

Distributed and Optimal Rate Allocation in Application-Layer Multicast

Jinyao Yan^{*†}, Martin May[‡], Bernhard Plattner^{*}, Wolfgang Mühlbauer^{*}

^{*}Computer Engineering and Networks Laboratory, ETH Zurich, CH-8092, Switzerland

[†]Computer and Network Center, Communication University of China, Beijing 100024, China

[‡]Paris Research Lab, Technicolor, Paris, France

Abstract—Application-layer Multicast (ALM) is widely used as an alternative to IP multicast for one-to-many data delivery. Besides delay or bandwidth optimized overlay tree construction, rate allocation is the most important issue: given that relaying end hosts may be limited in the available download or upload capacity, how to optimize the aggregate utility of all receivers?

While significant efforts have been dedicated to efficient construction of multicast trees, this paper focuses on the second problem. We, in this paper, propose a *fully distributed network model and distributed rate allocation approaches that optimize the overall utility perceived by the members of a multicast application while being TCP-friendly to co-existing flows.*

Contrary to previous work, this paper presents in addition to a dual solution a *primal* algorithm, which avoids oscillations during convergence. Importantly, we describe an *asynchronous* implementation for both variants.

To evaluate our approach, we have implemented the proposed algorithms and run extensive simulations. Overall, our findings reveal that the algorithms generate minimal messaging overhead, and efficiently optimize the aggregate utility.

I. INTRODUCTION

Multicast addresses the problem of efficient delivery of data over the Internet to a large number of recipients. Applications, amongst others, include live video streaming, Internet radio, video conferencing, and multiplayer games. Unfortunately, due to technical, administrative, and business related issues IP multicast is still not yet widely deployed in the public Internet. To overcome the deployment problems, a new type of multicast solution has become popular: application-layer multicast (ALM) ([1], [2] and its references) or overlay multicast, which does not require support from underlying IP routers. In application-layer multicast, some end hosts serve as relays for other members of the multicast group.

Deploying ALM is challenging for two reasons. First, algorithms to form multicast trees should compute efficient, robust overlays that for instance minimize the delay between peers or optimize the available bandwidth [3]. Second, even after establishing a multicast tree, it may be necessary to control the sending rates of overlay nodes. After all, the available download and upload capacities of end hosts (e.g., DSL customers) are generally limited. Ideally, a rate allocation strategy avoids situations where overlay nodes that receive a high data rate cannot provide high data rates to many of their children

in the multicast tree due to lack of bandwidth. Unfortunately, in existing ALM approaches intermediate nodes determine the download rate without considering the application quality (or utility) and information from neighbor and parent nodes. For example, in end system multicast [1], the rate for each flow is calculated locally using the unicast TCP-friendly rate control algorithm (TRFC) [4], considering neither the structure of the multicast tree nor the streaming quality.

We, in this paper, study the latter problem, i.e., how to allocate rates such that the overall utility perceived by the members of a multicast application is optimized. One potential application is multicast video streaming [5] if overlay nodes can either simply forward the received video stream or truncated versions to their children¹. Yet, in principle any multicast application can benefit from rate allocation algorithms.

The challenge in our work is to develop *optimal, distributed, yet scalable* rate allocation algorithms for multicast sessions that optimize the overall utility of all receiving nodes. Contrary to existing work, we introduce a new notion of TCP-friendliness into our problem model, making sure that individual multicast flows compete fairly with any co-existing TCP traffic. Importantly, we decompose the global rate allocation problem into smaller more local problems, thus paving the way for a distributed approach to solve the optimization problem. The algorithms we will propose have to limit the messaging overhead in the multicast tree. To this end, we extend existing utility-pricing models [7], [8], [9] that have been proposed as an analytical tool for optimization based rate allocation mechanisms. Efficient solutions for the rate allocation problem are alleviated by our finding that bottleneck links, i.e., physical links where the rates of all its multicast flows exceed the available bandwidth, are generally only shared between nodes that are siblings in a well-formed multicast tree.

To the best of our knowledge, we are the first to devise such a *fully distributed and TCP-friendly* approach to the rate allocation problem, and to even present a *primal* solution. The proposed algorithms solely rely on asynchronous coordination of neighboring nodes and do not require communication of

¹Implementations such as MPEG-4 Fine Granularity Scalable (FGS) [6] demonstrate how to adapt the rate of video streams based on feedback from the underlying congestion control.

any link prices to maximize utility across all receivers. In this context, we describe two solutions, namely a typical dual algorithm, and a novel primal algorithm based on a feasible direction method which significantly differs from the primal algorithm in Kelly’s work [8] and other penalty algorithms such as in [10]. Our primal algorithm always returns a feasible and stable solution and avoids oscillations during convergence.

We have implemented our algorithms in order to demonstrate their feasibility and to study their performance. Extensive simulation-based experiments reveal:

- 1) the primal and dual algorithm indeed manage to maximize the aggregate utility for all receivers.
- 2) while the dual algorithm shows oscillations, the primal algorithm avoids oscillations during convergence.
- 3) the message overhead is very small, alleviating the deployment of such multicast systems.
- 4) both algorithms are TCP-friendly to co-existing unicast and multicast traffic (see our new definition in section IV-B), while rate allocation is proportionally fair in the multicast tree.

The rest of this paper is structured as follows. In Section II, we briefly review related work. Then, in Section III, we discuss the construction of overlay trees and validate our observation that only sibling flows share bottleneck links, which is a prerequisite for the theoretical work in this paper. Section IV proposes the distributed model and formulates the rate allocation problem, before we explain in Section V our proposed algorithms to solve the problem. We extend the algorithms to asynchronous environments which better reflect the reality of large networks in Section VI. After discussing and evaluating our algorithms in Section VII, we conclude in Section VIII.

II. RELATED WORK

Our approach is to study the problem of rate allocation from a theoretical perspective. Yet, we see peer-to-peer Internet video broadcast (e.g., [1],[5]) as a major field of application for our results. For a review of such technologies we refer to [2].

For IP multicast McCanne et al. presented rate control algorithms already more than 10 years ago, e.g., [11]. Unfortunately, their approach determines downstream sending rates at intermediate routers without taking into account any information from neighbor or parent routers in the multicast tree. Even today for application-layer multicast streaming, overlay nodes generally relay traffic to children nodes based on standard TCP or solutions such as TCP Friendly Rate Control (TFRC) [4]. Consequently, the overall structure of the multicast tree is ignored and aggregate utility is not optimized. In contrast, Akbari et al. [12] propose a simple optimal rate control algorithm, but they only consider access links and their model only allows a limited set of possible streaming rates.

The literature introduces pricing models [7], [8] as an alternative for optimizing the aggregate utility across all receivers. Kar et al. [13] have applied such models to IP multicast, and

the authors of [14] even considered co-existing unicast TCP sessions when optimizing the overall utility.

Five years later, Cui et al. [15] applied pricing models for utility optimization in application-layer multicast systems. However, their algorithm creates significant message overhead, since they exchange information about flow rates and all physical link prices via centralized delegate nodes. Our work avoids this issue and, therefore, considerably minimizes message overhead. Moreover, in addition to a dual solution, as proposed in a very early version of our work [16], we introduce a novel primal approach based on a feasible direction method that avoids oscillations during convergence.

III. OVERLAY TREES

Application-layer multicast requires to organize the participating nodes into a tree-like structure with “parent-child” relationships before data can be transmitted. Whenever a parent node receives a packet, it creates a copy and forwards it to each of its children. The construction of overlay trees is challenging for two reasons. First, only overlay trees that are optimized for properties such as delay and bandwidth will result in good performance for the recipients. Second, joining or leaving nodes may require continuous adjustments of the dissemination tree.

Before we focus on the rate allocation problem for given overlay trees, this section briefly discusses some aspects of overlay tree construction. Given their importance for our rate allocation solutions, one major focus of our discussion will be on bottleneck links, i.e., links in the multicast tree that are fully saturated.

A. Sharing of Bottleneck Links

Underlying networks can consist of heterogeneous nodes with diverse access bandwidths. Forwarding packets from one multicast node to another, which we refer to as a *flow*, can involve multiple intermediate physical hops. Even if bandwidth or delay-based approaches are used to construct an overlay multicast tree, there can remain **bottleneck** links. A *bottleneck* is a directed physical link that fully utilizes its available capacity.

Figure 1 provides an example of an application-layer multicast tree. The underlying physical topology is depicted by solid lines while arcs represent the individual flows of the multicast tree rooted at source h_0 . Let us further assume that the links l_1 , l_2 and l_3 are bottleneck links, which fully utilize their available capacities in the indicated direction.

For performance reasons² and for a simpler design of the algorithms themselves, it will be desirable that the following holds: *flows generally do not share a physical link that is fully utilized, unless they are sibling flows in the well-formed multicast tree*. Note that *siblings* are flows that are sent by the same parent node. In our example of Figure 1 flows f_3 and f_4 share the bottleneck link l_3 , but both flows are siblings.

Figure 1 provides an example of non-sibling flows sharing a bottleneck link: flows f_1 and f_5 both pass the bottleneck link

²limiting the number of exchanged messages

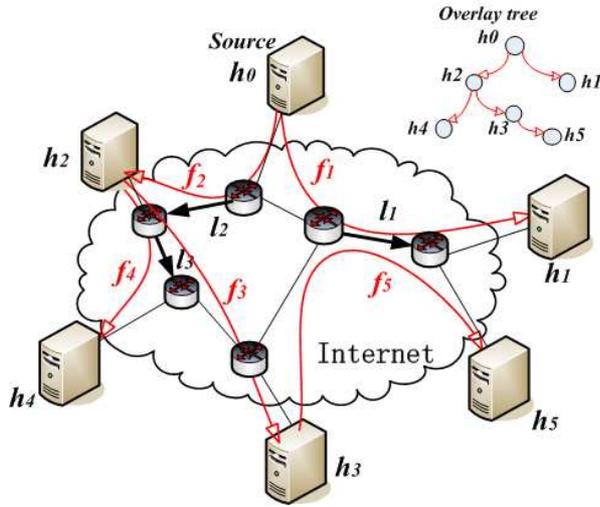


Fig. 1. Example of application-layer multicast tree and bottleneck links. Solid lines correspond to physical links while arcs represent the individual flows of a multicast tree rooted at source h_0 . The bold directed arrows l_1 , l_2 and l_3 are *bottlenecks* according to our definition. Note that an optimized overlay tree construction mechanism is likely to prevent f_5 from going through bottleneck link l_1 .

l_1 . However, f_5 is not a sibling of flow f_1 . In the following, we rely on various topologies and on testbed experiments to justify the validity of the assumption that it is predominantly sibling flows that share bottleneck links if this happens at all.

