Fair Allocation of Utilities in Multirate Multicast Networks: A Framework for Unifying Diverse Fairness Objectives

Saswati Sarkar and Leandros Tassiulas

***Abstract*—We study fairness in a multicast network. We assume that different receivers of the same session can receive information at different rates. We study fair allocation of utilities, where utility of a bandwidth is an arbitrary function of the bandwidth. The utility function is not strictly increasing, nor continuous in general. We discuss fairness issues in this general context. Fair allocation of utilities can be modeled as a nonlinear optimization problem. However, nonlinear optimization techniques do not terminate in a finite number of iterations in general. We present an algorithm for computing a fair utility allocation. Using specific fairness properties, we show that this algorithm attains global convergence and yields a fair allocation in polynomial number of iterations.**

I. Introduction

WE WILL STUDY resource allocation and congestion control in real time multicast networks in this paper. Multicast is communication between a single source and a group of destinations. The source sends one copy of a message and this copy is replicated only at the branching points of the multicast route. More traditional forms of communication like unicast with a single source and a single destination, and broadcast with a single source communicating to all other nodes in the network are special cases of multicast. Real time multicast applications are increasing all the time. Examples are collaborative applications like audio or video teleconferencing, video-on-demand services, distance learning, etc. Resource allocation in multicast networks is more complex than that in unicast and broadcast networks. This calls for a separate study of these problems in the multicast scenario.

Multicast congestion control is more challenging than usual unicast congestion control on account of network heterogeneity. A multicast application may have many destinations, and end systems can have widely varying bandwidth requirements. The paths to different destinations may have different bandwidth capacities, e.g., one may consist of multimegabit links, such as T3 (45 Mb/s), and another may have a 128 kb/s ISDN line. 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. This is not an issue in unicast networks because there is only one receiver per application. This can be countered in real time multicast applications using multirate transmission. Here, different receivers of the same application are allowed to receive information at different rates. For this purpose, the source signal is encoded into a number of layers that can be incrementally combined to provide progressive refinement. The source transmits all the layers. The receivers adapt to the individual bandwidth requirements and capabilities by adding and dropping layers. As each layer is added, there is an improvement of quality of the received signal and additional bandwidth is needed to transport the combined stream. When a layer is dropped, there is graceful degradation in the reception quality. This layered transmission scheme has been used for both video [19] and audio [3] transmissions over the Internet and has potentials for use in ATM networks as well [6]. Multirate transmission introduces its own challenges though. At every link, the transmission rate of a session is equal to that of the fastest session receiver downstream of the link. So the transmission rate of the same session may be different in different links depending on the configuration of the receivers downstream. Thus, unlike unicast networks, there is no concept of session rates as such. One needs to consider receiver rates separately, and congestion control schemes need to be modified suitably so as to consider the receivers separately.

Every receiver would like to receive as many layers as possible. However, networks have resource limitations, and delivering a large number of layers would cause acute congestion. Resource allocation strategies become important in this situation. Resource sharing objectives determine the individual allocations. We adopt fairness as our resource sharing objective. There are several possible notions of fairness. A simplistic notion is to serve all users at the same rate. However, some users may receive information through low bandwidth links, while some others may receive information through high bandwidth links. Serving all users at the same rate would tie all users down to the low quality service in this case. So this notion of fairness is not useful. We describe a few more sophisticated notions of fairness here, e.g., maxmin fairness [2], proportional fairness [7]. A bandwidth allocation is maxmin fair if it is not possible to respect the resource constraints and increase the bandwidth of a receiver without decreasing that of any other receiver which has equal or lower rate. Recently, [7] has advocated a new notion of fairness, namely proportional fairness. A feasible rate

Manuscript received September 30, 2000; revised June 10, 2001. Recommended by the Guest Editors W. Gong and T. Başar. The work of S. Sarkar was supported in part by the National Science Foundation under Grant ANI01-06984.

S. Sarkar is with the Department of Electrical Engineering, the University of Pennsylvania, Philadelphia, PA 19104 USA (e-mail: swati@ee.upenn.edu).

L. Tassiulas is with the Department of Electrical and Computer Engineering and Institute for Systems Research, University of Maryland, College Park, MD 20742 USA (e-mail: leandros@isr.umd.edu).

Publisher Item Identifier S 0018-9286(02)05500-9.

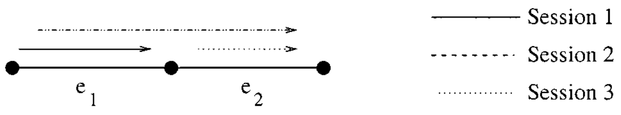

Fig. 1. Session 2 traverses both links, and sessions 1 and 3 traverse one link each. Both links have one unit of bandwidth each.

vector $\vec{r}^1$ is proportionally fair, if for any other feasible rate vector $\vec{r}^2$, the aggregate of proportional changes is nonpositive, i.e., $\sum_{i=1}^{M}(r_2^i - r_1^i)/(r_1^i) \leq 0$. Consider the network in Fig. 1 as a distinguishing example. Maxmin fair allocation allocates 0.5 units to all sessions. Proportionally fair allocation allocates $1/3$ unit of bandwidth to the long session and $2/3$ units of bandwidth to the short sessions each. So it is apparent that the proportionally fair allocation discriminates against long sessions. The argument in favor of this discrimination is that the long sessions consume more resources than the short ones, and hence should not be treated in the same manner as the short sessions. This argument raises the question that if we need to discriminate between sessions, then how should the amount of discrimination be determined. In our opinion, this should depend on several issues like the bandwidth requirements, revenues earned, etc. For example, a video source requires more bandwidth than an audio source, and these two should not be allocated equal resources. Also, certain users may be willing to pay higher revenues, and then it is fair that they get better quality service. Thus, it is important to generalize the notions of fairness so that one can suitably differentiate among users within the framework of fairness depending upon user requirements and network objectives. We generalize the concept of maxmin fairness, using the notion of utilities.

Utility of a receiver is a function connecting the bandwidth granted to the receiver and the "value" associated with the bandwidth. The term "value" is an abstract notion. It simply indicates a measure of the satisfaction of the user, or even the amount paid by the user. A possible fairness objective can be to allocate bandwidth so that the utilities are distributed fairly. We would consider fair allocation of utilities in the multirate, multicast scenario. Users need to announce their utility functions. The network would charge the users in accordance with their utilities, and at the same time, allocate the resources so that the corresponding utility allocation is fair. Consider a simple one link example to clarify this concept. A user who needs a high bandwidth, and is willing to pay high revenues will declare a slowly increasing utility function, whereas a user who is satisfied with a low bandwidth and does not want to pay high revenue, would declare a rapidly increasing utility function. The fair allocation of utilities in this case is to equalize the utilities of the two users, and the former would receive a higher bandwidth than the latter at this fair operating point.

Utility fairness provides a unified framework for diverse fairness objectives. We study a concrete example to argue this point. We have studied fair allocations of discrete bandwidth layers in [16] under the assumption that the layer structure is the same for different sessions. More precisely, layers for different sessions have the same bandwidth. Under this assumption, fairness of layer allocation is equivalent to the fairness of rate allocation, and the two need not be investigated separately. This does not hold any longer when different sessions have different layer bandwidth. Consider a specific example. Consider a single link network. Two sessions traverse the link. Source of the first session transmits unit bandwidth layers. Every layer of the second session consumes 2 units of bandwidth. Let the capacity of the link be 9 units. A bandwidth allocation allocating 5 units of bandwidth to the first session and 4 units to the second one is intuitively fair (discrete bandwidth layers do not allow allocation of 4.5 units each). However, this allocation gives session 1, 5 layers and session 2, 2 layers. Intuitively, (3, 3) is a "fairer" layer allocation. The corresponding bandwidth allocation, (3, 6) does not seem fair. So, fair allocation of layers and bandwidth are different objectives. The fairness objective depends on the specific requirements of particular networks, and can be decided on a case by case basis. However, these different fairness objectives fall under the same framework, if we consider the fair allocation of utilities, where utility of a bandwidth can be the bandwidth itself, or the number of layers obtained from the bandwidth. The former would lead to the rate fairness problem, and the latter to the fair layer allocation problem. Likewise, utility can be any other function of bandwidth as well.

We review the related work in this area. Well known network protocols for layered transmission, receiver-driven layered multicast (RLM) [11] and layered video multicast with retransmissions (LVMR) [8] do not handle fairness among sessions very well, when there are multiple sessions competing for bandwidth [9]. Fair allocation of utilities have not been investigated in depth in either unicast or multicast networks. Algorithms for computation of fair bandwidth have been proposed in [12], [15], [16]. These papers consider special cases of the utility functions only, e.g., $U(x) = x$ [12], [15] and $U(x) = b\lfloor x/b \rfloor$ [16] for all receivers. Cao *et al.* have presented an algorithm for maxmin fair allocation of utilities in the unicast case [4]. This paper considers strictly increasing utility functions only. None of the previous algorithms [4], [15], [16] attain fair allocations for more general utility functions [14] even in the unicast case. As we discuss later, in many cases of practical interest utility functions do not increase strictly and are more general than the functions considered before. Fairness issues differ significantly when utility functions are more general.

This paper is organized as follows. We discuss our assumptions on the utility functions and motivate the study for more general utility functions in Section II. We present our network model in Section III. We also introduce our fairness notion in this section. We show that if the utility functions do not increase strictly then usual fairness notions are not applicable. For example, maxmin fair allocation of utilities may not exist and the computation of a lexicographically optimal allocation turns out to be NP-hard in this case. We introduce a weaker notion of fairness, maximal fairness, and show that it has several intuitively appealing fairness properties. We present an algorithm for computing maximally fair allocation of utilities in Section IV. The algorithm does not use network specific properties, and is hence a fairly general purpose one. We prove that the algorithm yields a fair allocation of utilities in polynomial number of iterations in Section V. We conclude this paper in Section VI.

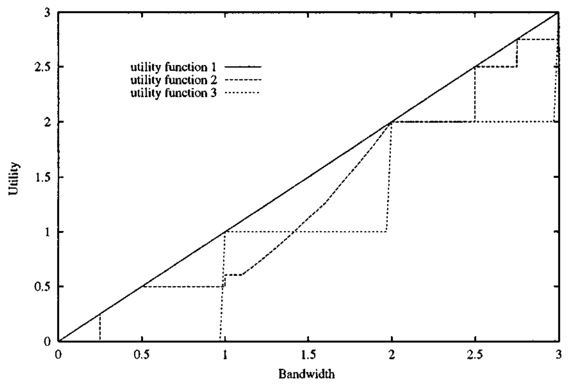

Fig. 2. Three utility functions. For utility functions 1 and 3, utility of bandwidth x is the actual bandwidth which can be allocated when x units of bandwidth are available and layered encoding is used. Utility function 1 models the case when the layer structure is completely flexible and any bandwidth can be allocated, if it is available. The utility function is linear and the utility set (set of allowable utilities) is the nonnegative real line. Utility function 3 models the case when the layer structure is completely predetermined. Only a discrete set of rates can be allocated and the utility function is stair-case. Utility set is the set of nonnegative integers. Utility function 2 models the case when the layer structure is flexible in certain ranges of bandwidth and predetermined in other ranges. Here, the utility is the qualitative gain from the bandwidth. In the flexible region, the function increases strictly, linearly or sublinearly. In the predetermined region, it is a stair-case function. Note that the step lengths are unequal, indicating that different layers of the same source have different bandwidth. The utility set consists of closed intervals and some discrete points.

