Revisiting Optimization Based Rate Allocation for Application Layer Multicast: Distributed Model and Approaches

Jinyao Yan*†, Martin May‡, Bernhard Plattner*

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

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

‡Paris Research Lab, Thomson, Paris, France

***Abstract*— Multicast is an efficient method to deliver data to a large number of receivers. In this paper, we revisit the topic of distributed and optimal rate allocation in application layer multicast. First, we propose a fully distributed network model based on our observation and assumptions, and form an optimization problem to maximize the aggregate utilities of all receivers. Second, we propose a novel primal approach and a typical dual approach as well as the accordingly designed algorithms that solve the optimization problem. Third, we extend the algorithms to an asynchronous setting to match the reality of large networks. Finally, we evaluate the performance of the proposed two algorithms in terms of aggregate utility, time complexity (convergence rate), and messaging complexity. Extensive experiments show that both proposed algorithms generate minimal messaging overhead, and that they are optimal in terms of overall utility for multicast applications.**

I. Introduction

IP multicast is a networking technology, addressing the problem of efficient delivery of data over the Internet to a large number of receivers. Application examples using multicast technologies are real-time IPTV, distance learning and online game, etc. Unfortunately, IP multicast is still not yet widely deployed in the public Internet, mainly due to technical and commercial deployment issues in the inter-domain multicast case. To overcome the deployment problems, a new type of multicast solution was developed: application-layer multicast ([4], [5] and its references)or overlay multicast. In application-layer multicast, end hosts implement the multicast functionality without relying on the support from the underlying IP routers. In addition, application-layer multicast enables the flexibility of adding higher layer functionality, such as rate scaling at intermediate nodes.

To deal with the diverse and changing network conditions for the multicast streaming channel, rate control protocols are used to adapt the sending rate such that the current available network resources are neither overloaded nor underutilized. However, in existing rate control approaches for application layer multicast, intermediate nodes determine the downstream rate without considering the application quality and information from neighbor and parent nodes. For example in End System Multicast [4], the rate for each flow is calculated locally using the unicast TCP-friendly rate control algorithm (TFRC) [1], considering neither the structure of the multicast tree nor the streaming quality. Moreover, for rate allocation algorithms to be efficient, they have to be fully distributed to allow the multicast sessions to scale up to very large sizes.

Therefore, the challenge is to develop ***optimal*** and ***distributed*** rate allocation algorithms for multicast sessions, with the goal to optimize the overall utility of all receiving nodes. In the mean time, these algorithms have to minimize the messaging overhead in the multicast tree.

It is commonly believed that application data could be sent multiple times over the same physical link in application layer multicast [5][24][4]. In particular, bottleneck links may be shared by different level nodes in the tree. Contrary to this common sense, our observation is only sibling flows may traverse the same physical links. In other words, flows were not going through the same physical links with non-sibling flows (such as with their parent flow, children flows or uncle flows etc.). Based on this observation, we propose a ***fully distributed*** network model for application layer multicast sessions using a utility-pricing model [2][13][18], which was proposed as an analytical tool for rate allocation mechanisms. We also formulate and solve an optimization problem to optimize the aggregate utility for the multicast session. Specifically, we describe two approaches that solve the optimization problem, namely an originally proposed primal algorithm (a feasible direction algorithm) and a typical dual algorithm. Note that the algorithms solely rely on the coordination of neighbour nodes and do not require IP routers to compute and communicate link prices. With the help of extensive experiments, we evaluate and compare the performance of the proposed primal and the dual algorithm in terms of aggregate utility, time complexity (convergence rate), and messaging complexity. We summarize the properties of the two algorithms:

1) Both algorithms maximize the overall utility in the multicast tree;

2) The dual algorithm converges faster but introduces oscillations whereas the primal algorithm converges slower while being more stable;
3) Most importantly, both algorithms are fully distributed with the very small messaging overhead and thus, in terms of message complexity, are more applicable for large multicast systems.

The rest of this paper is organized as follows. In section II, we briefly review the related work in rate allocation for IP multicast and application layer multicast. In section III, we introduce in great detail the distributed and optimal rate allocation for multicast sessions including the network model, the primal and the dual algorithm. We extend the algorithms to asynchronous environments which better reflects the reality of large networks in section IV. Section V presents the evaluation and comparison of algorithms and validation of assumptions. Finally, we conclude the paper in section VI.

II. Related Work

Rate control for multicast and application-layer multicast is difficult because it not only requires to scale up to a large number of receivers, it also needs to work in heterogeneous network and host environments. Rate control algorithms for IP multicast applications, such as RLM [7], decide on the downstream sending rate at intermediate routers independently with local information, without consideration information about the applications or information from neighbor and parent routers in the multicast channel. In most congestion control schemes for application layer multicast streaming, each flow in the multicast tree is sent via a unicast algorithms such as TCP or TFRC algorithm without considering the structure of the multicast tree. An optimal rate control algorithm was proposed in [11] for layer coded streaming in application layer multicast, however only access links and a set of possible streaming rates were considered in the model.

The pricing models proposed in [2] and [13] optimize the overall utility of the application and have been applied to IP multicast with router support in [9] and further in [20] in consideration of the co-existing unicast TCP. The pricing model was introduced to application-layer multicast first in Cui algorithm [3] and our previous work [22] to optimize the overall application utility. Cui's work is based on a dual approach, and it is most related to our dual approach in this work. In [3] however, the authors design a rate control algorithm for application-layer multicast that creates significant message overhead, where the flow rates and all physical link prices are explicitly exchanged with some centralized nodes for the optimal rate computation. To minimize the message overhead, we improve the utility-price model based on our observations to a fully distributed model for multicast applications in this paper. Moreover, we propose a novel primal algorithm as well as a typical dual algorithm to solve the optimization problem based on the distributed network model.