B. Sharing of Bottleneck Links in Simulated Topologies

Our goal is to study the location of bottleneck links in different physical and overlay topologies, and to confirm our assumption.

To this end, we rely on BRITE [17] to generate ten different network topologies each consisting of 1,000 routers³. As for our overlay topology, we need a mechanism to construct multicast trees on top of a given physical topology. A survey of proposed tree construction mechanisms can be found in [3]. Since applications such as multimedia streaming are often sensitive to time and bandwidth demands, existing mechanisms frequently seek to optimize the delay while considering bandwidth constraints, e.g., [18][19][20][21][22][23][24][25].

Hence, we rely on the same strategy and implement a tree construction mechanism that considers bandwidth and degree constraints but mainly optimizes for delay: new peers select a peer as parent in the multicast tree if this node is the closest in terms of end-to-end flow delay. To provide sufficient bandwidth, we further require that each peer has at most four children. For each of our physical topologies, we vary the number of nodes in the multicast trees between 5 and 1,000 nodes, and apply our tree construction algorithm.

Our results are clear: *there exists no single instance across our experiments where non-sibling flows share a bottleneck link*. This reflects our intuition that delay- and bandwidth-based tree construction mechanism generally avoid non-sibling

³More details of topologies are in Section VII-A

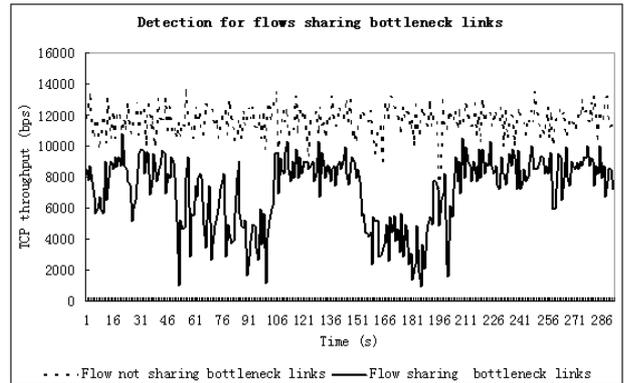


Fig. 2. Observed throughput when injecting `iperf` traffic at flow f during 50s - 100s and 150s - 200s. The solid line represents a flow that sees decreases in throughput while there is no correlation with the second flow (dashed line).

flows sharing bottleneck links such as link l_1 in Figure 1. Currently, this link is used by the non-sibling flows f_1 and f_5 . However, if we optimize for delay, we expect that node h_5 will for example receive data directly from h_1 and not any more from h_3 .

C. Sharing of Bottleneck Links in Testbed Experiments

When studying potential occurrences of bottleneck links, we do not want to limit ourselves to pure simulations. Therefore, we have implemented a proof-of-concept for an application-layer multicast system, which we deploy in the PlanetLab testbed [26]. Again, the goal is to illustrate that even for setups with real traffic we generally cannot find bottleneck links that are shared between non-sibling flows.

More precisely, we create a PlanetLab slice with 30 nodes located at different sites. We have implemented the delay-based tree construction algorithm that we introduced in the simulations. Based on this, we build a multicast tree that uses TCP⁴ to distribute data from a source to all other nodes of the PlanetLab slice. Finally, we run multiple experiments varying the source and number of nodes that are part of the multicast tree.

Overall, we aim at uncovering potential bottleneck links in the generated multicast trees. Our approach is as follows: For each experiment, we artificially add load to a flow f in the multicast tree by injecting `iperf` TCP traffic carefully⁵ between the source and destination node of the flow. Simultaneously, we examine all other flows in the multicast tree. By doing so, we try to find out flows in the tree sharing the bottleneck links that are traversed by the flow f . If the observed throughput decreases during traffic injection for any of the other flows (flow g), we take this as a hint that flow g shares a bottleneck link with flow f . The solid line of Figure 2 provides an examples where we observe significant declines in throughput. In addition to observing changes in throughput, we run traceroutes to detect shared links.

⁴TFRC is not yet practical and widely used due to several limitations [27].

⁵`Iperf` TCP traffic should not introduce any additional bottleneck links.

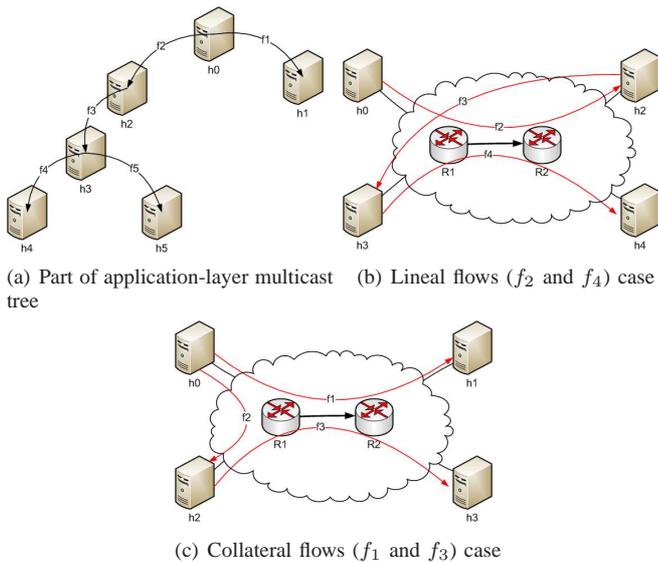


Fig. 3. Two cases of non-sibling flows sharing a bottleneck link. Bold arrows are bottleneck links in the direction

Note that “false positives” can occur where we decide that a flow f shares a bottleneck with flow g although this is not true in reality. However, across all PlanetLab experiments we cannot find a single case where non-sibling flows share a common bottleneck. Again, this is in accordance with our intuition that delay- and bandwidth-based tree construction approaches avoid such situations.

D. Cases of non-sibling flows sharing bottleneck links

We run more tests on random overlay trees which determine parent nodes for each joining peer *randomly* both in simulations and PlanetLab experiments. We are aware that such a tree construction mechanisms is unlikely to be used in practice. Yet, we want to confirm or refute our expectation that such an approach can cause non-sibling flows to share a common bottleneck link.

Fig.3(b) and 3(c) illustrate two general cases for all non-sibling flows share bottleneck links cases experienced during our experiments. One is lineal flows sharing bottleneck links case shown Fig.3(b) and the other one is collateral flows sharing bottleneck links case shown 3(c). Moreover, we find intuitive and practical methods to optimize the overlay tree for reducing the overall delay and to eliminate cases of non-sibling flows sharing a common bottleneck link. However, the details are out of the scope of this paper. We illustrate intuitive methods in Fig.4(a) and Fig.4(b) for cases of linear flows sharing bottleneck links, and in Fig.4(c) and Fig.4(d) for cases of collateral flows sharing bottleneck links.

E. Summary

We have found in this section that bottlenecks, i.e., links that fully utilize their available capacity, are generally only shared between sibling flows if at all. The following section will explain how to leverage this finding to design optimal, distributed, yet scalable rate allocation algorithm. Even if this

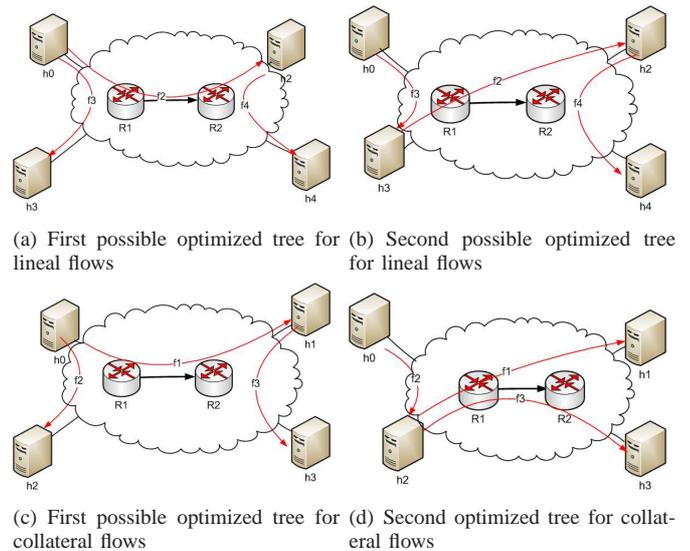


Fig. 4. Possible methods to avoid cases of non-sibling flows sharing bottlenecks while the overall delay is reduced and degree constraints is violated. Bold arrows are bottleneck links in the direction

assumption does not hold, our algorithms can outperform previously proposed solutions for rate allocation, see Section VII. Last but not least, we point out that there exist theoretical approaches for eliminating situations where non-sibling flows share a common bottleneck link [28].

IV. DISTRIBUTED AND OPTIMAL RATE ALLOCATION

Now, we propose a fully distributed network model and establish a formulation of the rate allocation problem, based on which we will develop our algorithms in Section V and VI. Importantly, we outline how to decompose the global problem into smaller pieces, thus paving the way for a distributed approach to solve the optimization problem. In contrast to existing work, we extend utility-pricing models with a new notion of TCP-friendliness (Section IV-B), before we formulate the optimization problem for maximizing the aggregate utilities of all receivers (Section IV-C).