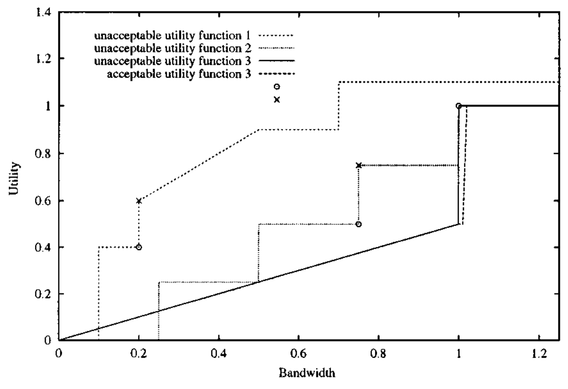

Fig. 3. This figure shows three unacceptable utility functions. The utility functions are not acceptable with data point "o." In each case, the utility function becomes acceptable if the data point 'o' is replaced by the one marked "x." The utility sets violate property 2) for functions 1 and 3, and property 3 for function 2.

II. Allocation of Utilities

We first discuss some possible utility functions and our assumptions on the utility functions. Utility of a bandwidth can be a measure of the reception quality a receiver obtains at that bandwidth. This measure can be objective as well as subjective. Objective measure can be quantities like information theoretic rate distortion, or the signal to noise ratio, where noise consists of the quantization noise. Obviously, these measures depend on the coding used, and hence it is important to allow different sessions to have different utility function, because different sessions may use different coding schemes. Other measures can be the number of layers obtained from a particular bandwidth. Subjective measures can be perceptual quality.

We define our assumptions on the utility functions now. These assumptions are mild, and a large number of utility functions satisfy these assumptions. Subsequently, we will present some sample utility functions, and motivate the requirement for assuming such general properties. Formally, "utility" is a function of the bandwidth. We assume that the receivers of the same session have the same utility function. "Utility set" of session i, A_i is the range of the utility function of session i. Receivers of a session i can possibly receive a utility if it belongs to the utility set of the session. For example, if utility of a bandwidth is the number of layers assigned out of the bandwidth, then the utility set is the set of nonnegative integers. If utility of a bandwidth is the bandwidth itself, then the utility set is the nonnegative real line. Refer to Fig. 2 for some example utility functions. We assume that the utility function satisfies the following technical assumptions for every i.

1) $A_i \cap (x, \infty) \neq \phi, \forall x \in R$.
2) Utility set, A_i is a closed set.
3) Given any utility x, there always exists a minimum bandwidth required for session i to attain utility x, i.e., the set $\{y : U_i(y) = x\}$ has a minimum, where $U_i(y)$ is the session i utility corresponding to bandwidth y. The minimum bandwidth for utility x is denoted $\Upsilon_i(x)$. Refer to Fig. 4 for some example minimum bandwidth functions, $\Upsilon_i(x)$.
4) The minimum bandwidth function Υ_i is right continuous and strictly increasing.

Fig. 2 shows some sample utility sets.

The first assumption indicates that the utility set is not upper bounded. If necessary, we can impose maximum utility constraints separately. The other assumptions are more technical. In the last assumption [Assumption 4)], continuity is intuitive because if the valid utilities in the utility set are very close, then the minimum bandwidth required for attaining them should also be very close. Strictly increasing property of the minimum bandwidth is also intuitive. However, the utility function may not increase strictly and the same utility can correspond to a range of bandwidth. Only, the minimum bandwidth required to attain any particular utility needs to increase strictly. This property implies that the utility functions must be nondecreasing. Hence, we do not make this assumption explicitly. The last three assumptions exclude some functions from being valid utility functions. Refer to Fig. 3 for examples of unacceptable utility functions. Any utility function satisfying properties 1)–4) is acceptable.

The assumptions on the utility structure are mild and most of the utility functions encountered in the networking context satisfy these assumptions. At the same time such general assumptions are needed for modeling several utility functions of practical importance. Consider a specific example to motivate the general assumptions. Let utility of a bandwidth $x, U_i(x)$ be the actual bandwidth obtained by a receiver i when x units of bandwidth are available for the receiver. Note that for layered transmission of signals $U_i(x)$ may not equal x as receivers subscribe to layers. Often times receivers can not receive a layer partially and hence the allowed bandwidth allocations form a discrete set. Thus, $U_i(x) \leq x$ in this case. Utility functions of Fig. 2 give a pictorial representation. Relation between the layers and the bandwidth determine the nature of the utility functions in this case, and the former depends on the encoding mechanism used. Depending on the coding mechanism, in certain bandwidth ranges, the layers can be fine tuned in accordance with the bandwidth requirements. This means that given the desired rates of various receivers, the source generates as many layers as distinct receiver rates and adjusts the layer bandwidth so as to match the receiver rates. The layer structure is said to be "flexible" in these ranges. The utility function representing the actual bandwidth obtained from a given available bandwidth will increase strictly, either linearly or sublinearly, for these bandwidth ranges. The layer structure may be predetermined in other bandwidth ranges, i.e., layer bandwidth cannot be tuned to match all the desired receiver rates and receivers can not partially subscribe to layers. The utility function will be stair case in these bandwidth ranges. The step sizes represent the bandwidth of different layers. Different layers may have different bandwidths, e.g., the codec of [10] generates nonuniform bandwidth layers. Hence, the step jumps may have unequal sizes. Thus, in general, the utility functions will neither be continuous nor strictly increasing and will have different shapes in different bandwidth ranges. Fairness under these assumptions is vastly different from the case in which the utility functions are strictly increasing. Thus it would be interesting from a mathematical as well as a practical standpoint to study this more general case.

Different sessions may have different encoding mechanisms, and hence, different layer structures. This leads to different utility functions for different sessions in the previous example. Thus utility functions will be different for different sessions in general. However, we assume that the receivers of the same session have the same utility functions. This assumption is reasonable, because utility structure (e.g., layer structure) often depends on source and there is only one source per session. Note that we only assume same utility functions for different receivers of the same session, but utilities allocated to different receivers of the same session can be different.

III. Network Model and Fairness Notions

We describe our network model in this section. We consider an arbitrary topology network with N multicast sessions. A multicast session is identified by the pair (v, U), where v is the source node of the session and U is the group of intended destination nodes. We assume that the traffic from node v is transported across a predefined multicast tree to nodes in U. The tree can be established during connection establishment phase if the network is connection oriented or can be established by some well known multicast routing protocol like DVMRP [5], CBT [1], etc., in an Internet-type network.

We call every source destination pair of a session a virtual session. For example, if a session n has source v_n and destination set U_n, where $U_n = \{u_{n1}, \dots, u_{nt}\}$, then this session would correspond to t virtual sessions, $(v_n, u_{n1}), \dots, (v_n, u_{nt})$. To ensure fairness in a multirate network, we need to consider fair utility allocation for the virtual sessions separately, instead of those for the overall sessions.

Let r_{ij} denote the utility allocated to the jth virtual session of the ith session. Let there be M virtual sessions in the entire network. A utility allocation vector $\vec{r}$ is an M-dimensional vector, with components r_{ij} (i.e., a utility allocation vector gives the utility allocations for all virtual sessions in the network). For simplicity, we will use only one set of indexes for the virtual sessions, i.e., $(r_1, \dots, r_M)$ is a utility allocation vector. An M-dimensional vector $\vec{r} = (r_1, \dots, r_M)$ is a feasible utility allocation if the following hold true.

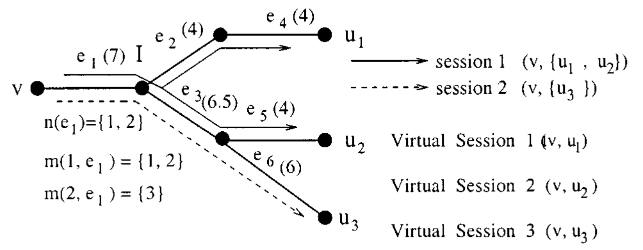

Fig. 4. The numbers in the brackets, ( ) denote the capacities of the respective links, e.g., capacity of link e_1 is 7 units. Let $U_1(x) = \lfloor x - 1 \rfloor, U_2(x) = x$. It follows that $\Upsilon_1(x) = x + 1$, and $\Upsilon_2(x) = x$, for $x > 0, \Upsilon_1(0) = \Upsilon_2(0) = 0$. The capacity constraint for link e_1 is $\max(r_1, r_2) + 1 + r_2 \leq 7$, where r_i is the utility allocated to virtual session i. Also, $\lambda_{1e_1} = \max(r_1, r_2)$ and $\lambda_{2e_2} = r_2$..

1) The utility allocated to every receiver belongs to the utility set, i.e., $r_s \in A_{\chi(s)}, \forall s \in \{1, \dots, M\}$, where $\chi(s)$ is the session of virtual session s.
2) $\mu_i \leq r_i \leq p_i \; \forall i$, where μ_i and p_i are respectively the minimum and maximum utiliities of virtual session $i, p_i \geq \mu_i \geq 0$.
3) Total bandwidth consumed in every link is less than or equal to the capacity of the link. Bandwidth consumed in a link is the sum of the bandwidth consumed by the different sessions traversing the link. Bandwidth consumed by a session in a link is the maximum of the bandwidth consumed by the virtual sessions of the session traversing the link, i.e.,

$$\sum_{i \in n(l)} \max_{j \in m(i,l)} \Upsilon_i(r_j) \leq C_l \text{ or equivalently}$$

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

where $\lambda_{il} = \max_{j \in m(i,l)} r_j, n(l)$ denotes the set of sessions traversing link $l, m(k, l)$ denotes the set of virtual sessions of session k passing through link l, C_l denotes the capacity of link l and $\Upsilon_i(x)$ denotes the minimum bandwidth required by session i to attain utility x. The quantity λ_{il} will be called the *session link utility*, where session link utility is the maximum utility allocated to virtual sessions of the same session traversing the link. Fig. 4 presents some example capacity constraints.