III. Distributed and Optimal Rate Control for Application Layer Multicast

A. Network model and Problem Formulation

Consider an overlay network of $n+1$ end hosts, denoted as $H = \{h_0, h_1, \ldots, h_i, \ldots, h_n\}$. End host h_0 is the source of the multicast channel. Other end hosts are consumers of the multicast channel. The structure of the overlay tree is given by the used application-layer multicast protocol. Non-leaf nodes are forwarding data to its children and are able to scale-down the flows with techniques such as fine granular scalable coding/transcoding technique [6], fulfilling the constraint of flow data. The multicast channel consists of n end-to-end unicast flows, denoted as $F = \{f_1, \ldots, f_i, \ldots, f_n\}$. Flow f_i (or f_{h_i})is the flow that terminates at h_i. Each flow $f_i \in F$ has a rate x_i (or x_{f_i}). 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.*, $h^p \rightarrow h \rightarrow h'$. We make two assumptions and the definition of bottleneck link as follows,

***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: $U''_f \geq 1/\kappa > 0$.

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 bandwidth 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 and $l(f)$ is the bottleneck link through which f goes. The location of the bottleneck of f can be inferred by topology tools [8][25]. The available bandwidth of bottleneck links can also be measured in an end-to-end manner by tools such as *pathchar* and *pathrate*. 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 $\Gamma = \{1, 2, \ldots, L\}$. 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 $C = (c_l, l \in \Gamma)$.

Bottleneck links are only active constraints of flow rate allocation. In practice, a bottleneck link is a saturated link where the link/router is dropping packets of flows sharing this link. Now we make the following two assumptions validated and discussed in section V-E.

***Assumption** 3:* Each flow f has a single bottleneck link at a particular point of time denoted as $l(f) \in \Gamma$.

Definition** 2:* ***Sibling flows are flows sent from the same

end host h, namely all $f'_h \in F'_h$ are sibling flows, and non-sibling flows ($f \notin F'_h, f \in F$) otherwise.

***Assumption** 4:* (**Observation**) In an efficiently formed multicast tree, only sibling flows may share the same bottleneck link. *i.e.*, if $l(f_i) = l(f_j)$, then $(h_i)^p = (h_j)^p$.

In Fig. 1, sibling flows f_2 and f_3 share the bottleneck link l_2,but sibling flow f_1 is going through bottleneck link l_1 and does not share l_2. Our observation is that an efficient overlay tree such as delay-optimized or bandwidth-optimized overlay tree [26][27] will avoid non-sibling flows going through the same directed bottleneck link. In this example, flows of f_h, f_4 and f_5 in dash line should not go through bottleneck link l_2, flow f_4 should not go through bottleneck link l_1. In particular, [11] provides an algorithm to eliminate inter-path and intra-path bottlenecks shared by non-sibling flows in an overlay tree. Flows f_h, f_4 and f_5 are intra-path flows of f_2 and f_3, while f_4 is an inter-path flow of f_1.

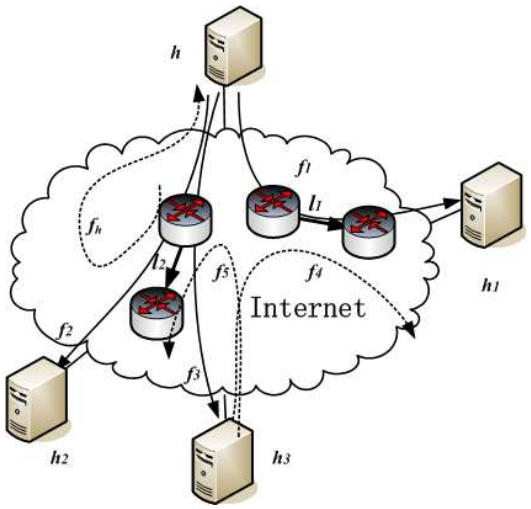


Fig. 1. Only sibling flows may share bottleneck links (The bold links l_1 and l_2 are bottleneck links in the arrow direction. An optimized overlay tree construction mechanism will avoid f_h and f_5 going through bottleneck link l_2 and f_4 going through bottleneck link l_1.)

We define further a $\Gamma \times F$ matrix A. $A_{lf} = 1$, if flow f goes through bottleneck link l in its direction, *i.e.*, $l = l(f), f \in F(l)$. Otherwise, $A_{lf} = 0$. It follows that the rate summation of all flows in the channel that go through the bottleneck link l should not exceed c_l . Such available bandwidth constraints at bottleneck links are expressed as follows:

$$A \cdot x \leq C \tag{1}$$

Now let us look at an end host h with its child flows f'_h. We define a $\Gamma_h \times F'_h$ matrix A, where Γ_h is the set of bottleneck links of all f'_h flows. If flow f'_h goes through the bottleneck link $l_h \in \Gamma_h$ *i.e.*, $l_h = l(f'_h)$, then $A^h_{l_h f'_h} = 1$, otherwise $A^h_{l_h f'_h} = 0$. By Assumption 4, non-sibling flows go through independent bottleneck links and the constraint in (1) can be **decomposed** in:

$$A \cdot x \leq C \Leftrightarrow \qquad A^h \cdot x^{F'_h} \leq C^h, \forall h \in H \tag{2}$$

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

Moreover, the downstream flow rate is constrained by the upstream flow rate, namely, if $f \rightarrow f'$ then $x_{f'} \leq x_f$. We define the data constraint or flow preservation $F \times F$ matrix B. $B_{f_1 f_2} = -1$,if $f_2 \rightarrow f_1$,*i.e.*, $f_1 = f'_2$; $B_{f_1 f_2} = 1$, if $f_1 = f_2$, and f_1 has a parent flow; Otherwise $B_{f_1 f_2} = 0$. Hence, given the application-layer multicast tree, the data constraints can be formalized as follows:

$$B \cdot x \leq 0 \tag{3}$$

A summary of the notations and an illustrative example for the distributed network model can be found in Appendix.