A. Distributed Model

In the following we borrow some notations from the utility pricing model described in [7], [15], [8].

Consider an overlay network of $n + 1$ nodes, denoted as $H = \{h_0, h_1, \dots, h_i, \dots, h_n\}$. Node h_0 is the source of the multicast channel, while all other nodes are receivers of the multicast channel. We assume that the overlay tree has already been computed. Non-leaf nodes are forwarding data to their children and are able to scale-down the flows with techniques such as fine granular scalable coding/transcoding techniques [6]. The multicast channel consists of n end-to-end unicast flows, denoted as $F = \{f_1, \dots, f_i, \dots, f_n\}$. Flow f_{h_i} (or f_i , for simplicity’s sake) is the flow that terminates at h_i . Each flow $f_i \in F$ has a rate x_{f_i} (or x_i , for simplicity’s sake). We collect all the x_i into a rate vector $x = (x_i, f_i \in F)$. We denote $U_f(x_i)$ as the utility of flow f_i , when f_i transmits at rate x_i .

Let $I_f = [m_f, M_f]$ denote the rate range of flows. F'_h is the set of flows sent from h . If a host $h \in H$ is the destination of a flow f_h and the source of another flow $f'_h \in F'_h$, then f'_h is the child flow of f_h , denoted as $f_h \rightarrow f'_h$. We denote h' as the child of h and h^p as the parent node of h , i.e., $h^p \rightarrow h \rightarrow h'$.

For the remainder of this paper, we rely on the following two assumptions:

Assumption 1: U_f is strictly increasing and concave, and twice continuously differentiable on I_f .

Assumption 2: The curvatures of U_f are bounded away from zero on I_f : $-U''_f \geq 1/\kappa > 0$.

Formally, we define a *bottleneck link* as follows,

Definition 1: A link l is a **bottleneck link** for the time being if, and only if, $c_l = \sum_{f \in F(l)} x_f$. i.e., the link is fully utilized where the rate summation of all flows in the channel that go through the link l reaches the available bandwidth.

c_l is available bandwidth⁶ for the multicast channel at link l . Please note links and flows in our model are **unidirectional**. For each bottleneck link l , $F(l) = \{f \in F \mid l(f) = l\}$ is the set of flows in the channel that pass through it in the direction. Most flow f has a single bottleneck link at a particular point of time denoted as $l(f)$. We find in our experiments that flows rarely experience multiple bottleneck links at the same time. The location of the bottleneck links of f can be inferred by topology tools [29][30]. Bottleneck locations may change over time. However we assume that the bottleneck link locations are more stable than dynamics of flow traffic.

Now, suppose that the overlay network consists of L bottleneck links, denoted as $\Gamma = \{1, 2, \dots, L\}$. Note that a bottleneck link is a directed link and can be any link in the flow path and not necessary an access downlink or uplink. We store the c_l of all bottleneck links in vector $C = (c_l, l \in \Gamma)$.

Sibling flows are flows from the same node h , i.e., all $f'_h \in F'_h$, towards different children nodes. In accordance with our discussion of Section III bottleneck links are generally only shared between sibling flows, that means if $l(f_i) = l(f_j)$, then $(h_i)^p = (h_j)^p$.

To capture which flows are affected by which bottleneck links, we define a *bottleneck constraint matrix* A with dimensions $\Gamma \times F$. The value of its entries A_{lf} is 1 if flow f traverses the directed bottleneck link l , i.e., $l = l(f)$, $f \in F(l)$, otherwise the value is 0. Summing all flow rates of flows using a particular bottleneck link l cannot exceed the available capacity c_l of the link:

$$A \cdot x \leq C \quad (1)$$

We envision a distributed approach and algorithm to solve the rate allocation problem. Fortunately, we can generally decompose the set of inequalities in Equation (1) into smaller independent subsets that are local to individual end hosts. This is possible since bottleneck links are generally only shared by sibling flows. To this end, for a host h with children flows f'_h we define a matrix A^h with dimensions $\Gamma_h \times F'_h$, where Γ_h is

⁶The available bandwidth of bottleneck links can be measured in an end-to-end manner by tools such as *pathchar* and *pathrate*.

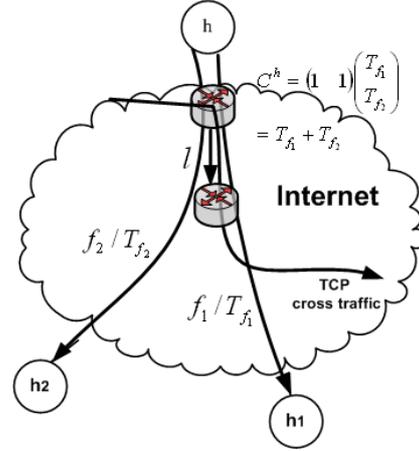


Fig. 5. Example of TCP-friendly available bandwidth for ALM. (a) Measure the unicast TCP-friendly throughput T_{f_1} for flow f_1 and T_{f_2} for flow f_2 going through bottleneck l . TCP-friendly available bandwidth for ALM is $C^h = A^h \cdot T^h = T_{f_1} + T_{f_2}$. (b) Re-allocate the rate for flows f_1 and f_2 while $x_1 + x_2 \leq T_{f_1} + T_{f_2}$ holds, thus the co-existing TCP traffic achieves no less throughput.

the set of bottleneck links for all f'_h flows. Again, the value $A^h_{l_h f'_h}$ is 1 if flow f'_h traverses the bottleneck link $l_h \in \Gamma_h$, otherwise the value is 0. Accordingly, we can decompose the problem as follows:

$$\begin{aligned} A^h \cdot x^{F'_h} &\leq C^h, \forall h \in H \\ \text{where } x^{F'_h} &= (x_{f'_h}, f'_h \in F'_h), C^h = (c_l, l \in \Gamma_h) \forall h \in H. \end{aligned} \quad (2)$$

While Equation (1) and Equation (2), respectively, ensure that the sum of all flow rates for a bottleneck link does not exceed the available capacity, we now add a second class of constraints. Evidently, the sending rates of children nodes cannot be higher than the rate at which they receive traffic from their parents. To formalize this, we define the *data constraint* or *flow preservation matrix* B with dimensions $F \times F$: its entries $B_{f_1 f_2}$ are -1 if f_1 is a child flow of f_2 (i.e., $f_1 = f'_2 \in F'_2$), are 1 if f_1 is f_2 and has a parent flow, and are 0 otherwise. Using this notation, the constraint can be described as follows:

$$B \cdot x \leq 0 \quad (3)$$

Table I summarizes the introduced variables and notations. Moreover, an example in Appendix A illustrates the model we have defined in this section for the rate allocation problem.

B. TCP-friendliness For Co-existing Traffic

The model described so far does not consider the fact that there may exist unicast TCP traffic and other multicast sessions using the same physical links. Now, we describe how to extend our current problem formulation to include a new notion of TCP friendliness for application-layer multicast, which we define as follows:

Definition 2: A rate control algorithm is *TCP-friendly*, if and only if any co-existing TCP traffic does not obtain less

throughput than it would if all flows of the application-layer multicast channel were using unicast TCP as congestion control algorithm [16].

Let $T_{f'_h}^h$ be the TCP-friendly available bandwidth for the unicast flow f'_h at the bottleneck link $l(f'_h)$ determined by a TFRC algorithm. We simply collect all $T_{f'_h}^h$ in vector $T^h = (T_{f'_h}^h, f'_h \in F'_h)$. Hence, we obtain C^h , the vector of TCP-friendly available bandwidths for the multicast channel for F'_h , as $C^h = A^h \cdot T^h$. Then, the constraints of Equation (2) look as follows:

$$\begin{aligned} A^h \cdot x^{F'_h} &\leq A^h \cdot T^h, \forall h \in H \\ \text{where } x^{F'_h} &= (x_{f'_h}, f'_h \in F'_h), \forall h \in H \end{aligned} \quad (4)$$

Figure 5 illustrates our approach to add TCP-friendliness. Assume two sibling flows f_1 and f_2 in the multicast tree sharing bottleneck link l . In addition, there is a TCP cross flow that shares the link l . With our TCP-friendliness for ALM model, we determine the unicast TCP-friendly throughput T_{f_1} for flow f_1 and T_{f_2} for flow f_2 by TFRC algorithm which allow the TCP cross flow getting its fair bandwidth. TCP-friendly available bandwidth for this multicast tree is the sum of T_{f_1} and T_{f_2} . Our distributed rate allocation approaches may re-allocate the rate between flow f_1 and flow f_2 , while the co-existing TCP traffic achieves no less throughput than its fair bandwidth.

C. Optimization Problem

Finally, we formulate the rate allocation problem to maximize the aggregate utility, i.e., the overall utilities of all flows in the application-layer multicast tree:

$$\max_{m_f \leq x_f \leq M_f} \sum_{f \in F} U_f(x_f) \quad (5)$$

fulfilling the following distributed/local constraints:

$$\begin{cases} A^h \cdot x^{F'_h} \leq A^h \cdot T^h \\ B \cdot x \leq 0 \end{cases}$$

where $x^{F'_h} = (x_{f'_h}, f'_h \in F'_h), \forall h \in H$.

V. ALGORITHMS

We first present a typical dual and then our novel primal approach for the optimization problem in Section IV-C.

A. Dual Algorithm

The literature suggests pricing models [7], [15], [13], [8] to maximize the aggregate utility across all receivers. Our goal is to develop a dual algorithm that is distributed, i.e., that leverages the fact that we have decomposed the global problem in the preceding section and have formulated local constraints. For details on our proposed dual algorithm, we refer the reader to Appendix B.