We define the fairness notions for the utility allocations. A feasible utility allocation is maxmin fair if it is not possible to maintain feasibility and increase the utility of a virtual session without decreasing that of any other virtual session which has equal or lower utility. More formally, a feasible utility allocation vector $\vec{r}^1$ is maxmin fair if it satisfies the following property with respect to any other feasible utility allocation vector $\vec{r}^2$: if

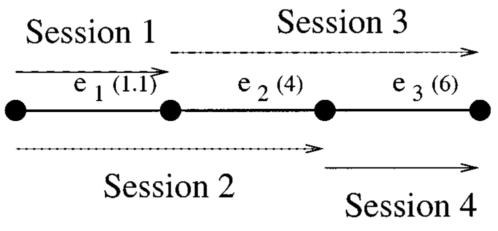

Fig. 5. This figure shows an example network which does not have a maxmin fair utility allocation. The numbers in the brackets denote the capacities of the corresponding links as in Fig. 4. Here, $U_i(x) = \lfloor x \rfloor$ for all i. This utility function arises in practice when the utility of bandwidth is the number of layers corresponding to the bandwidth, and the bandwidth of each layer is 1 unit. There are only a finite number of feasible utility allocations. None of these are maxmin fair. For each of these vectors, it is possible to maintain feasibility and increase the utility of one session, possibly decreasing the utility of another session with a higher utility. For example, consider the feasible utility allocation (1, 0, 3, 3). It is possible to increase the utility of session 2, by decreasing the utility of session 1. However session 1 has higher utility than session 2 in this allocation.

there exists i such that the ith component of $\vec{r}^2$ is strictly greater than that of $\vec{r}^1$ $(r_i^2 > r_i^1)$, then there exists j such that the jth component of $\vec{r}^1, r_j^1$ is less than or equal to the ith component of $\vec{r}^1, r_i^1$ $(r_j^1 \leq r_i^1)$ and the jth component of $\vec{r}^2$ (r_j^2) is strictly less than the jth component of $\vec{r}^1$ $(r_j^2 < r_j^1)$. A maxmin fair utility allocation may not exist in general for non strictly increasing utility functions. Refer to Fig. 5 for an example.

We will consider the notion of lexicographic optimality. Given an M-dimensional vector $\vec{V}$, define its lexicographically ordered version $\hat{V}$ as follows: $\hat{v}_j = v_k$ for some k and $\hat{v}_1 \leq \hat{v}_2 \leq \cdots \hat{v}_M$. In other words, components of $\hat{V}$ are an ordered version of those of $\vec{V}$. A vector $\vec{v}^1$ is lexicographically greater than another vector $\vec{v}^2$ if there exists i such that $\hat{v}_i^1 > \hat{v}_i^2$ and $\hat{v}_j^1 = \hat{v}_j^2$ if $j < i$. Vectors $\vec{v}^1$ and $\vec{v}^2$ are lexicographically equal if $\hat{v}^1 = \hat{v}^2$. Vector $\vec{v}^1$ is lexicographically less than $\vec{v}^2$ if $\vec{v}^2$ is lexicographically greater than $\vec{v}^1$. A feasible utility allocation $\vec{v}$ is lexicographically optimal if every feasible utility allocation vector is lexicographically less than or equal to $\vec{v}$. Informally, a utility allocation is lexicographically optimal, if its smallest component is the largest amongst the smallest components of all feasible vectors, subject to this it has the largest second smallest component and so on. For example, (1, 0, 3, 3) is a lexicographically optimal utility allocation in Fig. 5.

Lexicographic optimality and maxmin fairness are different concepts, though several researchers have used these interchangeably. Fig. 5 illustrates the difference. Maxmin fair utility allocation does not exist in the network of Fig. 5, but lexicographically optimal allocation exists, and is equal to (1, 0, 3, 3). We have shown in [14] that if a vector is maxmin fair in a feasible set, then it is lexicographically optimal. The important difference is that a lexicographically optimal allocation always exists in any closed, bounded set, but a maxmin fair allocation may not exist in such a set. The feasible set of utility allocations is closed [property 2)] and bounded (from capacity constraint). Thus a lexicographically optimal utility allocation vector always exists. However, its computation is NP-hard in general. It has been shown in [16] and [17] that the computation of lexicographic optimal utility allocation is NP-hard if the utility function is $U_i(x) = \lfloor x \rfloor$ for each session i. This utility function arises in practice when the utility of the bandwidth is the number of layers corresponding to the bandwidth, and the bandwidth of each layer is 1 unit.

So we revert to a weaker definition for fairness, maximal fairness. We first define the concept of *relative fairness* of utility vectors. This concept has been introduced in [13]. A utility allocation vector $\vec{r}^1$ is *fairer* than another utility allocation vector $\vec{r}^2$, if for every virtual session i which has greater utility under $\vec{r}^2$ than under $\vec{r}^1$, there is some other virtual session j whose utility was already no more than that of i under $\vec{r}^1$, and has been decreased further by $\vec{r}^2$. A more formal definition of relative fairness follows. A utility allocation vector $\vec{r}^1$ is *fairer* than another utility allocation vector $\vec{r}^2$ if

- $\vec{r}^1 \neq \vec{r}^2$;
- there exists an i such that $r_i^1 < r_i^2$, then there exists a j such that $r_j^1 \leq r_i^1$ and $r_j^2 < r_j^1$ (r_k^l is the kth component of $\vec{r}^l, l = 1, 2$).

A utility allocation vector $\vec{r}$ is maximally fair if it is feasible and if no other feasible utility allocation vector is fairer than $\vec{r}$.

As we show later, maximally fair utility allocation exists and can be computed in polynomial complexity. However, first we argue that maximal fairness is a good notion of fairness. Note that from the definition of maximal fairness, it is clear that if a utility allocation is maximally fair, then we can increase the utility of a receiver s, only by decreasing that of another receiver to a value below that of the new utility value of receiver s. Thus we can increase the utility of s only by being unfair to some other receiver. In other words, if $\vec{A}$ is a maximally fair utility allocation and $\vec{B}$ is a feasible utility allocation different from $\vec{A}$, then $\vec{B}$ is "unfair" to some component as compared to $\vec{A}$. More formally, $\vec{B}$ is not fairer than $\vec{A}$. Also, if maxmin fair utility allocation exists, then it is the only maximally fair utility allocation. This follows because by definition of relative fairness, the maxmin fair utility allocation (if one exists) is fairer than any other feasible utility allocation. We have proved this in [17]. So any algorithm for computation of a maximally fair utility allocation will yield a maxmin fair utility allocation, if one exists. Besides, any lexicographically optimal vector is maximally fair in any feasible set (refer to [14] for proof) and hence lexicographically optimal utility allocation belongs to the set of maximally fair ones. Thus, maximal fairness is a good notion of fairness.

We would like to point out that it is not obvious that a maximally fair utility allocation can be computed in polynomial complexity. Consider the simple case of strictly concave utility functions. Maxmin fair allocation exists in this case, and hence as observed before any algorithm for computing a maximally fair allocation will compute a maxmin fair allocation. Note that maxmin fair utility allocation can be computed by a series of concave optimizations with nonlinear feasible sets [14]. Standard nonlinear optimization theory does not guarantee computation of the optimum in a finite number of steps. The situation becomes more complicated when the utility functions are more general, as we are considering here (note that in our case utility functions need not be continuous and can have discrete jumps) and thus it is not obvious that a maximally fair utility allocation can be computed in polynomial complexity. However, we will exploit the specific properties of these optimizations and the nature of the feasible sets to develop a polynomial complexity algorithm.

We conclude this section with a few observations.

There exists a necessary and sufficient condition for a utility allocation to be maximally fair. A utility allocation is maximally fair iff every virtual session has a "generalized bottleneck link" on its path. We will introduce the notion of a "generalized bottleneck" link later in this paper. There are similar concepts of bottleneck links for the special cases studied in [15] $(U(x) = x)$ and [16] $(U(x) = b\lfloor x/b \rfloor)$. However, the bottleneck condition is more complicated in this general case. We use the term "generalized bottleneck" in order to distinguish with the different bottleneck conditions for the special cases.

This observation will illustrate the scope of fair allocation of utilities. In general, all bandwidth allocations which satisfy the capacity constraints in the links are not interesting. The interesting ones are those in which no component can be increased without decreasing another component[1] (similar to the notion of pareto optimality). We denote these as the "boundary allocations." We have shown in [18] that given any boundary allocation there exists a choice of linear utility functions such that the boundary allocation corresponds to the maxmin fair utility allocation for the chosen utility functions. The important point is that the system can be tuned to any operating point in the feasible region by choosing the utility functions appropriately, and the operating point can be allocations which satisfy other fairness notions like proportional fairness for example.

Maximum utility constraints can be incorporated by adding artificial links between receivers with maximum utility constraints and the rest of the network. Capacity of such an artificial link is equal to the bandwidth consumed by the maximum utility of the respective receiver. So, henceforth, we shall ignore the maximum utility constraints.

IV. Algorithm for Computation of Maximally Fair Utility Allocation

We present an algorithm for computing a maximally fair utility allocation in this section. First, we will briefly describe the essential idea behind the algorithm, and then present the details. Subsequently we will present an example to illustrate the functioning of the algorithm. We conclude this section with analytical performance results. We first define two functions.

Ψ_i is a real valued function defined as $\Psi_i(x) = \inf_{y \in A_i \cap (x,\infty)} y$, $\forall i$. Loosely speaking, $\Psi_i(x)$ is the smallest session i utility greater than x. If the utility set of session i, is a set of integers, then $\Psi_i(x) = \lfloor x \rfloor + 1$. This function is well defined by property 1) of the utility set.

$\Phi_i(x)$ is a real valued function defined as $\Phi_i(x) = \max_{y \in A_i \cap [0,x]} y$. Thus, $\Phi_i(x)$ is the largest session i utility not exceeding x. If the utility set of session i, is a set of integers, then $\Phi_i(x) = \lfloor x \rfloor$. This function is well defined by property 2) of the utility set.

[1]The component decreased may have a value greater than, equal to, or less than the component increased.

The maximally fair utilities are computed via an iterative procedure. The algorithm classifies every virtual session as either saturated or unsaturated. Initially, all virtual sessions are unsaturated, and their status progressively change from unsaturation to saturation. A virtual session is saturated when it satisfies certain saturation conditions described later. It turns out that when a virtual session satisfies these saturation conditions, its utility allocation is maximally fair. The utility allocation does not change after the virtual session saturates. The algorithm terminates when all virtual sessions are saturated.

At every iteration k, the algorithm computes a fair utility for every virtual session progressively. At the beginning of an iteration, every link l computes a "fair share" for every session i traversing the link ("session link parameter," $\eta_{il}(k)$), ignoring the constraint imposed by the utility structure and the bandwidth restrictions of other links on the path of the session. These constraints are imposed progressively. For the fair share computation, the link first computes a quantity called the link control parameter $(\eta_l(k))$ as per step (3) of the algorithm, and then computes the session link parameter $\eta_{il}(k)$ as the maximum of the link control parameter and the utility allocated to session i in link l in the previous iteration, $\lambda_{il}(k-1)$. Next, the utility structure of session i is considered. Since receivers of session i can only receive utilities in utility set A_i, utility $\Phi_i(\eta_{il}(k))$ is offered to all virtual sessions of session i traversing the link. Note that $\Phi_i(\eta_{il}(k)) \leq \eta_{il}(k)$, for all i, l, k. Subsequently, the bandwidth restrictions of other links are considered as follows. A virtual session s is allocated utility $\omega_s(k)$, the minimum utility offered on its path.