1) Problem Formulation: Our objective is to devise distributed rate control algorithms that maximize the aggregate utility, *i.e.*, the overall utilities of all flows in the application-layer multicast tree:

$$\max_{m_f \leq x_f \leq M_f} \sum_{f \in F} U_f(x_f) \tag{4}$$

fulfilling the following distributed/local constraints:

$$\begin{cases} A^h \cdot x^{F'_h} \leq C^h \\ B \cdot x \leq 0 \end{cases}$$

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

B. Primal Algorithm - A Feasible Direction Algorithm

Primal algorithms work on the original problem directly by searching the feasible region for an optimal solution. Thanks to our fully distributed model, solving the optimization problem (4) directly requires only coordination among those sibling flows sharing bottleneck links.

The novel primal algorithm in this paper is a feasible direction method [15] which finds a direction such that *(i)* a small move in that direction remains feasible, and *(ii)* the aggregate utility is improved. The primal algorithm then moves a finite step in the determined direction, obtaining a new and better rate allocation. The process is repeated until no direction satisfying both *(i)* and *(ii)* can be found. Each allocation generated in the process is feasible and the value of the aggregate utility improves constantly. In general, the convergent rate allocation is the global maximum of the convex optimization problem, however it would be a constrained local maximum of a general (non-convex) problem (Chapter 11.1 in [15]). Therefore, the primal algorithm does not require the convex property of the application utility. Please note that the proposed primal algorithm is different from the primal algorithm in Kelly's work or other penalty algorithms [13][14].

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 when 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 data constrained leaf flow f, its data shadow price is:

$$p_f = \Delta U_f / \Delta x_f = U'_f(x_f) \tag{5}$$

For data constrained intermediate flow f:

$$p_f = U'_f(x_f) + \sum_{f' \in F'} p_{f'} \quad (6)$$

When a flow f is not constrained by its parent flow, its data shadow price is zero ($p_f = 0$). For example, data shadow price of each dash line flow in Fig. 2 is zero.

We call a node data constrained node (Fig. 2(b)) when its incoming flow is a data constrained flow, otherwise it is a data unconstrained node (Fig. 2(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 parent node is 1-bit information and indicating the flow and node are data constrained or not. We assume the network is synchronous such that updates are synchronized to occur at times $t = 1, 2 \ldots$. Each end host h is assumed to be able to ommunicate with neighbors, to be able to measure A^h, C^h and to compute and adapt 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, and $A^h \cdot x_{F'_h}(0) = C^h$. 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 I. The algorithm purely depends on the coordination of neighboring nodes. ***(i)*** Each node receives the data shadow prices from its children nodes for children flows. ***(ii)*** The algorithm reallocates the bandwidth of the bottleneck link with stepsize 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. ***(iii)*** 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.

***Theorem** 1:* For any application layer multicast session, the rate allocation by the primal algorithm in Table I with sufficient small stepsize γ converges to the optimal allocation.

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 schadow 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,

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

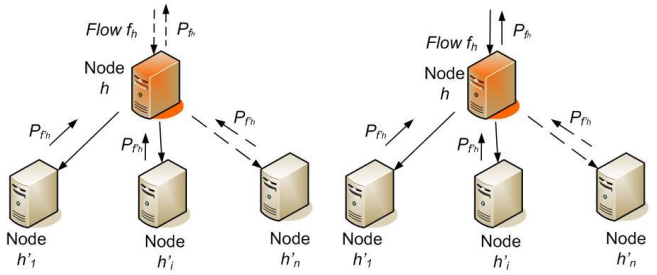
Each allocation generated in the primal algorithm process is feasible and the value of the aggregate utility of the subtree $\sum U(x_{f_i}(t))$ improves constantly. Given the receiving rate f_h, as there is a limit for the aggregate utility of the subtree, the algorithm will finally converge to a maximum point. For a convex optimization problem, the convergent rate allocation is the global maximum (the optimality) of the subtree(Chapter 11.1 in [15]).By each subtree converging to opimality for a given receiving rate iterately, optimality of the entire mutlicast tree rooted at h_0 will be eventually reached. ■

TABLE I

PRIMAL ALGORITHM OF END HOST h_i

Initialization
Sending data with the data constraints applied unicast rate for each flow.
Update the data shadow price from children
Get shadow price for children flows f'_i: $p_{f_j}(t), f_j \in F'_{h_i}$
Compute the median shadow price of children flows: $pj_{med}(t)$
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 $p_{f_j}(t) > pj_{med}(t)$
$x_j(t+1) = x_j(t) + \gamma$
else if $p_{f_j} < pj_{med}$
$x_j(t+1) = x_j(t) - \gamma$
end if
end if
Update Data Shadow Price to parent node
For data constrained node h_i : $p_{f_i}(t+1) = U'(x_i)$
for f_j sharing the same bottleneck $j := 1$ to n do
if $x_j(t) \geq x_i(t)$
$x_i(t+1) = x_j(t); p_{f_i}(t+1) = p_{f_i}(t+1) + p_{f_j}(t)$
else $x_{f_j}(t+1) = x_{f_j}(t)$
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

Unlike the fluctuating convergence procedure in dual approaches [3][22], a feasible direction algorithm converges to the optimality steadily.