First we obtain the maximizer by applying the Kuhn-Tucker theorem:

$$x_f(\mu^\alpha, \mu^\beta) = [U_f'^{-1}(\mu_{l(f)}^\alpha) + \mu_f^\beta - \sum_{f \rightarrow f'} \mu_{f'}^\beta]_{m_f}^{M_f} \quad (6)$$

Initialization

sending data with the data constraints applied
unicast TCP-friendly rate for each flow.

Link Price Update

(by bottleneck link $l = l(f'_i) \in \Gamma_i$) at $t = 1, 2, \dots$:

update price of the bottleneck link:

$$\mu_{l(f'_i)}^\alpha(t+1) = [\mu_{l(f'_i)}^\alpha(t) + \gamma(\sum_{f'_i \in F(l)} x_{f'_i}(t) - c_{l(f'_i)})]^+$$

Relay Price Update (by flow $f'_i \in F'_{h_i}$) at $t = 1, 2, \dots$:

update relay price of f'_i with updated flow rate $x_{f'_i}$:

$$\mu_{f'_i}^\beta(t+1) = [\mu_{f'_i}^\beta(t) + \gamma(x_{f'_i}(t) - x_{f_i}(t))]^+$$

Flow Rate Adaptation (by flow $f'_i \in F'_{h_i}$, At $t = 1, 2, \dots$

1 receive relay prices $\mu_{f'_j}^\beta(t)$

from all children flow $\{f_j \mid f'_i \rightarrow f_j\}$

2 calculate:

$$\lambda_{f'_i}^\alpha(t) = \mu_{l(f'_i)}^\alpha(t)$$

$$\lambda_{f'_i}^\beta(t) = \mu_{l(f'_i)}^\beta(t) - \sum_{f_j \rightarrow f'_i} \mu_{f_j}^\beta(t)$$

3 adjust rate:

$$x_{f'_i}(t+1) = [U_{f'_i}'^{-1}(\lambda_{f'_i}^\alpha(t) + \lambda_{f'_i}^\beta(t))]_{m_{f'_i}}^{M_{f'_i}}$$

communicates with the rate $x_{f'_i}(t+1)$ for flow f'_i

4 send $\mu_{f'_i}^\beta(t+1)$ to $(h_i)^P$

TABLE II
DUAL ALGORITHM OF END HOST h_i

$\mu^\alpha = (\mu_l^\alpha, l \in \Gamma)$ and $\mu^\beta = (\mu_f^\beta, f \in F)$ are vectors of Lagrangian multipliers. μ_l^α can be understood as the link price of bottleneck link l and μ_f^β can be understood as the relay price that f must pay its parent flow. We solve the dual problem using the gradient projection method.

The dual algorithm for each end host is presented in Table II. We assume the network is synchronous, i.e., updates occur in rounds at times $t = 1, 2, \dots$. Each end host h is capable of communicating with neighbors, of measuring A^h , T^h , and of computing and adapting the sending rate for each flow f'_h (i.e., sender-based flow). The bottleneck link price is locally calculated by end host with A^h , T^h and the sending rates of sibling flows sharing the bottleneck. End host h first calculates the relay price with the updated flow rate x_{f_h} . Then, end host h calculates the data price of flow f'_h with the updated relay benefit from its children nodes. End host h adapts the sending rate for each flow f'_h in the final step. Therefore, the algorithm solely depends on the coordination of end hosts, and avoids changing the existing infrastructure. We allocate the unicast rate for each flow and then apply data constraints for the initial rate in the algorithm, i.e., $x_{f'_h}(0) = T_{f'_h}^h$.

B. Primal Algorithm - A Feasible Direction Algorithm

Besides the typical dual approach, we propose a novel primal approach in this paper. Primal algorithms work on the original problem directly by searching the feasible region for an optimal solution. Thanks to our fully distributed model, directly solving the optimization problem (5) only requires coordination among those sibling flows sharing bottleneck links.

The proposed novel primal algorithm is a feasible direction method [31]. The primal algorithm continues to move in

TABLE I
SUMMARY OF NOTATIONS IN THE MODEL

Notation	Definition
$h \in H = \{h_0, h_1, \dots, h_n\}$	End Host
$h^p \rightarrow h \rightarrow h' \in H'_h$	h^p is the parent node of h , h' is a child of h
H'_h	Set of children of h
$f \in F = \{f_1, f_2, \dots, f_n\}$	Unicast flow in ALM channel
$f_i \rightarrow h_i$	Flow f_i terminated at h_i
x_i	Flow rate of $f_i \in F$
$l \in \Gamma$	Bottleneck link l
$\Gamma = 1, 2, \dots, L$	Set of bottleneck links
$c_l \in C, l \in \Gamma$	Available bandwidth for the channel of bottleneck link l
$f_i \rightarrow f'_i$	f'_i is a child flow of f_i
$F'_h = (f'_h)$	Set of flows sent from h in the channel
$\Gamma_h = \{l_h l(f'_h), f'_h \in F'_h\}$	Set of bottlenecks of flow f'_h
$l(f) \in \Gamma$	The bottleneck link that f goes through
$F(l)$	Set of siblings flows that go through bottleneck link l
$A = (A_{lf})_{L \times F}$	Bottleneck constraint matrix
$B = (B_{f'f})_{F' \times F}$	Data constraint matrix
$A^h = (A_{lf})_{\Gamma_h \times F'_h}$	Bottleneck constraint matrix of F'_h
$T^h_{f'_h}$	TCP-friendly available bandwidth for unicast for f'_h at $l(f'_h)$
T^h	Collection of $T^h_{f'_h}$ for $f'_h \in F'_h$
$C^h = A^h \cdot T^h$	Vector of TCP-friendly available bandwidth for ALM channel for F'_h
$I_f = [m_f, M_f]$	Feasible range of $U_f(x_f)$
$U_f(x_f)$	Utility function of streams at rate x_f
$x^{F'_h} = (x_{f'_h}, f'_h \in F'_h)$	Flow rate set of F'_h

finite steps in the determined direction, obtaining a new and better rate allocation. The process is repeated until the optimal utility is reached. We point out that each rate allocation computed in this process is feasible. Moreover, the aggregate utility continuously improves. In general, the convergent rate allocation is the global maximum of the convex optimization problem. However, it is a constrained local maximum of a general (non-convex) problem (see Chapter 11.1 in [31]). Therefore, the primal algorithm is more general and it does not require the convex property of the application utility.

Definition 3: The *data shadow price* of a flow is the change in the aggregate utility of the flow itself and its subtree by relaxing the data constraint by one unit (a small move).

We name a flow a data constrained flow f if it is actively constrained by its parent flow, i.e., $x_f = x_{f^p}$; otherwise it is a data unconstrained flow, i.e., $x_f < x_{f^p}$ and actively constrained by its bottleneck link.

For a data constrained leaf flow f the data shadow price is:

$$p_f = \Delta U_f / \Delta x_f = U'_f(x_f) \quad (7)$$

For a data constrained intermediate flow f , the data shadow price is:

$$p_f = U'_f(x_f) + \sum_{f' \in F'} p_{f'} \quad (8)$$

When a flow f is data unconstrained flow, its data shadow price is zero ($p_f = 0$). For example, the data shadow price of each dashed line flow in Figure 6 is zero.

We call a node a data constrained node (Figure 6(b)) if its incoming flow is a data constrained flow, otherwise it is a data unconstrained node (Figure 6(a)). In our primal

algorithm, data shadow price of a flow is computed by its receiver using its data constraint information and data shadow price of children flows. The data constraint information from the parent node is a 1-bit information and indicates the flow and whether nodes are data constrained or not. We assume the network is synchronous such that updates occur in rounds at times $t = 1, 2, \dots$. Again, each end host h is capable of communicating with neighbors, of measuring A^h , T^h , and of computing and adapting the sending rate for each flow f'_h (i.e., sender-based flow). We choose the data constraints applied TCP/TFRC rate as the initial rate in the algorithm, i.e., $x_{f'_h}(0) = T^h_{f'_h}$. The closer the initial rate is to the optimal rate, the faster the algorithm converges to the optimal rate.

We present the primal algorithm of an intermediate node in Table III. The algorithm purely depends on the coordination of neighboring nodes. Each node receives the data shadow prices from its children nodes. The algorithm reallocates the bandwidth of the bottleneck link with step size γ from children flows with lower shadow prices to children flows with higher shadow prices such that the available bandwidth constraints are not violated and flows with higher data shadow price get more bandwidth; and thus obtain a better rate allocation after each step with an improved aggregate utility. Then, each node sends an update of its data shadow price to its parent node, and sends an update of the flow rate and the data constraint information for each children flow to its children. In the following we discuss the termination of the primal algorithm.