The algorithm subsequently checks the saturation condition for each unsaturated virtual session under the utility allocation $\vec{\omega}(k)$. If no virtual session saturates in the current iteration, then the residual bandwidth is used to increment the utility of some virtual session s traversing this link and satisfying certain properties mentioned in step 8) of the algorithm. The utility of such a virtual session $s, \omega_s(k)$ is increased to the next higher value $\Psi_{\chi(s)}(\omega_s(k))$. 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. The new utility allocation of virtual session s is $r_s(k)$.

The algorithm terminates if all the virtual sessions are saturated under the utility allocation $\vec{r}(k)$, otherwise, there must be at least one more iteration. In the latter case, the algorithm makes computations which are used in the next iteration. We describe them now. 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 in the next iteration using this residual capacity as per step 3) of the algorithm. We introduce some notations next. Subsequently, the algorithm will be described formally.

The minimum session link utility, μ_{il} is the maximum of the minimum utilities of session i virtual sessions traversing l, i.e., $\mu_{il} = \max_{s \in m(i,l)} \mu_s$.

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

(Saturation conditions) A virtual session s is saturated w.r.t. a utility vector $\vec{r}$ if it traverses a link l in which

$$\begin{aligned} & r_s = \lambda_{\chi(s)l} \\ & \text{If } r_s = \Psi_{\chi(s)}(r_s), \quad \sum_{i \in n(l)} \Upsilon_i(\lambda_{il}) = C_l \\ & \text{If } r_s < \Psi_{\chi(s)}(r_s), \quad \sum_{i \in n(l)} \Upsilon_i(\lambda_{il}) \\ & \qquad > C_l - \left(\Upsilon_{\chi(s)}\left(\Psi_{\chi(s)}(r_s)\right) - \Upsilon_{\chi(s)}(r_s)\right). \end{aligned}$$

A session is *saturated* in 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.

$\eta_{il}(k)$ denotes the session link parameter of session i traversing link l.

$\omega_s(k)$ is the utility assigned to virtual session s in step 4) of the algorithm. It is the minimum utility offered on the path of virtual session s.

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

$r_s(k)$ is the utility allocated to virtual session s at the end of the kth iteration. The value of $r_s(k)$ can either be equal to or greater than $\omega_s(k)$. Allocation $\vec{r}(k)$ denotes the utility vector at the end of the kth iteration, with components $r_s(k)$.

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

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

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

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

$\Lambda(k)$ is the set of virtual sessions in $S(k-1)$ which are saturated w.r.t. utility allocation $\vec{\omega}(k)$ (the set of virtual sessions which were unsaturated earlier but saturates w.r.t. utility allocation $\vec{\omega}(k)$)

The algorithm follows.

1) $k = 0, \eta_l(0) = 0, F_l(0) = 0, \Xi_l(0) = n(l)\ \forall$ link $l, S(0) = \{1, \ldots, M\}, r_j(0) = \mu_j, \forall j \in S(0), \lambda_{il}(0) = \max_{j \in m(i,l)} r_j(0)$.
2) $k \rightarrow k+1$.
3) For every link l in the network compute the link control parameter. For this, first compute a variable $\alpha_l(k)$. If $\Xi_l(k-1) \neq \phi$, then $\alpha_l(k)$ is computed as follows:

$$\begin{aligned} \alpha_l(k) = \sup_{\theta}\{\theta : F_l(k-1) & \\ + \sum_{i \in \Xi_l(k-1)} & \Upsilon_i(\max(\Phi_i(\theta), \lambda_{il}(k-1))) \leq C_l\} \end{aligned}$$

(It can be shown that the above set is nonempty for all k.)

If $F_l(k-1) + \sum_{i \in \Xi_l(k-1)} \Upsilon_i(\max(\Phi_i(\alpha_l(k)), \lambda_{il}(k-1))) \leq C_l$, then link control parameter for a link l is $\eta_l(k) = \alpha_l(k)$, else link control parameter, $\eta_l(k)$ is chosen s.t. $\eta_l(k) < \alpha_l(k)$, and $\eta_l(k)$ satisfies the following conditions:

a) $\eta_l(k) \geq \max(\eta_l(k-1), \min_{i \in \Xi_l(k-1)} \lambda_{il}(k-1))$;
b) if session $i \in \Xi_l(k-1)$ and $\lim_{\theta \uparrow \alpha_l(k)} \max(\Phi_i(\theta), \lambda_{il}(k-1)) < \max(\Phi_i(\alpha_l(k)), \lambda_{il}(k-1))$, then $\eta_l(k) \geq \lambda_{il}(k-1)$ and $(\eta_l(k), \alpha_l(k)) \cap A_i = \phi$.

For all unsaturated sessions i passing through link l, $(i \in \Xi_l(k-1)), \eta_{il}(k) = \max(\eta_l(k), \lambda_{il}(k-1))$.

4) Compute $\omega_s(k)$ for all virtual sessions $s \in S$, where $\omega_s(k) = \Phi_{\chi(s)}(\min_{l \in L_s} \eta_{\chi(s)l}(k))$, 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 $n(l)$ as $\Omega_{il}(k) = \max_{s \in m(i,l)} \omega_s(k)$.
6) Compute the set of virtual sessions in $S(k-1)$ which saturate during the kth iteration under utility vector $\vec{\omega}(k), \Lambda(k)$. (refer later in the paper for saturation conditions).
7) If $\Lambda(k) \neq \phi$, compute the utilities 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 $s \in S(k-1)$ such that

$$\begin{aligned} & \omega_s(k) = \Phi_{\chi(s)}\left(\min_{l \in L_s} \eta_l(k)\right), \\ & \Psi_s(\omega_s(k)) > \omega_s(k) \end{aligned}$$

for some $l \in L_s$ s.t. $\Phi_{\chi(s)}(\min_{l_1 \in L_s} \eta_{l_1}(k)) = \Phi_{\chi(s)}(\eta_l(k))$

$$\begin{aligned} & \omega_s(k) = \Omega_{\chi(s)l}(k) \quad \text{and} \\ & \mathrm{Ran}(\Phi_{\chi(s)}) \cap (\eta_l(k), \alpha_l(k)) = \phi \end{aligned}$$

and for all $l \in L_s$ s.t. $\Phi_{\chi(s)}(\min_{l_1 \in L_s} \eta_{l_1}(k)) = \Phi_{\chi(s)}(\eta_l(k))$

$$\Omega_{il}(k) = \max(\Phi_i(\eta_l(k)), \lambda_{il}(k-1)) \quad \forall i \in \Xi_l(k-1).$$

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

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

9) For every link l in the network compute the session link utility in link l, for every session in $n(l)$ as $\lambda_{il}(k) = \max_{s \in m(i,l)} r_s(k)$.
10) Compute the set of virtual sessions in $S(k-1)$ which saturate w.r.t. utility allocation $\vec{r}(k)$. Remove these sessions from $S(k-1)$, and the resulting set is $S(k)$. (Refer to later in the paper for saturation conditions).
11) If $S(k) = \phi$, i.e., all virtual sessions are saturated, 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, $\Xi_l(k)$.
13) For every link l, for which $\Xi_l(k) \neq \phi$, compute the bandwidth consumed by the saturated sessions passing through link $l, F_l(k) = \sum_{i \in \Xi_l(k)} \Upsilon_i(\lambda_{il}(k))$.

14) Go to step 2).

We first present an illustrative example. Consider the network of Fig. 5. All sessions are unicast, and hence, we will consider utilities of the sessions instead of the virtual sessions (virtual sessions and sessions are equivalent in this case). Assume that the utility function of session 1 is $U_1(x) = x, x \leq 0.5, U_1(x) = 0.1\lfloor 10x \rfloor, x > 0.5$. Also, $U_2(x) = 0.2\lfloor 5x \rfloor, U_3(x) = U_4(x) = \lfloor x \rfloor$. The sessions do not have minimum utility requirements. Now, $\alpha_l(1)$ is 0.6, 2, 3 for $l = e_1, e_2, e_3$ respectively. Following the steps in the computation of $\eta_l(1)$ from $\alpha_l(1), \eta_l(1) = \alpha_l(1)$ for $l \in \{e_2, e_3\}$. However, $\eta_{e_1}(1) < \alpha_{e_1}(1)$. Note that step (3) allows any choice of $\eta_{e_1}(1)$ in the interval [0.5, 0.6). We choose $\eta_{e_1}(1) = 0.5$. It follows that $\vec{\omega}(1) = (0.5, 0.4, 2, 3)$. None of the sessions are saturated under the allocation $\vec{\omega}(1)$. Both sessions 1 and 2 satisfy conditions for incrementation in step 8). We choose session 2 arbitrarily. Thus $r_2(1) = 0.6, r_i(1) = \omega_i(1), i \in \{1, 3, 4\}$. Sessions 1 and 2 saturate under the allocation $\vec{r}(1) = (0.5, 0.6, 2, 3)$. Note that $F_{e_2}(1) = 0.6$. In iteration 2, $\alpha_{e_2}(2) = 4, \alpha_{e_3}(2) = 3$. It follows that $\eta_l(2) = 3$ for $l \in \{e_2, e_3\}$. Note that $\eta_{e_2}(2)$ can be selected arbitrarily in the interval 3, 4), we choose 3. It follows that $\omega_i(2) = 3, i = 2, 3$. Sessions 3 and 4 saturate now. All sessions are now saturated under the allocation $\vec{\omega}(2) = (0.5, 0.6, 3, 3)$. Thus, the algorithm terminates with the utility allocation $\vec{r}(2) = (0.5, 0.6, 3, 3)$.

Next we present analytical results which show that the algorithm generates a maximally fair utility allocation in a polynomial number of iterations.

Theorem 1 (Maximal Fairness): The output utility allocation vector is

1) maximally fair;
2) maxmin fair, if a maxmin fair utility allocation exists.

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

We prove these in the next section. We would like to point out that the termination result holds independent of the nature of the utility functions. The utility functions can have step jumps, and the size of the step jumps are allowed to be arbitrarily small. The number of iterations required in this case is still upper bounded by $M + |\mathcal{L}|M^2$. Thus the upper bound increases polynomially with the number of links and virtual sessions. This is somewhat surprising as it appears from the algorithm that the utility of a virtual session may increase one step at a time, and as such it is not even clear that the algorithm terminates in finite number of iterations as the utility functions may have arbitrarily large number of step jumps in any bandwidth range. However, the intuition behind the result is that the utility of a virtual session s can also increase by several steps depending on its utility function and the utility functions of the other sessions sharing links with virtual session s, and this is the case when utility functions may have arbitrarily large number of step jumps in any bandwidth range. Hence, the number of iterations is finite, and actually, the polynomial in the number of links and the virtual sessions. We prove this more formally later.