(a) data unconstrained node h (b) data constrained node h

Fig. 2. Nodes and flows in the primal algorithm (dashed line means data unconstrained flow, constrained flow otherwise)

C. Dual Algorithm

Besides the novel primal approach, we can also solve the dual problem of (4) with the goal to develop a distributed solution. The dual approach is typically used for the utility-price network model and solving the optimization problem in unicast [2][13] and in multicast [3][9]. However, the network

model and the optimization problem developed in this paper are decomposed, thus they are fully distributed with local constraints. The dual problem is formalized as follows:

$$\min_{\mu^\alpha,\mu^\beta\geq 0} D(\mu^\alpha,\mu^\beta)=\min_{\mu^\alpha,\mu^\beta\geq 0}\max_x L(x,\mu^\alpha,\mu^\beta) \tag{7}$$

The Lagrangian form of the primal optimization problem is:

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

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

$$\begin{aligned}
&L(x,\mu^\alpha,\mu^\beta)\\
&=\sum_{f\in F}(U_f(x_f)-\sum_{l\in L}\left[\mu_l^\alpha(\sum_{f\in F}A_{lf}x_f-c_l)\right]\\
&\quad-\sum_{f'\in F}\mu_{f'}^\beta(\sum_{f\in F}B_{f'f}x_f)\\
&=\sum_{f\in F}(U_f(x_f)-\sum_{f\in F}x_f\sum_{l\in L}\mu_l^\alpha A_{lf}\\
&\quad-\sum_{f\in F}x_f\sum_{f'\in F}\mu_{f'}^\beta B_{f'f}+\sum_{l\in F}\mu_l^\alpha c_l
\end{aligned} \tag{8}$$

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

$$\lambda_f^\alpha=\sum_{l=l(f)}\mu_l^\alpha=\mu_{l(f)}^\alpha \tag{9}$$

$$\lambda_f^\beta=\mu_f^\beta-\sum_{f\to f'}\mu_{f'}^\beta \tag{10}$$

Further, we get,

$$L(x,\mu^\alpha,\mu^\beta)=\sum_{f\in F}(U_f(x_f)-(\lambda_f^\alpha+\lambda_f^\beta)x_f)+\sum_{l\in L}\mu_l^\alpha c_l \tag{11}$$

μ^α, μ_l^α can be understood as the link price of bottleneck link l. Consequently, λ^α, λ_f^α 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, μ_f^β 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 $\sum_{f\to f'}\mu_{f'}^\beta$.

Now, we have

$$\begin{aligned}
&D(\mu^\alpha,\mu^\beta)=\max_x L(x,\mu^\alpha,\mu^\beta)\\
&=\max_x\sum_{f\in F}(U_f(x_f)-(\lambda_f^\alpha+\lambda_f^\beta)x_f)+\sum_{l\in L}\mu_l^\alpha c_l
\end{aligned} \tag{12}$$

Applying the Kuhn-Tucker theorem, we solve the dual problem and obtain the maximizer [2] [3]:

$$x_f(\mu^\alpha,\mu^\beta)=[U_f'^{-1}(\lambda_f^\alpha+\lambda_f^\beta)]_{m_f}^{M_f} \tag{13}$$

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 [16]). The dual optimal prices for Lagrangian multipliers $\mu^{\alpha*}$ and $\mu^{\beta*}$ exist (Proposition 5.1.4 in [16]). 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 [16]).

We solve the dual problem using the gradient projection method [3][17], where link prices and relay prices are adjusted in the opposite direction to the gradient $\nabla D(\mu^\alpha,\mu^\beta)$:

$$\mu_l^\alpha(t+1)=\left[\mu_l^\alpha(t)-\gamma\frac{\partial D(\mu^\alpha(t),\mu^\beta(t))}{\partial\mu_l^\alpha}\right]^+, \tag{14}$$

$$\mu_f^\beta(t+1)=\left[\mu_f^\beta(t)-\gamma\frac{\partial D(\mu^\alpha(t),\mu^\beta(t))}{\partial\mu_f^\beta}\right]^+. \tag{15}$$

where $\gamma>0$ is the step size. Substituting the maximizer into (12), then

$$\begin{aligned}
&D(\mu^\alpha,\mu^\beta)=\\
&\sum_{f\in F}(U_f(x_f(\mu^\alpha,\mu^\beta))-(\lambda_f^\alpha+\lambda_f^\beta)x_f(\mu^\alpha,\mu^\beta))\\
&\quad+\sum_{l\in L}\mu_l^\alpha c_l
\end{aligned} \tag{16}$$

Since U_f is strictly concave, $D(\mu^\alpha,\mu^\beta)$ is continuously differentiable [17] with derivatives given by

$$\frac{\partial D(\mu^\alpha(t),\mu^\beta(t))}{\partial\mu_l^\alpha}=c_l-\sum_{f\in F(l)}x_f(\mu^\alpha,\mu^\beta), \tag{17}$$

$$\frac{\partial D(\mu^\alpha(t),\mu^\beta(t))}{\partial\mu_f^\beta}=x_{f^p}(\mu^\alpha,\mu^\beta)-x_f(\mu^\alpha,\mu^\beta). \tag{18}$$

Substituting (17) into (14) and (18) into (15), we get:

$$\mu_{l(f)}^\alpha(t+1)=[\mu_{l(f)}^\alpha(t)+\gamma(\sum_{f\in F(l)}x_f(\mu^\alpha(t),\mu^\beta(t))-c_{l(f)})]^+ \tag{19}$$

$$\mu_f^\beta(t+1)=[\mu_f^\beta(t)+\gamma(x_f(\mu^\alpha(t),\mu^\beta(t))-x_{f^p}(t))]^+ \tag{20}$$

Equation (19) 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 (20), 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.