Theorem 1: For any application-layer multicast session, the rate allocation by the primal algorithm in Table III with sufficient small step size γ converges to the optimal allocation at most in $\sum_{l \in \Gamma} (c(l)/\gamma)$ steps.

initialization

sending data with the data constraints applied
unicast TCP-friendly rate for each flow.

update the data shadow price from children

get shadow price for children flows $f'_i: p_{f'_i}(t), f_j \in F'_{h_i}$
compute the median shadow price of children flows
for each bottleneck links: $p_{j_{med}}(t)$

update information from the parent node

get the flow rate $x_i(t)$ and the data constraint information

re-allocate the rate among the children flows for $f_j \in F'_{h_i}$

for f_j sharing the same bottleneck $j := 1$ to n do

if $p_{f'_j}(t) > p_{j_{med}}(t)$

$x_j(t+1) = x_j(t) + \gamma$

else if $p_{f'_j} < p_{j_{med}}$

$x_j(t+1) = x_j(t) - \gamma$

end if

end if

update Data Shadow Price to parent node

for data constrained node $h_i: p_{f'_i}(t+1) = U'_f(x_i)$

for f_j sharing the same bottleneck $j := 1$ to n do

if $x_j(t) \geq x_i(t)$

$x_j(t+1) = x_i(t); p_{f'_i}(t+1) = p_{f'_i}(t+1) + p_{f'_j}(t)$

else $x_j(t+1) = x_j(t)$

end if

for data unconstrained node $h_i: p_{f'_i} = 0$

send $p_{f'_i}(t+1)$ up to parent node h_i^p

Update Streaming and information to children

for $f_j \in F'_{h_i}$

stream media to child j with updated rate $x_j(t+1)$

update the data constraint information and $x_j(t+1)$ to h_j

TABLE III

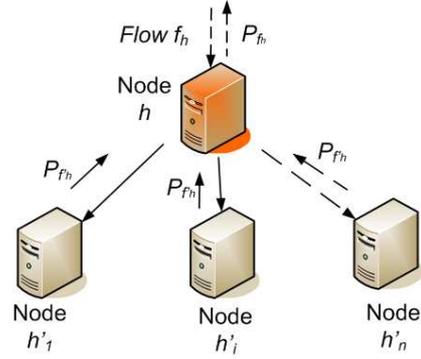
PRIMAL ALGORITHM OF END HOST h_i

Proof: For the subtree rooted at end host h denoted as $Tree(h)$, given the receiving rate f_h , the primal algorithm improves the aggregated utility by re-allocating the rate of shared bottleneck. In particular, the primal algorithm relocates the rate for each shared bottleneck link from children flows with low data shadow prices ($f'_h, p_{f'_h} < p_{h'_{med}}(t)$) to flows with high data shadow prices ($f'_h, p_{f'_h} > p_{h'_{med}}(t)$). Thus,

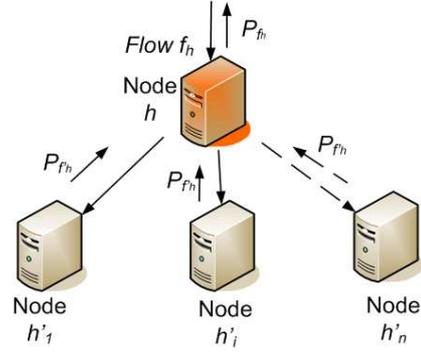
$$\sum_{i \in Tree(h)} U_f(x_{f_i}(t)) < \sum_{i \in Tree(h)} U_f(x_{f_i}(t+1))$$

Each allocation generated in the primal algorithm process is feasible and the value of the aggregate utility of the subtree $\sum U(x_{f_i}(t))$ improves constantly. Given the receiving rate f_h and the concave utility, there is a limit for the aggregate utility of the subtree. The algorithm will finally converge to the maximum point. For a convex optimization problem, the convergent rate allocation is the global maximum (the optimality) of the subtree (see Chapter 11.1 in [31]). Since each subtree iteratively converges to its local optimality for given receiving rate, the entire multicast tree rooted at h_0 will be eventually optimal. Each bottleneck link will move at most $c(l)/\gamma$ steps to reallocate the rate among the sibling flows, thus the algorithm will converge at most in $\sum_{l \in \Gamma} (c(l)/\gamma)$ steps. ■

In contrast to dual approaches [15], [32], a feasible direction algorithm shows steady convergence towards the optimal state.



(a) data unconstrained node h



(b) data constrained node h

Fig. 6. Nodes and flows in the primal algorithm (dashed line means data unconstrained flow, data constrained flow otherwise).

VI. TOWARDS ALGORITHMS IMPLEMENTATION IN ASYNCHRONOUS NETWORK

Both the dual algorithm and the primal algorithm assume a synchronized network environment. This means that relay price and flow rate updates in the dual algorithm, as well as the flow rate and data shadow price updates in the primal algorithm, occur in rounds. In this section, we describe how to deal with an asynchronous environment, where different peers and flows update their data shadow prices, sending rates, or prices at arbitrary times.

A. Asynchronous Model

For our work we use the asynchronous model as introduced in [7], [15]. Let $\tilde{T} = \{0, 1, 2, \dots\}$ be the set of time instances at which either a flow rate, data shadow price, or a relay price is updated. We define:

- 1) $\tilde{T}_f \subseteq \tilde{T}$ —the set of time instances at which a flow f (its sender) updates its rate x_f .
- 2) $\tilde{T}_f^\beta \subseteq \tilde{T}$ —the set of time instances at which a flow f (its sender) updates its relay price μ_f^β .
- 3) $\tilde{T}_f^d \subseteq \tilde{T}$ —the set of time instances at which a flow f (its receiver) updates its data shadow price.

We further define T is a time window size such that the time between any consecutive updates is bounded by T for each kind of updates.

B. Asynchronous Dual Algorithm

Link price update: End host h locates the bottleneck links $l(f'_h) \in \Gamma_h$, measures A^h and C^h , and adapts rates $x_{f'_h}(t)$. The bottleneck link price is calculated as in the synchronous model. End host h locally computes the link price $\mu_{l(f'_h)}^\alpha(t)$ with rates $x_{f'_h}(t)$ for sibling flows sharing the bottleneck l , namely $f'_h \in F_h(l)$ and C^h using Equation (29) in Appendix B.

Relay price update: End host h keeps track of all recent rate updates of flows $x_{f_h}(t')$, $(t-T) \leq t' \leq t$ and $t' \in \tilde{T}_{f_h}$ and computes its estimated rate $\hat{x}_{f_h}(t)$ by using a weighted average of these values:

$$\hat{x}_{f_h}(t) = \sum_{t'=t-T}^t b_{f_h}(t', t) x_{f_h}(t'), \quad \sum_{t'=t-T}^t b_{f_h}(t', t) = 1. \quad (9)$$

Then, end host h using this estimated rate for flow f_h computes the relay price for f'_h , namely $\mu_{f'_h}^\beta(t+1)$, as in Equation 30.

Flow rate update: End host h collects all recent relay price updates from children nodes $\mu_{f'_h}^\beta(t)$, where $(t-T) \leq t' \leq t$ and $t' \in \tilde{T}_{f'_h}^\beta$. To update the flow rate for flow f'_h , we first compute the prices $\mu_{l(f'_h)}^\alpha(t)$ of all bottleneck links it traverses, and the estimated prices $\hat{\mu}_{f'_h}^\beta(t)$ of all its children flows, where $f'_h \rightarrow f'_{h'}$.

$$\hat{\mu}_{f'_h}^\beta(t) = \sum_{t'=t-T}^t b_{f'_h, f'_h}^\beta(t', t) \mu_{f'_h}^\beta(t'), \quad \sum_{t'=t-T}^t b_{f'_h, f'_h}^\beta(t', t) = 1. \quad (10)$$

First end host h computes the prices $\mu_{l(f'_h)}^\alpha(t)$ of bottleneck links that flow f'_h traverses, and with $\mu_{l(f'_h)}^\alpha(t)$ end node h calculates $\lambda_{f'_h}^\alpha(t)$. With estimated $\hat{\mu}_{f'_h}^\beta(t)$ and $\mu_{f'_h}^\beta(t)$, end host calculates $\lambda_{f'_h}^\beta(t)$, and further computes children flow rates according to Equations (23).

Algorithm: To upgrade the synchronous dual algorithm to an asynchronous one, we estimate the flow rate $\hat{x}_{f_h}(t)$ in Equation (9) to compute the relay price during step ‘‘Relay Price Update’’, and estimate the relay price $\hat{\mu}_{f'_h}^\beta(t)$ in Equation (10) to compute the sending rate $x_{f'_h}(t+1)$. Now, one can easily extend the synchronous dual algorithm in Table II to an asynchronous one. We omit presenting the details of the asynchronous dual algorithm.

C. Asynchronous Primal Algorithm

Data shadow price update from children: End host h collects all recent data shadow price updates $p_{f'_h}(t')$ of its children flows f'_h , where $(t-T) \leq t' \leq t$ and $t' \in \tilde{T}_{f'_h}^d$, and computes its estimated data shadow price $\hat{p}_{f'_h}(t)$ by using a weighted average of these values:

$$\hat{p}_{f'_h}(t) = \sum_{t'=t-T}^t a_{f'_h}(t', t) p_{f'_h}(t'), \quad \sum_{t'=t-T}^t a_{f'_h}(t', t) = 1. \quad (11)$$

Flow rate update from parent node End host h collects all recent flow rate updates of flow $x_{f_h}(t')$, $(t-T) \leq t' \leq t$ and $t' \in \tilde{T}_{f_h}$, computes its estimated rate $\hat{x}_{f_h}(t)$ by using a weighted average of these values:

$$\hat{x}_{f_h}(t) = \sum_{t'=t-T}^t b_f(t', t) x_{f_h}(t'), \quad \sum_{t'=t-T}^t b_f(t', t) = 1. \quad (12)$$

Algorithm: To upgrade the synchronous primal algorithm to an asynchronous one, we only need to estimate data shadow prices $\hat{p}_{f'_h}(t)$ in Equation (11) to redistribute the rates among children and estimate the parent flow rate $\hat{x}_{f_h}(t)$ in Equation (12) to compute its shadow price. We again omit presenting the details of the asynchronous primal algorithm.

D. Weighted Average Policies

The weighted average policy in the model is very general. In particular, we implement two popular policies for both the primal algorithm and the dual algorithm:

- latest update only: only the most recently received data in $(t-T) \leq t' \leq t$ is used for the estimation.
- update average: all data in time window $(t-T) \leq t' \leq t$ are averaged for the estimation.

Throughout the remainder of this paper, we will rely on the ‘‘update average’’ policy.

VII. EVALUATION

After presenting our distributed rate allocation algorithms, we now evaluate the performance in terms of their ability to optimize the overall utility (Section VII-B), in terms of their convergence behavior (Section VII-C), and finally in terms of the number of messages that are exchanged (Section VII-D). To this end, we present an implementation of our asynchronous algorithms (Section VII-A) and compare it against Cui’s algorithm [15] for which we received an implementation from the authors.