Every step of this algorithm has a complexity of $O(\kappa|\mathcal{L}|M)$, where κ depends on the utility functions. If we assume κ to be a constant, then the complexity of every step is $O(|\mathcal{L}|M)$. The algorithm must terminate in $M + |\mathcal{L}|M^2$ iterations. Thus, the overall complexity of this algorithm is $O(|\mathcal{L}|^2M^3)$. The constants will depend on the computation of the limits for the utility functions. We conclude this section with a description of certain salient features of this algorithm.

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

We compare and contrast this algorithm to those for computing the fair allocations for the special cases addressed in [15], [16]. The algorithms presented in these are special cases of the current algorithm and follow the same general structure. There are several important differences though. We point them out now. We first consider the case for the utility function $U(x) = b\lfloor x/b \rfloor$ treated in [16]. The computation of the link control parameters differs from [16]. To accommodate the general nature of the problem, we need an additional quantity $\alpha_l(k)$ to compute the link control parameter $\eta_l(k)$ of link l in this case. There is no unique choice of the link control parameter, and among the possible choices, we need to choose a value which is at least as large as that of the previous iteration. There is a unique choice for the link control parameter in the algorithm of [16], and this choice ensures that the link control parameter does not decrease with subsequent iterations. The saturation conditions differ in the two algorithms. Also, selection of the virtual session for utility incrementation differs in the two algorithms. All these distinctions also apply to another special case $U(x) = x$, studied in [15]. In addition, every receiver just receives the minimum session link control parameter on its path in [15], and the utility incrementation step is not required there. The utility incrementation step is crucial for convergence here.

This algorithm terminates in fewer iterations in some special cases. For example, if the utility function is $U(x) = x$ for all sessions then the algorithm needs at most M iterations. If the utility functions are such that the ratio between the maximum and minimum step jumps are bounded by a constant c, then the algorithm needs at most $c|\mathcal{L}|M$ iterations. The utility function $U(x) = b\lfloor x/b \rfloor$ falls under this category. We have not made any assumptions on the step jump sizes of the utility functions here.

V. Proof of Theorems 1 and 2

We prove that the algorithm presented in Section IV outputs a maximally fair utility allocation in a finite number of iterations (Theorems 1 and 2). At first, we present some properties of the intermediate utility allocations of this algorithm. We will use these in proving both the theorems. The second subsection will prove Theorem 1 and the last will prove Theorem 2. On account of space constraints, we will omit some of the proofs. Refer to [14] and [18] for details.

A. General Assumptions and Properties of the Algorithm

We will present some properties of the intermediate utility allocations in this subsection. We first present some properties of the Φ and the Ψ functions. Recalling the definitions, $\Phi_i(x)$ and $\Psi_i(x)$ are real valued functions defined as $\Phi_i(x) = \max_{y \in A_i \cap [0,x]} y, \Psi_i(x) = \inf_{y \in A_i \cap (x,\infty)} y$, for session i. Intuitively, $\Phi_i(x)$ is the largest session i utility not exceeding x, and $\Psi_i(x)$ is the smallest session i utility greater than x. If the utility set of session i, is a set of integers, then $\Phi_i(x) = \lfloor x \rfloor$, then $\Psi_i(x) = \lfloor x \rfloor + 1$. Functions Φ_i and Ψ_i satisfy the following properties:

1) Φ_i and Ψ_i are nondecreasing functions for all i;
2) $\Phi_i(x) \leq x, \Psi_i(x) \geq x$;
3) $\mathrm{Ran}(\Phi_i) \cap (\Phi_i(x), x] = \phi$.

We will use these properties in our proofs.

Now we present Lemma 1 which we will use in proving Theorems 1 and 2. More precisely, we will use this result in proving that all intermediate utility allocations are feasible (Lemma 4) and a termination result (Lemma 8). Lemma 1 shows that the utilities assigned by the maximal fairness algorithm can not decrease in subsequent iterations. The result follows from the steps of the algorithm.

Lemma 1: For all iterations $k \geq 0$, and virtual sessions $s, r_s(k+1) \geq r_s(k)$ and $\omega_s(k+1) \geq r_s(k)$. For all sessions i and links $l, \lambda_{il}(k+1) \geq \lambda_{il}(k)$ if $k \geq 0$.

Finally, we present a relation which we will use repeatedly

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

B. Proof of the Maximal-Fairness Theorem

We will prove that the output of the algorithm is maximally fair (Theorem 1). We will use Lemma 1 stated in the first subsection and some additional lemmas stated below. We summarize the proof as follows. First, we will show that the output of every iteration is a feasible utility vector (Lemma 4). Next, we will introduce the notion of a generalized bottleneck link. A feasible utility allocation is maximally fair if and only if every virtual session has a generalized bottleneck link. We prove the sufficiency in the generalized-bottleneck lemma (Lemma 5). The last step is to show that whenever a virtual session saturates, it has a generalized bottleneck link in every subsequent iteration (Lemma 6). The maximal fairness of the output follows.

Now we show the feasibility result (Lemma 4). First, we state Lemmas 2 and 3, which we will use in the proof of Lemma 4. Lemma 2 establishes an intuitive property, i.e., the utilities allocated to the virtual sessions at the end of every iteration belong to the respective utility sets.

Lemma 2: For all virtual sessions s and iterations $k, \Phi_{\chi(s)}(r_s(k)) = r_s(k), \Phi_{\chi(s)}(\omega_s(k)) = \omega_s(k)$. For all sessions i, link l and iterations $k, \Phi_i(\lambda_{il}(k)) = \lambda_{il}(k), \Phi_i(\Omega_{il}(k)) = \Omega_{il}(k)$. It follows that $r_s(k), \omega_s(k) \in A_{\chi(s)}$, and $\Omega_{il}(k), \lambda_{il}(k) \in A_i$.

The result follows from the operation of the algorithm and has been proved in details in [18].

Next, we state another result without proof.

$$\sum_{i \in n(l)} \Upsilon_i(\Omega_{il}(k)) \leq C_l \tag{2}$$

This result follows from the computation of $\vec{\omega}(k)$ in step (4) of the algorithm, and relation (1). This result can be further strengthened in the special case of $\Lambda(k) = \phi$, (i.e., if no new virtual session saturates in step (6) of the algorithm in iteration k). Lemma 3 proves that in this case the utility of any one unsaturated virtual session j can be increased from $\omega_j(k)$ to the next higher level, if all other virtual sessions have utilities assigned by the utility vector $\vec{\omega}(k)$. The proof follows from the saturation condition for a virtual session.

Lemma 3: If $k \geq 1, \Lambda(k) = \phi$, then for all virtual sessions $j \in S(k-1)$

$$\Upsilon_{\chi(j)}\left(\max(\Omega_{\chi(j)l}(k), \Psi_{\chi(j)}(\omega_j(k)))\right) + \sum_{i \in n(l), i \neq \chi(j)} \Upsilon_i(\Omega_{il}(k)) \leq C_l, \quad \forall l \in L_j.$$

Lemma 4 (feasibility lemma) shows that the utility allocation at the end of every iteration is feasible. This is argued as follows. Virtual session utilities belong to the respective utility sets (Lemma 2). Utility of a virtual session at the 0th iteration is its minimum utility. Utility of a virtual session does not decrease in subsequent iterations (Lemma 1). Thus minimum utility constraints are satisfied. We show that the capacity constraints hold using Lemma 3 and (2). The formal proof follows. We will use this feasibility result in the proof of Theorem 1.

Lemma 4: The utility allocation $\vec{r}(k)$ at the end of the kth iteration is feasible, $k \geq 0$.

Proof of Lemma 4: We prove by induction. Note that $r_s(0) = \mu_s$, for any virtual session s. Thus $\vec{r}(0)$ satisfies the minimum utility requirements. Also, $\mu_j \in A_{\chi(j)}$ by assumption for all virtual sessions j. Note that $\lambda_{il}(0) = \mu_{il}$, for all sessions $i \in n(l)$ and all links l. Since the set of feasible rate allocation vectors is nonempty, $\sum_{i \in n(l)} \Upsilon_i(\mu_{il}) \leq C_l$, for any link l. Thus, $\vec{r}(0)$ is a feasible utility 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. For all virtual sessions, $j, r_j(k+1) \in A_{\chi(j)}$ by Lemma 2. For all virtual sessions, $s, r_s(k+1) \geq r_s(k) \geq \mu_s$. The first inequality follows from Lemma 1 and the last from the feasibility of $\vec{r}(k)$. Thus, $\vec{r}(k+1)$ satisfies the minimum utility requirements.

Now, we prove that $\vec{r}(k+1)$ satisfies the capacity constraint. If $r_j(k+1) = \omega_j(k+1)$, for all virtual sessions j, traversing through link l, then $\lambda_{il}(k+1) = \Omega_{il}(k+1)$, for all sessions i traversing link $n(l)$. Thus, $\vec{r}(k+1)$ satisfies the capacity condition from (2). If $r_j(k+1) \neq \omega_j(k+1)$, for one or more virtual sessions j traversing through link l, then $r_s(k+1) > \omega_s(k+1)$ 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 $\lambda_{\chi(s)l}(k+1) = \max(\Psi_{\chi(s)}(\omega_s(k+1)), \Omega_{\chi(s)l}(k+1))$. Note that $s \in S(k)$ and $\Lambda(k+1) = \phi$ as $r_s(k+1) > \omega_s(k+1)$. Thus, Lemma 3 applies for virtual session s

$$\begin{aligned}\sum_{i\in n(l)} \Upsilon_i(\lambda_{il}(k+1)) &= \sum_{i\in n(l), i\neq\chi(s)} \Upsilon_i(\Omega_{il}(k+1)) \\ &\quad + \Upsilon_{\chi(s)}\left(\max\left(\Psi_{\chi(s)}(\omega_s(k+1)), \Omega_{\chi(s)l}(k+1)\right)\right) \\ &\le C_l \text{ (from Lemma 3).}\end{aligned}$$

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

Now we will introduce the notion of a *generalized bottleneck link*. Utility allocation $\vec{r}$ is maximally fair iff every virtual session has a generalized bottleneck link, w.r.t. $\vec{r}$. We will prove the sufficiency of this condition in Lemma 5. First, we introduce another function, $\psi_i(x)$. Let $\psi_i(x)$ be the minimum utility not less than x. More formally, $\psi_i(x)$ is a function defined as follows:

$$\psi_i(x) = \inf_{y\in R}\{y: x \le y = \Phi_i(z), \text{ for some } z \in R\}.$$

For example, if $U_i(x) = b\lfloor x/b \rfloor$, then the utility set consists of nonnegative integral multiples of b, and $\Phi_i(x) = b\lfloor x/b \rfloor, \psi_i(x) = b\lceil x/b \rceil$. Since Φ_i is a nondecreasing function, ψ_i is a nondecreasing function for all x.