TABLE II
DUAL ALGORITHM OF END HOST h_i

Initialization
Sending data with the data constraints applied unicast rate for each flow.
Link Price Update
(by bottleneck link $l = l(f_i') \in \Gamma_i$)at $t = 1, 2 \ldots$:
Update price of the bottleneck link:
$\mu^{\alpha}_{l(f_i')}(t+1) = [\mu^{\alpha}_{l(f_i')}(t) + \gamma(\sum_{f_i' \in F(l)} x_{f_i'}(t) - c_{l(f_i')})]^+$
Relay Price Update (by flow $f_i' \in F'_{h_i}$)at $t = 1, 2 \ldots$:
Update relay price of f_i' with updated flow rate x_{f_i}:
$\mu^{\beta}_{f_i'}(t+1) = [\mu^{\beta}_{f_i'}(t) + \gamma(x_{f_i'}(t) - x_{f_i}(t))]^+$
Flow rate Adaptation (by flow $f_i' \in F'_{h_i}$), At $t = 1, 2 \ldots$
1 Receive relay prices $\mu^{\beta}_{f_j}(t)$
from all children flow $\{f_j \mid f_i' \to f_j\}$
2 Calculate:
$\lambda^{\alpha}_{f_i'}(t) = \mu^{\alpha}_{l(f_i')}(t)$
$\lambda^{\beta}_{f_i'}(t) = \mu^{\beta}_{l(f_i')}(t) - \sum_{f_j \to f_i'} \mu^{\beta}_{f_j}(t)$
3 Adjust rate:
$x_{f_i'}(t+1) = [U'^{-1}_{f_i'}(\lambda^{\alpha}_{f_i'}(t) + \lambda^{\beta}_{f_i'}(t))]^{M_f}_{m_f}$
communicates with the rate $x_{f_i'}(t+1)$ for flow f_i'
4 Send $\mu^{\beta}_{f_i'}(t+1)$ to $(h_i)^p$

The dual algorithm for each end host is presented in Table II. Again, we assume the network is synchronous such that updates are synchronized and occur at times $t = 1, 2 \ldots$. Each end host h is assumed to be capable of communicating with neighbors, to be capable of measuring A^h, C^h, and to compute and adapt the sending rate for each flow f_h' (*i.e.*, sender-based flow). The bottleneck link price is calculated by end host h locally with A^h, C^h and the sending rates of sibling flows sharing the bottleneck. End host h communicates with its parent node to update flow rate x_{f_h} to calculate the relay price, and then, communicates with its children nodes to update relay benefit to calculate the data price of f_h'. 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 avoiding any change of existing infrastructure. We choose the data constraints applied unicast rate as the initial rate in the algorithm and $A^h \cdot x_{F_h'}(0) = C^h$.

Let's define

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

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

$V(f') = \sum_{f \in F} B_{f'f}$ 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 [3])

IV. EXTEND TO ASYNCHRONOUS ALGORITHMS

Both the primal algorithm and the dual algorithm assume a synchronized network environment, namely the flow rate updates and data shadow price updates in primal algorithm, the relay price updates and flow rate updates in dual algorithm are synchronized within the entire overlay session. In this section, we improve the algorithm to an asynchronous setting as we find in realistic network environments, where different peers, flows update their data shadow prices, sending rates, or prices at different times.

A. Asynchronous model

We use the asynchronous model as introduced in [2] [3] for our algorithm. Let $\tilde{T} = \{0, 1, 2, ...\}$ be the set of time instances at which either a flow rate,or 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) $\tilde{T}^{\beta}_f \subseteq \tilde{T}$–the set of time instances at which a flow f (its sender) updates its relay price μ^{β}_f.
3) $\tilde{T}^{d}_f \subseteq \tilde{T}$–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 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 flow f_h', where $(t - T) \le t' \le t$ and $t' \in \tilde{T}^{d}_{f_h'}$, and computes its estimated data shadow price $\hat{p}_{f_h'}(t)$ by using a weighted average of these values:

$$\hat{p}_{f_h'}(t) = \sum_{t'=t-T}^{t} a_{f'}(t', t) p_{f_h'}(t'), \sum_{t'=t-T}^{t} a_{f'}(t', t) = 1. \quad (21)$$

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

$$\hat{x}_{f_h}(t) = \sum_{t'=t-T}^{t} b_f(t', t) x_{f_h}(t'), \sum_{t'=t-T}^{t} b_f(t', t) = 1. \quad (22)$$

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 (21) redistribute the rates among children and estimate the parent flow rate $\hat{x}_{f_h}(t)$ in equation (22) to compute its shadow price. We omit presenting the details of the asynchronous algorithm due to space limitation.

C. 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 computes locally the link price $\mu^{\alpha}_{l(f'_h)}(t)$ with rates $x_{f'_h}(t)$ of sibling flows sharing the bottleneck l, namely $f'_h \in F_h(l)$ and C^h using equation (19).

Relay price update: End host h keeps track of all recent rate updates of flow $x_{f_h}(t')$, $(t-T) \le t' \le t$ 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:

$$\hat{x}_{f_h}(t) = \sum_{t'=t-T}^{t} b_{f_h}(t',t) x_{f_h}(t'), \sum_{t'=t-T}^{t} b_{f_h}(t',t) = 1. \quad (23)$$

Then, end host h using this estimated rate for flow f_h computes the relay price for f'_h, namely $\mu^{\beta}_{f'_h}(t+1)$, as in equation (20).