A. Evaluation Setup

The main focus of this paper is to study distributed and optimal rate allocation from a theoretical perspective, not to present an actual implementation of a real multicast system. Yet, we have implemented and tested our asynchronous algorithms in a simulation environment.

For this purpose, we rely on BRITE [17] to generate network topologies consisting of 1,000 routers. This topology size appears sufficient to study and compare the performance of our algorithms against existing approaches. The available bandwidth of all links is uniformly distributed between 10 Mbps and 1,000 Mbps and the average delay is 0.6 ms. Overall, we create 10 different network topologies.

As described in Section III-B, we design and implement a mechanism to construct multicast trees on top of a given physical topology for our experiments. Our designed tree construction mechanism is delay-based and takes into account bandwidth requirements and degree constraints. For every physical topology, we vary the number of nodes in the

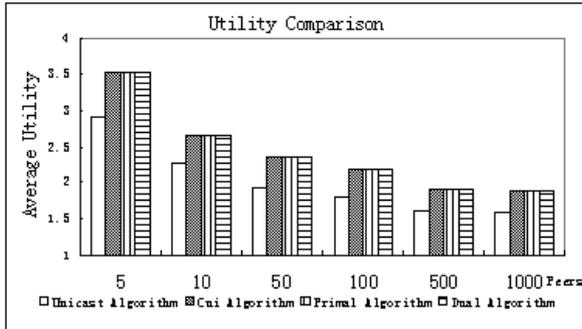


Fig. 7. Average utilities.

multicast trees between 5 and 1,000 nodes and repeat the experiment. The available bandwidth for the access links of multicast nodes are limited to 100 Mbps and are uniformly distributed between 1 Mbps and 100 Mbps. There was no single instance across our experiments where non-sibling flows share a bottleneck link. Our delay- and bandwidth-based tree construction avoids such situations.

As utility function we choose $U_f(x_f) = \ln(x_f)$, which is concave and strictly increasing for x_f . Intuitively, this reflects the fact a user can benefit from a rate increase from 1 to 2 Mbps much more than from a rate increase from 10 to 11 Mbps. As a matter of fact, our distributed model can easily be adapted to support other utility functions, e.g., the utility function of TCP [9] or TFRC). Generally, we assume that a technology such as MPEG-4 Fine-Grained Scalable Video Streams [6] is used to adapt sending rates.

To study the performance of the suggested algorithms, we choose an average update interval of 10 ms. The average measurement interval for detecting bottleneck links is set to 1,000 ms, as we assume a slowly time-varying asynchronous network where network dynamics are slower than the dynamics of flow traffic. Overall, this results in low overhead for measuring the bottlenecks. Finally, we set the step size γ to 0.0005 Mbps and time window T for the weighted average updates to 50ms for both the primal algorithm and the dual algorithm in the experiments.

B. Utility

We start by studying the ability of various algorithms to maximize the aggregate utility perceived by all multicast receivers. In this context, we compare our distributed primal and dual solutions against two other approaches. The standard unicast algorithm allocates the rates independently as unicast flows using the TCP/TFRC algorithm, and then applies the data constraint as described by Equation (3). Moreover, we apply Cui's algorithm in its corrected version, see [33]. In all cases, we obtain a set of rates and can determine for every experiment the aggregate utility across all receivers.

Figure 7 summarizes the achieved utilities, namely the average across all our experiments. We find that both the primal and the dual algorithm result in the optimal utility which is the same utility as Cui's algorithm. Yet, our approach

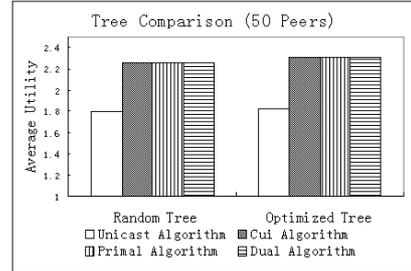


Fig. 8. Average Utility for Assumption Violated Trees

by design is TCP-friendly to cross traffic and incurs less overhead as we will see in the next subsection. The standard unicast approach is far from achieving the same utilities as the other three approaches, emphasizing the benefits of a distributed rate allocation approach. Please note rate allocation is proportionally fair for flows in the multicast tree. Moreover, we find that the minimum rates allocated by our algorithms no less than those obtained by a standard unicast approach in our experiments .

We assume that bottleneck links are in the vast majority of cases only shared between sibling flows, see Section III. Nevertheless, to study how well our algorithms maximize utility if this assumption is violated, we do the following: instead of minimizing delays, we construct random multicast trees for our physical topologies. By doing so, we obtain a few cases where non-sibling flows share a common bottleneck, and then apply our algorithms. Even for such adverse overlay topologies, our algorithms as shown in Figure 8 achieve aggregate utilities that are only marginally worse than those obtained in an optimized tree after eliminating non-sibling flows sharing a bottleneck link [28]. This small degradation is due to the fact that non-sibling flows are regarded as co-existing traffic that is not part of the multicast stream. Yet, even in such adverse scenarios our algorithms significantly outperform those of standard unicast algorithms.

C. Convergence Time

Convergence behavior of unicast congestion control algorithms have been extensively studied in the literature[7], [8], [34]. Now, we analyze for various step sizes the time that our algorithms take to converge to a stable allocation of rates. Our primary interest is not to study the absolute convergence times, but rather to compare convergence behavior of the primal and dual algorithm. Since Cui's algorithm is a dual algorithm with the same convergence properties as our proposed dual algorithm, we refrain from presenting results for Cui's algorithm.

Figure 9 shows that the distributed allocation algorithms converge to an optimal rate when new nodes join the multicast tree at time $t = 10s$ and $t = 40s$. Moreover, we observe:

- 1) The dual algorithm introduces oscillations during the convergence process. This leads to high link prices (i.e., high packet loss rates). The primal algorithm results in good rate allocations, avoiding oscillations during the convergence process.

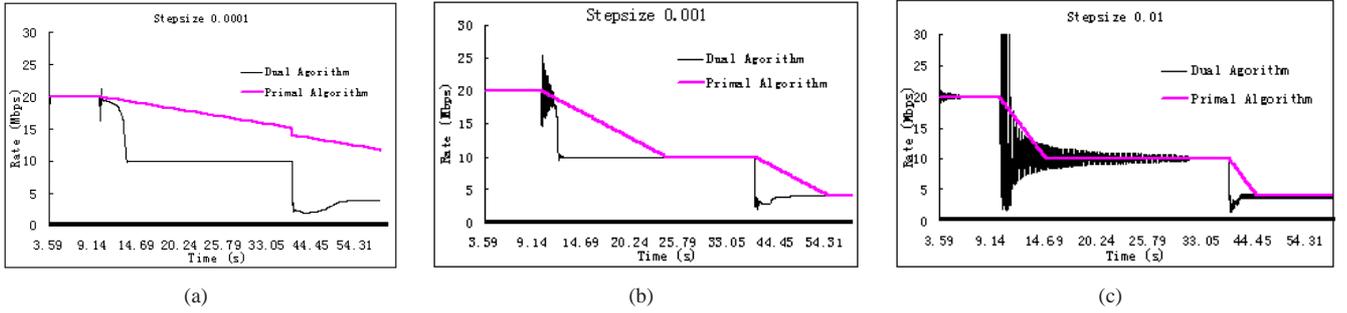


Fig. 9. Convergence behavior for various step sizes. Nodes join at $t = 10s$ and $t = 40s$

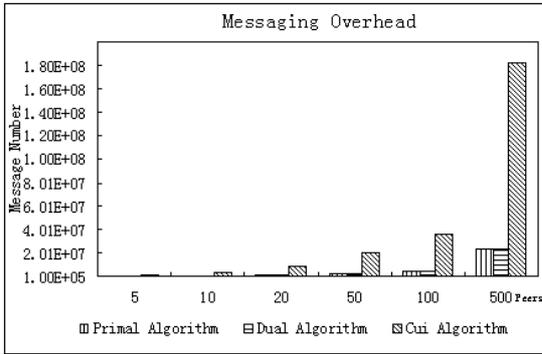


Fig. 10. Number of exchanged messages.

- 2) The primal algorithm converges more slowly than the dual algorithm in our experiment setting. However, the convergence rate of the primal algorithm is restricted, see Theorem 1. Note that the primal algorithm in Figure (9a) also converges to the optimal rate, but only after a certain time, which is not shown in this plot.
- 3) Large step sizes result in faster convergence than small step sizes. However, large step sizes may lead to oscillation and instability, in particular for the dual algorithm as shown in Figure (9c).

D. Messaging Overhead

Section VII-B reveals that our dual and primal solution indeed maximize the aggregate utility of all receivers. Now, we show that this can be achieved with low message overhead compared to Cui’s algorithm.

To this end, we rely on the same experiments as in the preceding sections and count the number of messages that are exchanged within a time period of 600s. Figure 10 presents the mean number of exchanged messages for various multicast tree sizes and compares our primal and dual solutions against Cui’s algorithm.

As expected the number of messages is the same for the primal and dual variants of our algorithms. Overall, we find that our algorithms produce significantly fewer messages than Cui’s algorithm, but as shown in Section VII-B, achieves similar results for the aggregate utility. For multicast trees with 500 nodes, there are more than 180 million messages

on average for Cui’s algorithm while this number is only approximately 20 million for our algorithms.

Generally, the fewer bottleneck links are traversed by the flows in the underlying physical topology, the more our algorithms reduce the number of exchanged messages compared to Cui’s algorithm. Actually, we find that most price messages in Cui’s algorithm include link prices with a value of zero. These are cases, where our algorithms avoids sending a message.

VIII. CONCLUSION

In this paper, we formalize and solve the problem of optimal rate allocation for a given application-layer multicast tree as a convex optimization problem with constraints. Moreover, our observation that bottlenecks are generally shared between sibling nodes in a multicast tree, allows to formulate localized constraints for distributed algorithms. Our proposed model is TCP-friendly to co-existing traffic in the network. We further propose a novel primal approach and a typical dual approach, and design the according algorithms to solve the optimization problem. Our experiments reveal that both proposed algorithms are optimal in terms of global utility and are fully distributed with a very small messaging overhead. We point out that our algorithms can be used for multi-tree multicast systems, by applying our algorithms for each tree individually.