A link l is a generalized bottleneck link for a virtual session s w.r.t. a utility allocation $\vec{r}$ if

1) $r_s = \lambda_{\chi(s)l}$, (recall that $\lambda_{il} = \max_{j\in m(i,l)} r_j$).
2) If $r_s = \Psi_{\chi(s)}(r_s)$, then $\sum_{i\in n(l)} \Upsilon_i(\lambda_{il}) = C_l$. Also, if $\lambda_{il} > \mu_{il}$ for any session i traversing the link l, then $\lambda_{il} \le \psi_i(r_s)$.
3) If $r_s < \Psi_{\chi(s)}(r_s)$, then

$$\begin{aligned}C_l - \sum_{i\in n(l)} \Upsilon_i(\lambda_{il}) + \sum_{\substack{i\in n(l),\\ \lambda_{il} > \max\left(\mu_{il}, \psi_i\left(\Psi_{\chi(s)}(r_s)\right)\right)}} &(\Upsilon_i(\lambda_{il}) \\ - \Upsilon_i(\max(\mu_{il}, \psi_i(\Psi_{\chi(s)}(r_s))))) \\ < \Upsilon_{\chi(s)}\left(\Psi_{\chi(s)}(r_s)\right) - \Upsilon_{\chi(s)}(r_s).\end{aligned}$$

We follow the convention that $\sum_{z\in A} z = 0$, if set $A = \phi$.

We briefly explain the bottleneck conditions here. The first condition states that virtual session s has the maximum utility among all other virtual sessions of its session traversing its generalized bottleneck link l. Conditions 2) and 3) deal with the two possible cases: $r_s = \Psi_{\chi(s)}(r_s)$ and $r_s < \Psi_{\chi(s)}(r_s)$. The first happens when the utility set of receiver s contains a right neighborhood $[r_s, r_s + \delta)$, for some $\delta > 0$. Again, this happens when the utility function for receiver s increases strictly in a right neighborhood of r_s. In this case, the bottleneck conditions require that the total bandwidth of the link l be fully utilized under the utility allocation $\vec{r}, (\sum_{i\in n(l)} \Upsilon_i(\lambda_{il}) = C_l)$. Also, consider a session i traversing link l with session link utility λ_{il} higher than its minimum required utility μ_{il}. The utility allocation of any such session i is upper bounded by $\psi_i(r_s)$ (if $\lambda_{il} > \mu_{il}$ then $\lambda_{il} \le \psi_i(r_s)$). Recall that $\psi_i(x)$ is the minimum utility of session i not lower than x. Thus a virtual session has "almost maximum" utility in its generalized bottleneck link in this case. Condition 3) applies when $r_s < \Psi_{\chi(s)}(r_s)$. Again, this happens when the utility function for receiver s does not increase strictly in a right neighborhood of r_s. Condition 3) can be explained as follows. The first summation in the left-hand side of the expression in condition 3) $(C_l - \sum_{i\in n(l)} \Upsilon_i(\lambda_{il}))$ is the available bandwidth in the link. Consider a session i whose session link utility in link l is higher than both μ_{il} and $\Psi_{\chi(s)}(r_s)$ (recall that $\psi_i(\Psi_{\chi(s)}(r_s))$ is the least utility value for session i not lower than $\Psi_{\chi(s)}(r_s)$). The second term in the left hand side $(\sum_{\substack{i\in n(l),\\ \lambda_{il} > \max(\mu_{il}, \psi_i(\Psi_{\chi(s)}(r_s)))}} (\Upsilon_i(\lambda_{il}) - \Upsilon_i(\max(\mu_{il}, \psi_i(\Psi_{\chi(s)}(r_s))))))$ is the additional bandwidth obtained by reducing the session link utility of any such session i to either μ_{il} or $\psi_i(\Psi_{\chi(s)}(r_s))$, whichever is higher. The term in the right hand side $(\Upsilon_{\chi(s)}(\Psi_{\chi(s)}(r_s)) - \Upsilon_{\chi(s)}(r_s))$ is the bandwidth required to increase the utility of virtual session s to the next higher level, $\Psi_{\chi(s)}(r_s)$. Note that $\Psi_{\chi(s)}(r_s)$ is a utility value which is one level higher than r_s in this case, i.e., $\Psi_{\chi(s)}(r_s)$ is greater than r_s, but there is no valid utility value for the session of receiver s between r_s and $\Psi_{\chi(s)}(r_s)$. Condition 3) states that the total bandwidth obtained after such reduction (the sum of the available bandwidth and the additional bandwidth released after reduction) is not sufficient to increase the utility of receiver s to the next higher value, $\Psi_{\chi(s)}(r_s)$. Thus, the utility of receiver s can be increased, only by decreasing the session link utility of one or more sessions to a value below r_s.

The following lemma shows that if every virtual session has a generalized bottleneck link, then the utility allocation is maximally fair. We sketch the proof for the forward part here. We will use this lemma in proving Theorem 1.

Lemma 5: A feasible utility allocation vector is maximally fair if every virtual session s has a generalized bottleneck link.

Sketch of Proof for Lemma 5: Let $\vec{r}^1$ be a feasible utility allocation. Let every virtual session have a generalized bottleneck link under $\vec{r}^1$. Let $\vec{r}^1$ not be a maximally fair utility allocation. Thus, from the definition of maximal fairness, there must exist a feasible utility allocation, $\vec{r}^2$ fairer than $\vec{r}^1$. Define the set τ of virtual sessions as follows, $\tau = \{j: r_j^1 \neq r_j^2\}$ (τ is the set of virtual sessions which have different utilities under allocations $\vec{r}^1$ and $\vec{r}^2$). Since $\vec{r}^2$ is fairer than $\vec{r}^1$, there exists a virtual session $s_{\min}$ such that $s_{\min}$ has the minimum utility in τ under $\vec{r}^2$, (i.e., $r_{s_{\min}}^2 = \min_{i\in\tau} r_i^2$) and the utility of $s_{\min}$ is even lower under $\vec{r}^1$ (i.e., $r_{s_{\min}}^2 > r_{s_{\min}}^1$). This follows from a necessary and sufficient condition for relative fairness stated in [13]. The condition applies for the relative fairness in any feasible set in R^M and hence holds for the relative fairness of utility allocations as well.

Now we state a property of utility allocation $\vec{r}^1$, which follows from the fact that every virtual session j has a generalized bottleneck link under $\vec{r}^1$. If the utility allocation of a component j in $\vec{r}^1$ is increased, then the utility allocation of one or more virtual sessions must be decreased to values below r_j^1, in order to maintain feasibility. The intuition is similar to the explanation for the bottleneck conditions stated before this lemma, and the precise argument has been stated in [18].

Note that the utility of component $s_{\min}$ is higher in $\vec{r}^2$ than in $\vec{r}^1$ $(r_{s_{\min}}^1 < r_{s_{\min}}^2)$ and $\vec{r}^2$ is feasible by assumption. There exists at least one component j whose utility allocation is lower under $\vec{r}^2$, (i.e., $r_j^2 < r_j^1$) and also the utility allocation of j under $\vec{r}^2$ is less than $r^1_{s_{\min}}$, i.e., $(r_j^2 < r^1_{s_{\min}})$. Thus, $r_j^2 < r^1_{s_{\min}} < r^2_{s_{\min}}$, and $j \in \tau$ as $r_j^2 < r_j^1$. However, this contradicts the definition of $s_{\min}, r^2_{s_{\min}} = \min_{i \in \tau} r_i^2$. The result follows. □

Now we show that if a virtual session saturates in an iteration k, it has a generalized bottleneck link in all subsequent iterations. Since the computation algorithm terminates only when all virtual sessions saturate, the maximal fairness of the output utility allocation follows from Lemma 5.

Lemma 6: If $s \notin S(k), k \geq 0$, then virtual session s has a generalized bottleneck link w.r.t. utility vector, $r(\vec{k})$.

Proof of Lemma 6: 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 $s \notin S(k)$. We need to show that virtual session s has a generalized bottleneck link w.r.t. utility allocation vector $r(\vec{k})$. Since $s \notin S(k), s$ saturates in the kth iteration or earlier. Let k saturate in the $t+1$th iteration, $t+1 \leq k$. Thus $s \in S(t) \setminus S(t+1), t+1 \leq k$. Thus step (4) of the algorithm implies that $r_s(k+1) = \omega_s(k+1) = r_s(k) = \cdots = r_s(t+1)$ (Note that the computation algorithm does not change the utility of a virtual session once it saturates). We consider the cases $r_s(k) = \Psi_{\chi(s)}(r_s(k))$ and $r_s(k) < \Psi_{\chi(s)}(r_s(k))$ separately

$$\text{First let } r_s(k) = \Psi_{\chi(s)}(r_s(k)). \tag{3}$$

We need to prove the general bottleneck conditions 1) and 2) for some link $l \in L_s$

$$r_s(t+1) = \Psi_{\chi(s)}(r_s(t+1)) \quad \text{(from (3) and since } r_s(t+1) = r_s(k)). \tag{4}$$

From (4) and the saturation conditions in iteration $t+1$, there exists a link $l \in L_s$ s.t.

$$r_s(t+1) = \lambda_{\chi(s)l}(t+1) \tag{5}$$

$$\sum_{i \in n(l)} \Upsilon_i(\lambda_{il}(t+1)) = C_l. \tag{6}$$

We will show that this link l satisfies the general bottleneck conditions 1) and 2) for virtual session s, w.r.t. utility allocation $r(\vec{k})$. From Lemma 1, $\lambda_{il}(k) \geq \lambda_{il}(t+1)$, for all sessions i traversing l, since $t < k$. Thus (6) shows that $\lambda_{il}(k) = \lambda_{il}(t+1)$, for all sessions i traversing l, otherwise $\sum_{i \in n(l)} \Upsilon_i(\lambda_{il}(t+1)) > C_l$, which violates the feasibility condition for $\vec{r}(k)$. We know from Lemma 4 that $\vec{r}(k)$ is a feasible utility allocation.

Two conditions follow. First, $\lambda_{\chi(s)l}(k) = \lambda_{\chi(s)l}(t+1) = r_s(t+1) = r_s(k)$. Thus general bottleneck condition 1) holds. The second condition is that $\sum_{i \in n(l)} \Upsilon_i(\lambda_{il}(k)) = \sum_{i \in n(l)} \Upsilon_i(\lambda_{il}(t+1)) = C_l$. Thus, the first part of the general bottleneck condition 2) holds.

We sketch the proof for the second part of the general bottleneck condition 2). We show that $r_s(t+1) \geq \Phi_{\chi(s)}(\eta_l(t+1))$. Note that $\Phi_i(x)$ is the largest session i utility not greater than x. Now using certain technical properties of the Φ function and the fact that $r_s(t+1) = \Psi_{\chi(s)}(r_s(t+1))$ [see (4)], we show that furthermore $r_s(t+1) \geq \eta_l(t+1)$ in this case. Thus, $\psi_i(r_s(t+1)) \geq \psi_i(\eta_l(t+1))$, for all sessions i, since ψ_i is a nondecreasing function. Next, we argue that $\lambda_{il}(t+1) \leq \psi_i(\eta_l(t+1))$, for all sessions i, if $\lambda_{il}(t+1) > \mu_{il}$. Thus, it follows that $\lambda_{il}(t+1) \leq \psi_i(r_s(t+1))$ for all sessions i, if $\lambda_{il}(t+1) > \mu_{il}$. The second part of condition 2) follows since $r_s(k) = r_s(t+1)$ and $\lambda_{il}(t+1) = \lambda_{il}(k)$ for all sessions i traversing link l.