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

$$\hat{\mu}^{\beta}_{f'_{h'}}(t) = \sum_{t'=t-T}^{t} b^{\beta}_{f'_h f'_{h'}}(t',t) \mu^{\beta}_{f'_{h'}}(t'), \sum_{t'=t-T}^{t} b^{\beta}_{f'_h f'_{h'}}(t',t) = 1. \quad (24)$$

End host h computes first the prices $\mu^{\alpha}_{l(f'_h)}(t)$ of bottleneck links flow f'_h goes through, and with $\mu^{\alpha}_{l(f'_h)}(t)$ end node h calculates $\lambda^{\alpha}_{f'_h}(t)$. With estimated $\hat{\mu}^{\beta}_{f'_{h'}}(t)$ and $\mu^{\beta}_{f'_h}(t)$, end host calculates $\lambda^{\beta}_{f'_h}(t)$, and further computes children flow rate according to equations (13).

Algorithm: To upgrade the synchronous dual algorithm to an asynchronous one, we estimate the flow rate $\hat{x}_{f_h}(t)$ in equation (23) to compute the relay price during step "Relay Price Update", and estimate relay price $\hat{\mu}^{\beta}_{f'_{h'}}(t)$ in equation (24) to compute sending rate $x_{f'_h}(t+1)$. We again omit presenting the details of the asynchronous algorithm.

D. Weighted average policies

The weighted average policy in the model is very general and 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 $(t-T) \le t' \le t$ is used for the estimation.
- update average: all data in time window $(t-T) \le t' \le t$ are averaged for the estimation.

Due to space limitations, we only use the "update average" policy for our performance evaluation.

V. Evaluation and Comparison

A. Experiment Setting

While we have carried out experiments on a very large number of network topologies, we present only some representative results of experiments on network topologies generated with Brite [19] in the router level topology model with 1000 routers. The available bandwidths of all links are randomly distributed between 10Mbps and 1000Mbps with 0.6ms average delay. We set up some other smaller network topologies with 100 routers of the same average link delay and available bandwidth properties (part of the 1000 routers topology). We ran each experiment for 10 times with random nodes consisting of the application layer multicast trees on the top of network topologies. Available bandwidth of the access link of each node is randomly distributed from 1Mbps to 100Mbps. We set the utility function $U_f(x_f) = \ln(x_f)$, which is concave and strictly increasing for x_f.

The average update interval is 10ms while the average location measurement interval of bottleneck links is 1000ms as we consider slowly time-varying asynchronous network that the dynamics of network is slower than the dynamics of flow traffic. Hence, the overhead of the bottleneck measurement is very small. We set the time window T for the weighted average updates to 50ms for both the primal algorithm and the dual algorithm in the experiments.

B. Rate Allocation Results

First, we compare the rate allocation results of our proposed algorithms with Cui algorithm[1] and also with a standard unicast algorithm. We generate various application layer multicast systems sizes from 5 to 1000 nodes. In our experiments, the stepsize γ is set to 0.0005. Fig. 3 shows that both the primal algorithm and the dual algorithm are optimal in terms of average utility and aggregate utility for various number of peers, which is equivalent to the optimal result obtained with Cui algorithm. We first allocate the rates independently as unicast flows using the TCP/TFRC algorithm, and apply the data constraint. Then get a set of rates for data constraints applied unicast and lower average/aggregate utility.

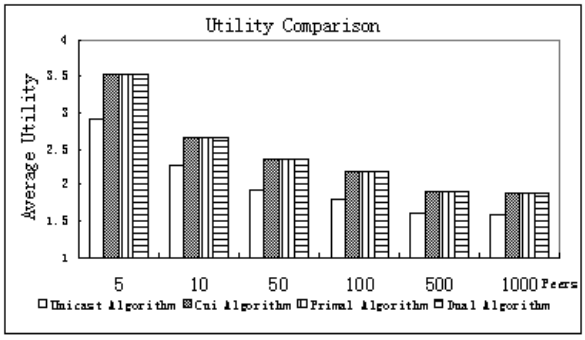


Fig. 3. Comparison of Average Utility

[1]By Cui algorithm in our experiments, we mean the network model and Cui algorithm with its correction in [10].

C. Time complexity

Stability and convergence properties for unicast congestion control algorithms have already been analyzed in literature [2][13][21]. Here, we compare with the help of experiments the two proposed algorithms for application-layer multicast in terms of convergence. To further understand the convergence rate of the two algorithms, we compare them under different stepsizes. Please note that Cui algorithm is a dual algorithm with same convergence property as our proposed dual algorithm. We present some representative results from our experiments in Fig. 4 to illustrate our careful observations, where a flow rate converges to optimal rate when new flows come at time $t = 10s$ and $t = 40s$.

1) The dual algorithm introduces oscillations and unrealistic rate allocation during the convergence process which introduces high link prices (*i.e.*, high packet loss rates), whereas the primal algorithm leads to good rate allocations without oscillation at any time of convergence process, shown in Fig. 4(a,b,c)
2) The primal algorithm converges slower than dual algorithm in our experiment setting, plotted in Fig. 4(a,b). The primal algorithm in Fig. 4(a) also converges to optimal rates, but only at the end (not shown in the figure).
3) Large stepsizes result in faster convergence than small stepsizes (see Fig. 4(a,b)). However, large stepsizes may lead to oscillation and instabilility in particular for the dual algorithm shown in Fig. 4(c).