A possible topic for future work is to derive a mathematical description of convergence rates for the two proposed algorithms in asynchronous environments. Finally, we plan to integrate the proposed algorithms into large application-layer multicast systems to test their performance.

IX. ACKNOWLEDGMENTS

We thank Dr. Y. Cui for providing us his implementation and Mr. H Huang for his discussion and help with implementing the algorithms.

REFERENCES

- [1] Y. Chu, S. G. Rao, and H. Zhang, “A Case for End System Multicast (keynote address),” in *SIGMETRICS ’00: Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, New York, NY, USA, 2000, pp. 1–12, ACM.
- [2] J. Liu, S. G. Rao, B. Li, and H. Zhang, “Opportunities and Challenges of Peer-to-Peer Internet Video Broadcast,” in *In (invited) Proceedings of the IEEE, Special Issue on Recent Advances in Distributed Multimedia Communications*, 2007.

- [3] M. Hosseini, D.T.Ahmed, S. Shirmohammadi, and N.D. Georganas, "A Survey of Application-Layer Multicast Protocols," *IEEE in Communications Surveys and Tutorials*, vol. 9, no. 3, pp. 58–74, 2007.
- [4] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *SIGCOMM '00: Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, New York, NY, USA, 2000, ACM.
- [5] Peercast.org, "p2p broadcasting for everyone," <http://www.peercast.org>.
- [6] W. Li, "Overview of fine granularity scalability in MPEG-4 video standard," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 11, no. 3, pp. 301–317, 2001.
- [7] S. H. Low and D. E. Lapsley, "Optimization flow control—I: basic algorithm and convergence," *IEEE/ACM Trans. Netw.*, vol. 7, no. 6, pp. 861–874, 1999.
- [8] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate Control for Communication Networks: Shadow Prices, Proportional Fairness and Stability," *Journal of the Operational Research Society*, vol. 49, 1998.
- [9] H. Low, F. Paganini, and J. C. Doyle, "Internet Congestion Control," *IEEE Control Systems Magazine*, vol. 22, pp. 28–43, 2002.
- [10] R. Srikant, *The Mathematics of Internet Congestion Control (Systems and Control: Foundations and Applications)*, SpringerVerlag, 2004.
- [11] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast," in *SIGCOMM '96: Conference proceedings on Applications, technologies, architectures, and protocols for computer communications*, New York, NY, USA, 1996, pp. 117–130, ACM.
- [12] B. Akbari, H. R. Rabiee, and M. Ghanbari, "An optimal discrete rate allocation for overlay video multicasting," *Comput. Commun.*, vol. 31, no. 3, pp. 551–562, 2008.
- [13] K. Kar, S. Sarkar, and L. Tassioulas, "Optimization Based Rate Control for Multirate Multicast Sessions," in *IEEE INFOCOM*, 2000, pp. 123–132.
- [14] S. Deb and R. Srikant, "Congestion Control for Fair Resource Allocation in Networks with Multicast Flows," *IEEE/ACM Trans. Netw.*, vol. 12, no. 2, pp. 274–285, 2004.
- [15] Y. Cui, Y. Xue, and K. Nahrstedt, "Optimal Resource Allocation in Overlay Multicast," *IEEE Trans. Parallel Distrib. Syst.*, vol. 17, no. 8, pp. 808–823, 2006.
- [16] J. Yan, M. May, and B. Plattner, "Optimal TCP-Friendly Rate Control for P2P Streaming: An Economic Approach," in *IWSOS '09: Proceedings of the 4th IFIP TC 6 International Workshop on Self-Organizing Systems*, Berlin, Heidelberg, 2009, pp. 113–124, Springer-Verlag.
- [17] A. Medina, I. Matta, and J. Byers, "BRITE: A Flexible Generator of Internet Topologies," Tech. Rep., Boston University, Boston, MA, USA, 2000.
- [18] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller, "Construction of an Efficient Overlay Multicast Infrastructure for Real-Time Applications," in *In Proceedings of IEEE INFOCOM*, 2003, pp. 1521–1531.
- [19] S. S. Jonathon and J. S. Turner, "Routing in Overlay Multicast Networks," in *In Proceedings of IEEE INFOCOM*, 2002, pp. 1200–1208.
- [20] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable Application Layer Multicast," in *SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, New York, NY, USA, 2002, pp. 205–217, ACM.
- [21] E. Brosh, A. Levin, and Y. Shavitt, "Approximation and Heuristic Algorithms for Minimum-Delay Application-Layer Multicast Trees," *IEEE/ACM Trans. Netw.*, vol. 15, no. 2, pp. 473–484, 2007.
- [22] M. L. Castro, P. Druschel, Y. C. Hu, and A. Rowstron, "Proximity Neighbor Selection in Tree-Based Structured Peer-to-Peer Overlays," Tech. Rep. MSR-TR-2003-52, Microsoft Research, Boston, MA, USA, 2003.
- [23] D. Pendarakis, S. Shi, D. Verma, and M.I. Waldvogel, "ALMI: An Application Level Multicast Infrastructure," in *USITS'01: Proceedings of the 3rd conference on USENIX Symposium on Internet Technologies and Systems*, Berkeley, CA, USA, 2001, pp. 5–5, USENIX Association.
- [24] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distributing Streaming Media Content Using Cooperative Networking," in *NOSSDAV '02: Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video*, New York, NY, USA, 2002, pp. 177–186, ACM.
- [25] M. Kwon and S. Fahmy, "Topology-Aware Overlay Networks for Group Communication," in *NOSSDAV '02: Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video*, New York, NY, USA, 2002, pp. 127–136, ACM.
- [26] "PlanetLab," <http://www.planet-lab.org>.
- [27] I. Rhee and L. Xu, "Limitations of equation-based congestion control," *Networking, IEEE/ACM Transactions on*, vol. 15, no. 4, pp. 852–865, aug. 2007.
- [28] M. S. Kim, Y. Li, and S. S. Lam, "Eliminating bottlenecks in overlay multicast," in *In Proceedings of IFIP Networking 2005*, 2005, pp. 893–905.
- [29] T. Bu, N. Duffield, F. L. Presti, and D. Towsley, "Network tomography on general topologies," in *SIGMETRICS '02: Proceedings of the 2002 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, New York, NY, USA, 2002, pp. 21–30, ACM.
- [30] M. S. Kim, T. Kim, Y. J. Shin, S. S. Lam, and E. J. Powers, "A Wavelet-Based Approach to Detect Shared Congestion," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 4, pp. 293–306, 2004.
- [31] D. G. Luenberger, *Linear and Nonlinear Programming*, Addison-Wesley, 1984.
- [32] J. Yan, M. May, and B. Plattner, "Distributed and Optimal Congestion Control for Application-Layer Multicast: A Synchronous Dual Algorithm," in *5th IEEE CCNC*, 2008.
- [33] J. Yan, M. May, and B. Plattner, "Comments on "Optimal Resource Allocation in Overlay Multicast","" *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 4, pp. 572–573, 2008.
- [34] F. Kelly, "Fairness and Stability of End-to-End Congestion Control," *European Journal of Control*, vol. 9, pp. 159–176, 2003.
- [35] D. Bertsekas, *Nonlinear Programming*, Athena Scientific, 1995.
- [36] D. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computation*, Prentice-Hall, 1989.

APPENDIX A

ILLUSTRATION OF THE PROPOSED NETWORK MODEL

We illustrate our distributed network model with a detailed example in Figure 11. For simplicity, no TCP cross traffic is introduced in this example.

The full matrix of link capacity constraints in the example is given in Inequalities (13). Please note that (i) links in the model are directed links; and (ii) the link capacity constrains the flows that pass through it in each direction independently.

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} \leq \begin{pmatrix} 6 \\ 3 \\ 8 \\ 8 \\ 15 \\ 10 \\ 10 \\ 2 \\ 2 \end{pmatrix} \quad (13)$$

In the example, there are five end-to-end unicast flows ($F = 5$). The network is composed of four directed bottleneck links ($L = 4$). The available bandwidth for the channel is shown in Fig. 11(c). Hence, the available bandwidth constraint at the bottleneck link, *i.e.*, the inequality (1), becomes:

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} \leq \begin{pmatrix} 6 \\ 8 \\ 2 \\ 2 \end{pmatrix} \quad (14)$$

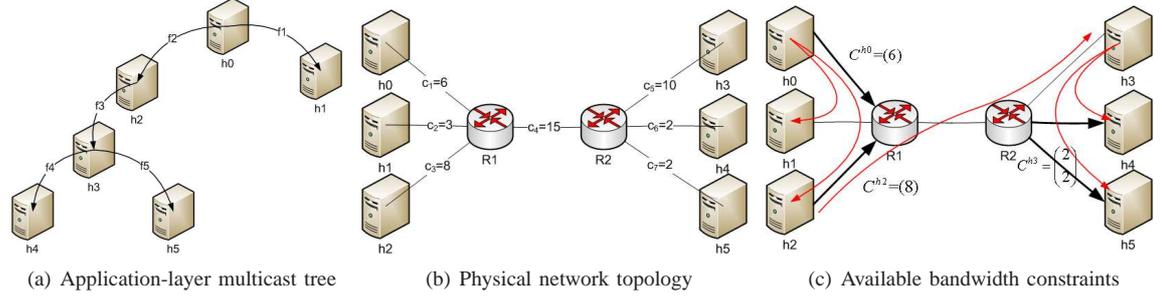


Fig. 11. Example of the proposed network model. The unit used for bandwidth is Mbps. The bold links in (c) are bottleneck links in the arrow direction, no TCP cross traffic introduced in this example. Let the utility function be $U_f(x_f) = \ln(x_f)$. The optimal rates (Mbps) allocated by proposed algorithms are $x_1^* = 2.0, x_2^* = 4.0, x_3^* = 4.0, x_4^* = 2$ and $x_5^* = 2$. Then the total utility is $\sum_{f \in F} U_f(x_f^*) = 4.852$. If we allocate the rates independently as unicast flows using TFRC and apply data constraints, we get a different set of rates: $x_1^* = 3, x_2^* = 3, x_3^* = 3, x_4^* = 2$ and $x_5^* = 2$. Thus, the total utility is $\sum_{f \in F} U_f(x_f^*) = 4.682$, which is worse than the optimal result 4.852.