Now let $r_s(k) < \Psi_{\chi(s)}(r_s(k))$. We need to show that there exists a link l which satisfies the generalized bottleneck conditions 1) and 3) for virtual session s w.r.t. utility allocation $\vec{r}(k)$. Since $r_s(k) = r_s(t+1)$ as observed before, $r_s(t+1) < \Psi_{\chi(s)}(r_s(t+1))$. Since $s \in S(t) \setminus S(t+1)$, there exists a link $l \in L_s$ s.t.

$$r_s(t+1) = \lambda_{\chi(s)l}(t+1) \tag{7}$$

and

$$\sum_{i \in n(l)} \Upsilon_i(\lambda_{il}(t+1)) > C_l - \left(\Upsilon_{\chi(s)}\left(\Psi_{\chi(s)}(r_s(t+1))\right) - \Upsilon_{\chi(s)}(r_s(t+1))\right). \tag{8}$$

We next show that this link l satisfies the required generalized bottleneck conditions 1) and 3).

$$\Upsilon_{\chi(s)}\left(\Psi_{\chi(s)}\left(\lambda_{\chi(s)l}(t+1)\right)\right) + \sum_{\substack{i \in n(l)\\ i \neq \chi(s)}} \Upsilon_i(\lambda_{il}(t+1)) > C_l \text{ (using (7) and (8))}. \tag{9}$$

Note that $\sum_{i \in n(l)} \Upsilon_i(\lambda_{il}(k)) \leq C_l$ from feasibility of $r(\vec{k})$ (Lemma 4). Thus, from (9) and since $\lambda_{il}(t+1) \leq \lambda_{il}(k)$ (Lemma 1), $\lambda_{\chi(s)l}(k) < \Psi_{\chi(s)}(\lambda_{\chi(s)l}(t+1))$. Also, $\lambda_{\chi(s)l}(t+1) \leq \lambda_{\chi(s)l}(k)$. From the definition of the $\Psi_{\chi(s)}$ function, and since $\lambda_{\chi(s)l}(k)$ is in the range of $\Phi_{\chi(s)}$ function (Lemma 2), $\lambda_{\chi(s)l}(k) = \lambda_{\chi(s)l}(t+1)$. As $r_s(t+1) = r_s(k)$, bottleneck condition 1) of the lemma follows from (7).

We sketch the proof of bottleneck condition 3). We have argued in [18] that

$$\begin{aligned}
&\text{If } \mu_{il} \geq \psi_i\left(\Psi_{\chi(s)}(r_s(t+1))\right), \\
&\qquad \text{then } \lambda_{il}(t+1) = \mu_{il} \\
&\text{If } \mu_{il} < \psi_i\left(\Psi_{\chi(s)}(r_s(t+1))\right), \\
&\qquad \text{then } \lambda_{il}(t+1) \leq \psi_i\left(\Psi_{\chi(s)}(r_s(t+1))\right)
\end{aligned}$$

Using $r_s(k) = r_s(t+1)$, it follows that:

$$\lambda_{il}(t+1) \leq \max\left(\mu_{il}, \psi_i\left(\Psi_{\chi(s)}(r_s(k))\right)\right) \quad \text{for all } i \in n(l) \tag{10}$$

Using (10), we can upper bound the second summation in the left-hand side of (3)

$$\begin{aligned}
&\sum_{\substack{i \in n(l),\\ \lambda_{il}(k) > \max\left(\mu_{il}, \psi_i\left(\Psi_{\chi(s)}(r_s(k))\right)\right)}} \left(\Upsilon_i(\lambda_{il}(k)) - \Upsilon_i\left(\max\left(\mu_{il}, \psi_i\left(\Psi_{\chi(s)}(r_s(k))\right)\right)\right)\right) \\
&\leq \sum_{\substack{i \in n(l),\\ \lambda_{il}(k) > \max\left(\mu_{il}, \psi_i\left(\Psi_{\chi(s)}(r_s(k))\right)\right)}} \left(\Upsilon_i(\lambda_{il}(k)) - \Upsilon_i(\lambda_{il}(t+1))\right) \\
&\leq \sum_{i \in n(l)} \left(\Upsilon_i(\lambda_{il}(k)) - \Upsilon_i(\lambda_{il}(t+1))\right).
\end{aligned} \tag{11}$$

The last inequality follows since Υ_i is an increasing function and $\lambda_{il}(k) \geq \lambda_{il}(t+1)$ for all i.

$$\begin{aligned}
C_l - &\sum_{i \in n(l)} \Upsilon_i(\lambda_{il}(k)) \\
+ &\sum_{\substack{i \in n(l), \\ \lambda_{il}(k) > \max\left(\mu_{il}, \psi_i\left(\Psi_{\chi(s)}(r_s(k))\right)\right)}} (\Upsilon_i(\lambda_{il}(k)) \\
- &\Upsilon_i\left(\max\left(\mu_{il}, \psi_i\left(\Psi_{\chi(s)}(r_s(k))\right)\right)\right)) \\
&\leq C_l - \sum_{i \in n(l)} \Upsilon_i(\lambda_{il}(t+1)) \quad \text{(from (11))} \\
&< \Upsilon_{\chi(s)}\left(\Psi_{\chi(s)}(r_s(t+1))\right) - \Upsilon_{\chi(s)}(r_s(t+1)) \quad \text{(by (8))} \\
&= \Upsilon_{\chi(s)}\left(\Psi_{\chi(s)}(r_s(k))\right) - \Upsilon_{\chi(s)}(r_s(k)) \\
&\qquad\qquad \text{(since } r_s(k) = r_s(t+1)\text{)}.
\end{aligned}$$

Thus, bottleneck condition 3) of the lemma holds in this case.□

Now we prove Theorem 1. The proof follows from Lemmas 4 and 6.

Proof of Theorem 1: Let the computation algorithm terminate in k iterations. Note that $\vec{r}(k)$ is feasible by Lemma 4. Since $S(k) = \phi$, there does not exist a virtual session s in $S(k)$. Thus from Lemma 6 for every virtual session s there exists a link which satisfies the properties of a generalized bottleneck link for virtual session s w.r.t. utility vector $\vec{r}(k)$. Thus $\vec{r}(k)$ is a maximally fair utility allocation by Lemma 5. Thus, part 1) follows.

If a maxmin fair utility allocation exists, then it is fairer than any other utility allocation, by definition of relative and maxmin fairness [16]. Thus it is the only maximally fair utility allocation in the feasible set, by definition of maximal fairness. Since the output utility allocation, $\vec{r}(k)$ is maximally fair under all circumstances, part 2) follows. □

C. *Proof of the Finite-Termination Theorem*

Now we will prove that the algorithm terminates in a finite number of iterations (Theorem 2). We prove it using Lemma 1 in the first subsection and the following additional lemmas.

The following lemma shows that if no virtual session saturates in step 6) of the algorithm, then at least one virtual session meets the condition for utility incrementation in step 8). Thus there is either a saturation or a utility incrementation in every iteration of the algorithm. The proof looks at the link $l_{\min}$ which attains the minimum-link control parameter, amongst all those which carry at least one unsaturated virtual session. It chooses an unsaturated session $i_{\min}$ traversing link $l_{\min}$ by certain criteria, and argues that all the unsaturated virtual sessions of $i_{\min}$ satisfy the required properties. Refer to [18] for details.

Lemma 7: If $\Lambda(k) = \phi$, there always exists a virtual session $s \in S(k-1)$ such that

$$\omega_s(k) = \Phi_{\chi(s)}\left(\min_{l \in L_s} \eta_l(k)\right), \quad \Psi_{\chi(s)}(\omega_s(k)) > \omega_s(k)$$

For some $l \in L_s$ s.t. $\Phi_{\chi(s)}(\min_{l_1 \in L_s} \eta_{l_1}(k)) = \Phi_{\chi(s)}(\eta_l(k))$

$$\omega_s(k) = \Omega_{\chi(s)l}(k) \quad \text{and} \quad \mathrm{Ran}(\Phi_{\chi(s)}) \cap (\eta_l(k), \alpha_l(k)) = \phi$$

and for all $l \in L_s$ s.t. $\Phi_{\chi(s)}(\min_{l_1 \in L_s} \eta_{l_1}(k)) = \Phi_{\chi(s)}(\eta_l(k))$, $\Omega_{il}(k) = \max(\Phi_i(\eta_l(k)), \lambda_{il}(k-1)) \, \forall i \in \Xi_l(k-1)$.

The following lemma shows that the number of iterations in which there is a utility incrementation in step 8) is upper-bounded by $|\mathcal{L}|M^2$. Since Lemma 7 indicates that there is always a utility incrementation if no virtual session saturates in step 6), this means that the number of iterations in which no virtual session saturates in step 6) is at most $|\mathcal{L}|M^2$. The proof uses Lemmas 1 and 7. Interestingly, this result is independent of the nature of the utility functions.

Lemma 8: Let $T = \{k: \Lambda(k) = \phi\}$. The cardinality of T is at most $|\mathcal{L}|M^2$ where $\mathcal{L}$ is the set of links and M is the number of virtual sessions.

Proof of Lemma 8: Set T is the set of iterations which execute step 8). From Lemma 7, for every iteration $k \in T$ there exists at least one virtual session s and link l pair s.t. $l \in L_s, s \in S(k-1)$ which satisfies the following properties:

$$\Phi_{\chi(s)}\left(\min_{l_1 \in L_s} \eta_{l_1}(k)\right) = \Phi_{\chi(s)}(\eta_l(k)) \tag{12}$$

$$\omega_s(k) = \Phi_{\chi(s)}(\eta_l(k)) \tag{13}$$

$$\omega_s(k) = \Omega_{\chi(s)l}(k) \tag{14}$$

$$\forall i \in \Xi_l(k-1)$$

$$\Omega_{il}(k) = \max(\Phi_i(\eta_l(k)), \lambda_{il}(k-1)) \tag{15}$$

$$\Psi_s(\omega_s(k)) > \omega_s(k) \tag{16}$$

$$\mathrm{Ran}\left(\Phi_{\chi(s)}\right) \cap (\eta_l(k), \alpha_l(k)) = \phi. \tag{17}$$

Now $r_s(k) = \Psi_{\chi(s)}(\omega_s(k)) > \omega_s(k)$ for one such virtual session s. We call such a virtual session $s(k)$ and the link $l(k)$. We will show that the pair (j, l_1) can occur for at most $|n(l_1)|$ times in the sequence $\{(s(k), l(k))\}$. There can be at most $|\mathcal{L}|M$ different virtual session, link pairs. Thus $|T| \leq |\mathcal{L}|M \max_{l \in \mathcal{L}} |n(l)| \leq |\mathcal{L}|MN \leq |\mathcal{L}|M^2$, where N is the total number of sessions.