D. Message complexity

Next, we compare the overall messaging overhead of the two proposed algorithms in Fig. 5 during the experiments time of 600 seconds in various sizes of multicast systems. Again, each messaging overhead is the average of 10 times experiments for overlay trees with random nodes. The results show that our proposed algorithms produce much less messaging overhead compared to that of Cui algorithm. The longer the length of flow path, and the less bottleneck links a flow is going through, the smaller is the messaging overhead of our algorithms compared to Cui algorithm. In our algorithms, there is no link price update message to be exchanged between hosts, as only sibling flows may share bottleneck links triggering link prices and they are controlled and sent by the same host. Indeed, we found that most link price messages in Cui algorithm report link prices with a value of zero. Moreover, bottleneck link constrains are active constrains for flows in the channel, and other links are inactive constraints with both shadow prices in the primal algorithm and link price in the dual algorithm equal to zero. Moreover, the overhead of the bottleneck measurement is very small. Hence, we conclude that our algorithms minimize the messaging overhead of the distributed algorithms, and they are scalable to very large size multicast systems.

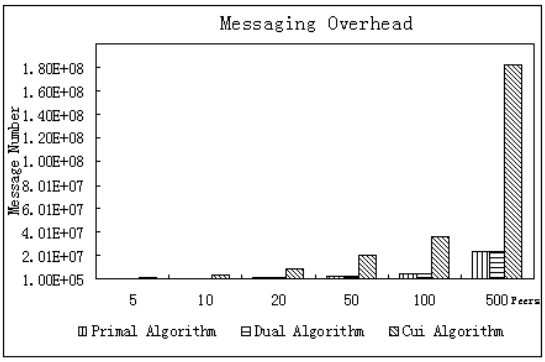


Fig. 5. Comparison of Messaging Overhead in 600s

E. Discussion on the assumptions and validations

Validation of Assumption 3: One bottleneck link per flow path

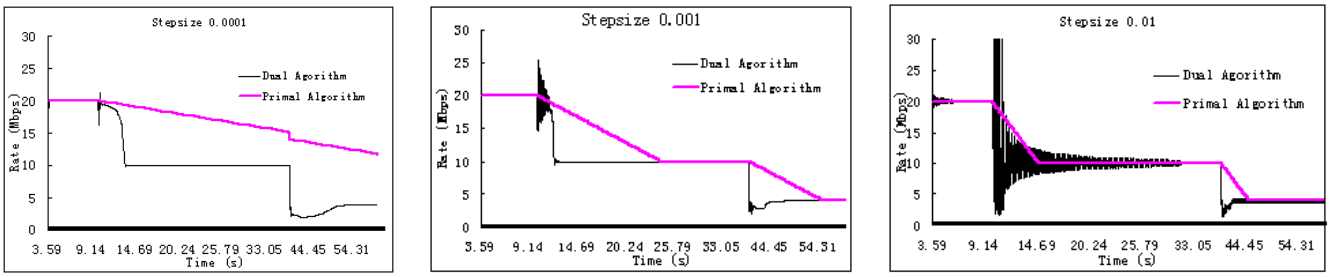
To verify how many bottleneck links per flow exist in the network, we run 10 times with 50 independent unicast TFRC flows to replace a multicast session with 50 flows in the same setting. From the results in Fig. 6, we see that the average bottleneck link number of a unicast flow is much smaller than the length of the whole path; and most of the unicast flows have only one bottleneck link. Indeed, we found the worst case was four unicast flows experienced two bottleneck links for a short time in the experiments. Flows may have more than one bottleneck links at one time instance in the asynchronous experiment setting. For simplicity, we made Assumption 3 in the model. Indeed, the proposed model and the algorithms works well for multi-bottleneck links per flow cases as indeed examined in our experiments. The more bottleneck links were in the path, the more messaging overhead might be produced in both the primal and the dual algorithm.

A flow in the multicast session may have no active bottleneck link, which means no active link constraint but a data constraint is applied to the flow. This is why the average number of bottleneck links is smaller than one for each flow in multicast sessions in Fig. 6.

For most of the cases considered in this paper, we validated the "One bottleneck link per flow path" assumption. We found that the number of bottleneck links on a flow path depends on the dynamic network topology, the cross traffic, and the time window T for the measurement of the bottleneck links. These details however are out of scope for this paper.

Validation of Assumption 4: Only sibling flows share the bottleneck links

Tree construction mechanism: Tree construction mechanisms are typical delay and bandwidth optimized in application layer multicast systems since applications such as multimedia streaming are often very time sensitive and bandwidth demanding. In our experiments, we designed a typical tree construction mechanism optimized for real-time applications, which is delay-based with degree constraints [26][27][28]. A new peer selects the closest peer in the tree as its parent node in terms of end-to-end flow delay. We further constrain that each



(a) (b) (c)

Fig. 4. Convergence of proposed algorithms with various stepsizes (Tree churn happens when nodes join at $t = 10$ and $t = 40$)

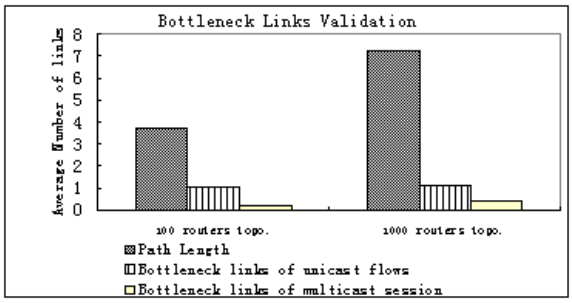


Fig. 6. Bottleneck link validation (The worst case was four unicast flows experienced two bottleneck links instantaneously, the rest 46 unicast flows experienced exactly one bottleneck link in one time experiment. Note that bottleneck link can be any link in the flow path in experiments)

peer has at most four children for high available bandwidth.

