atmost M iterations, where $S(0) = \{1, \ldots, M\}$. Lemma 15 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 14 and the algorithm. Thus (1) holds in this case. □

References

[1] E. Amir, S. McCanne and M. Vetterli. A layered DCT coder for Internet video. *Proceedings of IEEE International Conference on Image Processing,* pages $13-16$, Lusanne, Switzerland, September, 1996.

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

[3] S. P. Abraham and A. Kumar. Max-Min Fair Rate Control of ABR Connections with Nonzero MCRs. *Proceedings of IEEE Globecom'* 97, pp $498-502$

[4] Y. Bernet, J. Binder, S. Blake, M. Carlson, B. Carpenter, S. Keshav, E. Davies, B. Ohlman, D. Verma, Z. Wang and W. Weiss. A framework for differentiated services, *Internet Draft,* February 1999.

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

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

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

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

[9] S. Chiung, M. Ammar and X. Li. On the use of Destination Set Grouping to Improve Fairness in Multicast Video Distribution *IEEE INFOCOM'*96

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

[11] F. Chiussi, Y. Xia and V. Kumar. Performance of Shared-Memory Switches under Multicast Bursty Traffic *IEEE Journal on Selected Areas In Communications*, Vol 15, No. 3, 1997.

[12] D. Cheriton and S. Deering. Host groups: A multicast extension for datagram internetworks. *Proceedings of the 9th Data Communications Symposium* (Sept. 1985), ACM/IEEE New York, 1985, 172-179

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

[14] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C.-G. Liu, and L. Wei. An architecture for wide area multicast routing, *Proceedings of ACM SIGCOMM,* London, UK, Aug. 1994, pp. 126-135.

[15] F. Kishino, K. Manabe, Y. Hayashi and H. Yasuda. Variable Bit-Rate Coding of Video Signals for ATM Networks, *IEEE Journal on Selected Areas In Communications*, Vol. 7, No. 5, June 1989.

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

[17] Y. T. Hou, H. H. Y. Tzeng and S. S. Panwar. A Generalized Max-Min Rate Allocation Policy and its Distributed Implementation Using the ABR Flow Control Mechanism, *Proceedings of IEEE Infocom' 98*, San Francisco, CA, March 1998

[18] International Business Machines Corporation. *Technical Reference PC Network.* Doc. 6322916

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

[20] X. Li, S. Paul and M. H. Ammar. Multi-session Rate Control for Layered Video Multicast *Technical Report* GT-CC-98−21, College of Computing, Georgia Institute of Technology, 1998

[21] J. Moy, Multicast routing extensions for OSPF, *Comm, ACM* vol 37, no. 8, pp 61-66, Aug 94.

[22] Sun Microsystems. *Remote Procedure Call Reference Manual.*Mountain View, California, Oct. 1984

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

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

[25] M. Parsa, J.J.Garcia-Luna-Aceves. A protocol for Scalable Loop-Free Multicast Routing, *IEEE Journal on Selected Areas In Communications*, Vol 15, No. 3, 1997.

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

[27] M. Satyanarayanan and F. Siegal. MultiRPC: A parallel remote procedure call mechanism. Tech Rep. CMU-CS-86-139, Carnegie-Mellon Univ., Aug. 1986

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

[29] S. Sarkar and L. Tassiulas: Fair Allocation of Resources in Multirate Multicast Trees *Technical Report* TR 99 − 42, Institute for Systems Research and University of Maryland, 1999.

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

[31] H. Y. Tzeng and K. Y. Siu. On Max-Min Fair Congestion Control for Multicast ABR Service in ATM, *IEEE Journal on Selected Areas In Communications*, Vol 15, No. 3, 1997.

[32] D. L. Tennenhouse and D. J. Wetherall. Towards an active network architecture. *Computer Communication Review,*, 26(2) : 5 − 18, April 1996.

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

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

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