According to our observation, only x_1 and x_2, x_4 and x_5 may share bottlenecks. As shown in Equation 2, Inequality (14) can be decomposed into:

$$\begin{cases} \begin{pmatrix} 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leq 6 \\ \begin{pmatrix} 1 \end{pmatrix} \begin{pmatrix} x_3 \end{pmatrix} \leq \begin{pmatrix} 1 \end{pmatrix} \begin{pmatrix} 8 \end{pmatrix} \\ \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_4 \\ x_5 \end{pmatrix} \leq \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 2 \\ 2 \end{pmatrix} \end{cases} \Leftrightarrow \begin{cases} x_1 + x_2 \leq 6 \\ x_3 \leq 8 \\ x_4 \leq 2 \\ x_5 \leq 2 \end{cases} \quad (15)$$

$$\begin{aligned} L(x, \mu^\alpha, \mu^\beta) &= \sum_{f \in F} (U_f(x_f) - \sum_{l \in L} \left[\mu_l^\alpha (\sum_{f \in F} A_{lf} x_f - c_l) \right] \\ &\quad - \sum_{f' \in F} \mu_{f'}^\beta (\sum_{f \in F} B_{f'f} x_f)) \\ &= \sum_{f \in F} (U_f(x_f) - \sum_{f \in F} x_f \sum_{l \in L} \mu_l^\alpha A_{lf} \\ &\quad - \sum_{f \in F} x_f \sum_{f' \in F} \mu_{f'}^\beta B_{f'f} + \sum_{l \in L} \mu_l^\alpha c_l) \end{aligned} \quad (18)$$

Now, we find that Inequality (15) from our model is much simpler than the full link capacity constraint (Inequality (13)). The decomposed Inequality (15) turns our proposed algorithms into fully distributed algorithms with the lowest message complexity.

In this example, Inequality (3) becomes:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} \leq 0 \quad (16)$$

APPENDIX B DUAL APPROACH

The dual problem is formalized as follows:

$$\min_{\mu^\alpha, \mu^\beta \geq 0} D(\mu^\alpha, \mu^\beta) = \min_{\mu^\alpha, \mu^\beta \geq 0} \max_x L(x, \mu^\alpha, \mu^\beta) \quad (17)$$

The Lagrangian form of the primal optimization problem is:

$$L(x, \mu^\alpha, \mu^\beta) = \sum_{f \in F} (U_f(x_f) - \mu^\alpha (\mathbf{A} \cdot x - c) - \mu^\beta (\mathbf{B} \cdot x))$$

$\mu^\alpha = (\mu_l^\alpha, l \in L)$ and $\mu^\beta = (\mu_f^\beta, f \in F)$ are vectors of Lagrangian multipliers.

Vectors $\lambda^\alpha = (\lambda_f^\alpha, f \in F)$ and $\lambda^\beta = (\lambda_f^\beta, f \in F)$ are defined as:

$$\lambda_f^\alpha = \sum_{l=l(f)} \mu_l^\alpha = \mu_{l(f)}^\alpha \quad (19)$$

$$\lambda_f^\beta = \mu_f^\beta - \sum_{f \rightarrow f'} \mu_{f'}^\beta \quad (20)$$

Further, we get,

$$L(x, \mu^\alpha, \mu^\beta) = \sum_{f \in F} (U_f(x_f) - (\lambda_f^\alpha + \lambda_f^\beta) x_f) + \sum_{l \in L} \mu_l^\alpha c_l \quad (21)$$

μ^α, μ_l^α can be understood as the link price of bottleneck link l . Consequently, $\lambda_f^\alpha, \lambda_f^\beta$ is the bottleneck link price that f has to pay for its single bottleneck, namely $\mu_{l(f)}^\alpha$. For μ^β, μ_f^β is the relay price that f must pay its parent flow f^p for relaying data to f . If f has no parent flow, then $\mu_f^\beta = 0$. Meanwhile, for $f^p, \mu_{f'}^\beta$ can be understood as its relay benefit from f . For λ^β , we interpret λ_f^β as data price of f , which is the relay price μ_f^β subtracts the relay benefit from all its children flows $\sum_{f \rightarrow f'} \mu_{f'}^\beta$.

Now, we have

$$\begin{aligned} D(\mu^\alpha, \mu^\beta) &= \max_x L(x, \mu^\alpha, \mu^\beta) \\ &= \max_x \sum_{f \in F} (U_f(x_f) - (\lambda_f^\alpha + \lambda_f^\beta) x_f) + \sum_{l \in L} \mu_l^\alpha c_l \end{aligned} \quad (22)$$

Applying the Kuhn-Tucker theorem, we solve the dual problem and obtain the maximizer [7] [15]:

$$x_f(\mu^\alpha, \mu^\beta) = [U_f'^{-1}(\lambda_f^\alpha + \lambda_f^\beta)]_{m_f}^{M_f} \quad (23)$$

By Assumption 1, U_f is concave and the constraints of the problem are linear, there is no duality gap (Proposition 5.2.1 in [35]). The dual optimal prices for Lagrangian multipliers $\mu^{\alpha*}$ and $\mu^{\beta*}$ exist (Proposition 5.1.4 in [35]). Moreover, if $\mu^{\alpha*} \geq 0$ and $\mu^{\beta*} \geq 0$ are dual optimal, then $x_f(\mu^{\alpha*}, \mu^{\beta*})$ is also primal optimal (Proposition 5.1.5 in [35]).

We solve the dual problem using the gradient projection method [15][36], where link prices and relay prices are adjusted in the opposite direction to the gradient $\nabla D(\mu^\alpha, \mu^\beta)$:

$$\mu_l^\alpha(t+1) = \left[\mu_l^\alpha(t) - \gamma \frac{\partial D(\mu^\alpha(t), \mu^\beta(t))}{\partial \mu_l^\alpha} \right]^+, \quad (24)$$

$$\mu_f^\beta(t+1) = \left[\mu_f^\beta(t) - \gamma \frac{\partial D(\mu^\alpha(t), \mu^\beta(t))}{\partial \mu_f^\beta} \right]^+. \quad (25)$$

where $\gamma > 0$ is the step size. Substituting the maximizer into (22), then

$$\begin{aligned} D(\mu^\alpha, \mu^\beta) = & \sum_{f \in F} (U_f(x_f(\mu^\alpha, \mu^\beta)) - (\lambda_f^\alpha + \lambda_f^\beta)x_f(\mu^\alpha, \mu^\beta)) \\ & + \sum_{l \in L} \mu_l^\alpha c_l \end{aligned} \quad (26)$$

Since U_f is strictly concave, $D(\mu^\alpha, \mu^\beta)$ is continuously differentiable [36] with derivatives given by

$$\frac{\partial D(\mu^\alpha(t), \mu^\beta(t))}{\partial \mu_l^\alpha} = c_l - \sum_{f \in F(l)} x_f(\mu^\alpha, \mu^\beta), \quad (27)$$

$$\frac{\partial D(\mu^\alpha(t), \mu^\beta(t))}{\partial \mu_f^\beta} = x_{f^p}(\mu^\alpha, \mu^\beta) - x_f(\mu^\alpha, \mu^\beta). \quad (28)$$

Substituting (27) into (24) and (28) into (25), we get:

$$\mu_{l(f)}^\alpha(t+1) = [\mu_{l(f)}^\alpha(t) + \gamma(\sum_{f \in F(l)} x_f(\mu^\alpha(t), \mu^\beta(t)) - c_{l(f)})]^+ \quad (29)$$

$$\mu_f^\beta(t+1) = [\mu_f^\beta(t) + \gamma(x_{f^p}(\mu^\alpha(t), \mu^\beta(t)) - x_f(\mu^\alpha(t), \mu^\beta(t)))]^+ \quad (30)$$

Equation (29) is consistent with the law of supply and demand: if the demand $\sum_{f \in F(l)} x_f$ for bandwidth at bottleneck link $l(f)$ exceeds its available bandwidth supply $c_{l(f)}$ for the channel, the available bandwidth constraint is violated. Thus, the bottleneck link price $\mu_{l(f)}^\alpha$ is raised. Otherwise, $\mu_{l(f)}^\alpha$ is reduced. In equation (30), if f demands a flow rate higher than its parent flow f^p , the relay price μ_f^β is raised. Otherwise, μ_f^β is reduced.

Let's define

$$Y(f) = \sum_l A_{lf} + \sum_{f'} B_{f'f} \text{ and } \bar{Y} = \max_{f \in F} Y(f),$$

$$U(l) = \sum_{f \in F} A_{lf} \text{ and } \bar{U} = \max_{l \in L} U(l),$$

$$V(f') = \sum_{f \in F} B_{f'f} \text{ and } \bar{V} = \max_{f' \in F} V(f'),$$

$$\bar{Z} = \max\{\bar{U}, \bar{V}\},$$

$$\kappa = \max_{f \in F} \kappa_f.$$

Theorem 2: Assume that $0 < \gamma < 2/\bar{\kappa}\bar{Y}\bar{Z}$, the rate allocation by the dual algorithm in Table II converges to primal-dual optimal for any application layer multicast session. (see Theorem 1 in [15])