Now, we show that the pair (j, l_1) can occur at most $|n(l_1)|$ times in the sequence $\{(s(k), l(k))\}$. Let $(s(k), l(k)) = (s(t), l(t)) = (s, l), k < t$. Note that the saturation status of a session in a link can only change from unsaturation to saturation. Thus, $\Xi_l(t-1) \subseteq \Xi_l(k-1)$. We will show that $\Xi_l(t-1) \subset \Xi_l(k-1)$. Thus, at least one session saturates in link l between any two occurrences of (s, l) in the sequence. Clearly (s, l) can not occur in the sequence after all sessions saturate in link l. Thus, (s, l) can occur at most $|n(l)|$ times in the sequence.

Now we show that $\Xi_l(t-1) \subset \Xi_l(k-1)$. Assume otherwise, i.e., $\Xi_l(t-1) = \Xi_l(k-1)$. Since $r_s(k) = \Psi_{\chi(s)}(\omega_s(k))$, it follows from (16) and (13) that $r_s(k) > \omega_s(k) = \Phi_{\chi(s)}(\eta_l(k))$. Also $r_s(k) \leq \omega_s(t) = \Phi_{\chi(s)}(\eta_l(t))$. The first inequality follows from Lemma 1, since $t > k$. The second follows from (13). It follows that $\Phi_{\chi(s)}(\eta_l(t)) > \Phi_{\chi(s)}(\eta_l(k))$. Since $\Phi_{\chi(s)}$ is a nondecreasing function, it follows that $\eta_l(t) > \eta_l(k)$. Since $\Xi_l(t-1) = \Xi_l(k-1), F_l(t-1) = F_l(k-1)$:

$$\begin{aligned} C_l &\geq F_l(t-1) + \sum_{i \in \Xi_l(t-1)} \Upsilon_i(\max(\Phi_i(\eta_l(t)), \lambda_{il}(t-1))) \\ &\qquad \text{(from (1))} \\ &\geq F_l(k-1) + \sum_{i \in \Xi_l(t-1)} \Upsilon_i(\max(\Phi_i(\eta_l(t)), \lambda_{il}(k-1))) \\ &\qquad \text{(since } \lambda_{il}(k-1) \leq \lambda_{il}(t-1) \text{ by Lemma 1)} \\ &= F_l(k-1) + \sum_{i \in \Xi_l(k-1)} \Upsilon_i(\max(\Phi_i(\eta_l(t)), \lambda_{il}(k-1))) \\ &\qquad \text{(since } \Xi_l(k-1) = \Xi_l(t-1)). \quad (18) \end{aligned}$$

Thus, from definition, $\alpha_l(k) \geq \eta_l(t)$. If $\alpha_l(k) = \eta_l(t)$, then from (18), $F_l(k-1) + \sum_{i \in \Xi_l(k-1)} \Upsilon_i(\max(\Phi_i(\alpha_l(k)), \lambda_{il}(k-1))) \leq C_l$. This means that $\eta_l(k) = \alpha_l(k)$. Thus $\eta_l(k) = \alpha_l(k) = \eta_l(t)$. But, we have shown that $\eta_l(t) > \eta_l(k)$. It follows that $\alpha_l(k) > \eta_l(t)$. Let, if possible, $\Phi_{\chi(s)}(\eta_l(t)) > \eta_l(k)$. Since $\Phi_{\chi(s)}(\eta_l(t)) \leq \eta_l(t)$ by property (2) of $\Phi_{\chi(s)}, \Phi_{\chi(s)}(\eta_l(t)) \in (\eta_l(k), \eta_l(t)]$. Since $\eta_l(t) < \alpha_l(k), \Phi_{\chi(s)}(\eta_l(t)) \in (\eta_l(k), \alpha_l(k))$. This contradicts (17). Thus, $\Phi_{\chi(s)}(\eta_l(t)) \leq \eta_l(k)$. Since $\Phi_{\chi(s)}(\eta_l(t)) > \Phi_{\chi(s)}(\eta_l(k)), \Phi_{\chi(s)}(\eta_l(t)) \in (\Phi_{\chi(s)}(\eta_l(k)), \eta_l(k)]$. This contradicts property (3) of the $\Phi_{\chi(s)}$ function. Thus, $\Xi_l(t-1) \neq \Xi_l(k-1)$. Thus, $\Xi_l(t-1) \subset \Xi_l(k-1)$. □

Now, we prove Theorem 2. The proof uses Lemmas 7 and 8.

Proof of Theorem 2 (Finite Termination Theorem): Observe that at least one of the following holds for every iteration $k \geq 1$:

1) $\Lambda(k) = \phi$;
2) $S(k) \subset S(k-1)$ (proper subset).

The status of a virtual session can change from saturation to unsaturation only. Thus, $S(k) \subseteq S(k-1)$ for all $k \geq 1$. So (2) can hold for atmost M iterations, where $S(0) = \{1, \ldots, M\}$. Lemma 8 shows that (1) can hold for at most $|\mathcal{L}|M^2$ iterations. Thus the algorithm must terminate in $M + |\mathcal{L}|M^2$ iterations.□

VI. Conclusion and Discussion

This paper develops a framework for studying diverse fairness objectives in multirate, multicast networks. The key contribution is to present an algorithm for computing the fair utility allocations for general utility functions. The algorithm terminates in polynomial number of iterations. We mention some interesting topics for related future research. We pointed out that computation of lexicographically optimal utility allocation is NP-hard. Hence, we focused on a weaker notion of fairness, maximal fairness which has intuitively appealing fairness properties. Another direction is to develop approximation algorithms and heuristics for computing lexicographically optimal utility allocation. It may also be possible to develop a lower complexity algorithm for computing maximally fair utility allocation. Another promising direction is to explore the connection between the choice of utility functions and pricing mechanisms. We hope that this study would initiate further research in these areas.

References

[1] T. Ballardie, P. Francis, and J. Crowcroft, "Core based trees: An architecture for scalable inter-domain multicast routing," in *Proc. ACM SIGCOMM*, Ithaca, NY, Sept. 1993, pp. 85–95.
[2] D. Bertsekas and R. Gallager, *Data Networks*. Upper Saddle River, NJ: Prentice-Hall, 1987.
[3] T. Bially, B. Gold, and S. Seneff, "A technique for adaptive voice flow control in integrated packet networks," *IEEE Trans. Commun.*, vol. COM-28, pp. 325–333, Mar. 1980.
[4] Z. Cao and E. Zegura, "Utility max-min: An application-oriented bandwidth allocation scheme," in *Proc. IEEE INFOCOM'99*.
[5] S. Deering and D. Cheriton, "Multicast routing in datagram Internetworks and extended LANs," *ACM Trans. Comp. Syst.*, vol. 8, no. 2, pp. 54–60, Aug. 1994.
[6] F. Kishino, K. Manabe, Y. Hayashi, and H. Yasuda, "Variable bit-rate coding of video signals for ATM networks," *IEEE J. Select. Areas Commun.*, vol. 7, no. 5, June 1989.
[7] F. Kelly, A. Maulloo, and D. Tan, "Rate control for communication networks: Shadow prices, proportional fairness and stability," *J. Oper. Res. Soc.*, vol. 49, no. 3, pp. 237–252, Mar. 1998.
[8] X. Li, S. Paul, and M. H. Ammar, "Layered video multicast with retransmission (LVMR): Evaluation of hierarchical rate control," in *Proc. IEEE INFOCOM'98*, San Francisco, CA, Mar. 1998.
[9] ——, "Multi-session rate control for layered video multicast," College of Computing, Georgia Inst. Technol., Tech. Rep. GT-CC-98-21, 1998.
[10] S. McCanne, "Scalable compression and tansmission of Internet multicast video," Ph.D. dissertation, Univ. California, Berkeley, CA, Dec. 1996.
[11] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast," in *Proc. ACM SIGCOMM'96*, Stanford, CA, Sept. 1996.
[12] D. Rubenstein, J. Kurose, and D. Towsley, "The impact of multicast layering on network fairness," in *Proc. ACM SIGCOMM'99*, Cambridge, MA, Sept. 1999.
[13] S. Sarkar and K. N. Sivarajan, "Fairness in cellular mobile networks," in *Proc. 34th Ann. Allerton Conf. Communication, Control, Computing*, 1996, pp. 457–469.
[14] S. Sarkar, "Fairness and congestion Ccontrol in multirate multicast networks," Ph.D. dissertation, Univ. Maryland, College Park, MD, July 2000.
[15] S. Sarkar and L. Tassiulas, "Distributed algorithms for computation of fair rates in multirate multicast trees," in *Proc. IEEE INFOCOM'2000*, Tel Aviv, Israel, Mar. 2000.
[16] ——, "Fair allocation of discrete bandwidth layers in multicast networks," in *Proc. IEEE INFOCOM'2000*, Tel Aviv, Israel, Mar. 2000.
[17] ——, "Fair allocation of discrete bandwidth layers in multicast networks," Inst. Systems Research, Univ. Maryland, College Park, MD, Tech. Rep. TR 99-43, 1999.
[18] ——, (2001) Fair allocation of utilities in multirate multicast networks. Electrical Engineering Department, Univ. Pennsylvania, Philadelphia, PA. [Online]. Available: http://www.seas.upenn.edu/~swati/publication.htm
[19] T. Turletti and J. C. Bolot, "Issues with multicast video distribution in heterogeneous packet networks," presented at the 1994 Packet Video Workshop, Portland, OR.

Saswati Sarkar received the Master of Engineering degree in electrical communication engineering from the Indian Institute of Science, Bangalore, and the Ph.D. degree in electrical and computer engineering from the University of Maryland, College Park, in 1996 and 2000, respectively.

She is currently an Assistant Professor in the department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA. Her research interests are in resource allocation and performance analysis in communication networks.

Leandros Tassiulas was born in Katerini, Greece, in 1965. He received the Diploma in electrical engineering from the Aristotelian University of Thessaloniki, Greece, and the M.S. and Ph.D. degrees in electrical engineering from the University of Maryland, College Park, in 1987, 1989, and 1991, respectively.

From 1991 to 1995, he was an Assistant Professor in the Department of Electrical Engineering, Polytechnic University, Brooklyn, NY. In 1995, he joined the Department of Electrical Engineering, University of Maryland, where he is now an Associate Professor. He holds a joint appointment with the Institute for Systems Research, and is the member of the Center for Satellite and Hybrid Communication Networks, established by NASA. His research interests are in computer and communication networks, with emphasis on wireless communications (terrestrial and satellite systems) and high-speed network architectures and management, in control and optimization of stochastic systems in parallel and distributed processing. He coauthored a paper that received the INFOCOM 1994 Best Paper Award.

Dr. Tassiulas received a National Science Foundation (NSF) Research Initiation Award in 1992, an NSF Faculty Early Career Development Award in 1995, and an Office of Naval Research Young Investigator Award in 1997.