By examining all bottleneck link constraint matrices for overlay trees of various sizes in our experiments, we found that the assumption "only sibling flows may share bottlenecks" holds always with the delay-based tree construction mechanism; therewith non-sibling flows are constrained by different bottlenecks. As a result, available bandwidth constraints at bottleneck links (eq.(1)) can be decomposed for each subtree to eq. (2), which makes the network model fully distributed. Inter-path and intra-path bottlenecks can be eliminated in an optimized overlay tree [11]. Therefore, only sibling flows may share bottlenecks when the overlay tree is efficiently formed. Note that our observation is apparently also true for IP multicast.

VI. Concluding Remarks

In this paper,we formalized and solved the problem of optimal rate allocation of a given application layer multicast tree as a convex optimization problem with constraints. Moreover, our observation and assumptions reduce the complexity of the problem with localized constraints for distributed algorithms. We further propose a novel primal approach and a typical dual approach, and design the according algorithms to solve the optimization problem. Thanks to the distributed model, our experiments show both proposed algorithms are optimal in terms of global utility and are fully distributed with a very small messaging overhead. Note that our proposed algorithms can be applied to multi-tree multicast systems as each tree can be applied respectively.

Formal derivation of convergence rate for two proposed algorithms in asynchronous environments is an interesting future issue from the theoretical point of view. We will measure real-time video streaming quality as utility functions. We are also about to integrate the proposed algorithms into large application-layer multicast systems to test the performance in our future work.

VII. Acknowledgments

Thank Dr.Y. Cui for his code of Cui algorithm. Mr.Haocheng Huang from Communication university of China for his discussion and help in algorithms implementation.

References

[1] S. Floyd, M. Handley, and J. Padhye "Equation-Based Congestion Control for Unicast Application", ACM SIGCOMM'00, September 2000.
[2] S. Low, D. E. Lapsley, "Optimization Flow Control, I: Basic Algorithm and Convergence", IEEE/ACM TON, vol. 7, no. 6, Dec. 1999.
[3] Y.Cui, Y.Xue and K.Nahrstedt, "Optimal Resource Allocation in Overlay Multicast", Parallel and Distributed Systems, IEEE Transactions on, 17(8), 808-823,2006
[4] Y. Chu, S. G. Rao, and H. Zhang, "A case for End System Multicast" in Proc. ACM Sigmetrics, June 2000
[5] J. Liu, S. G. Rao, B. Li, and H. Zhang, "Opportunities and Challenges of Peer-to-Peer Internet Video Broadcast", IEEE JSAC, Vol. 96, No. 1, pp. 11-24, January 2008.
[6] W. Li, "Overview of Fine Granularity Scalability in MPEG-4 Video Standard", IEEE Trans.on CSVT, Vol.11, No.3, March 2001, pp301-317V.
[7] S. McCanne, Van Jacobson and M. Vetterli, "Receiver-driven Layered Multicast"New York,USA, ACM SIGCOMM, Aug. 1996.
[8] T. Bu, N G Duffield, F Lo Presti, D. Towsley, "Network tomography on general topologies",ACM SIGMETRICS 2002, Marina Del Rey, USA.
[9] Kar, S. Sarkar and L. Tassiulas,"Optimization basd rate control for multirate multicast sessions",in IEEE INFOCOM, 2001.
[10] J. Yan, M. May, and B. Plattner, "Comments on 'Optimal Resource Allocation in Overlay Multicast"', IEEE Trans. on Paral. and Dist. Sys.,Vol.19, No.1, Jan. 2008.
[11] Min Sik Kim, Yi Li, Simon S. Lam, "Eliminating Bottlenecks in Overlay Multicast", Proceedings of IFIP Networking 2005, Waterloo, Canada, May 2-6, 2005.
[12] B.Akbari, H.R.Rabiee, M. Ghanbari, "An optimal discrete rate allocation for overlay video multicasting"Vol.31 Issue 3, Computer Communications, Feb. 2008.
[13] F. P. Kelly, A.K. Maulloo D.K.H. Tan, "Rate Control for Communication Networks:Shadow Prices, Proportional Fairness and Stability" Vol.49, Journal of the Operational Research Society, 1998.
[14] R.Srikant, The Mathematics of Internet Congestion Control,Birkhauser,2003
[15] David G. Luenberger, "Linear and Nonlinear Programming",Addison-Wesley, 1984.

[16] D.Bertsekas, "Nonlinear Programming", Athena Scientific, 1995.
[17] D.Bertsekas, J.Tsitsiklis, "Parallel and Distributed Computation", Prentice-Hall, 1989.
[18] S.H. Low, F. Paganini,J.C. Doyle, "Internet congestion control"IEEE Control Systems Magazine, Vol.22, Issue 1, Feb. 2002.
[19] www.cs.bu.edu/BRITE/
[20] S. Deb and R. Srikant, "Congestion control for fair resource allocation in networks with multicast flows", IEEE/ACM Transactions on Networking,Vol.12, Issue 2. p.274-285, Apr. 2004.
[21] F. Kelly, "Fairness and stability of end-to-end congestion control", European Journal of Control 9 (2003) 159-176.
[22] J. Yan, M.May, B.Plattner, "Distributed and Optimal Congestion Control for Application-layer Multicast: A Synchronous Dual Algorithm", 5th IEEE CCNC,10-12 Jan. 2008.
[23] J. Yan, K. Katrinis, M. May, and B. Plattner, "Media and TCP-Friendly Congestion Control for Scalable Video Streams", IEEE Transactions on Multimedia,8(2),196-206, 2006
[24] John Jannotti, et al,"Overcast: Reliable Multicasting with an Overlay Network", in Proc. USENIX OSDI, Oct. 2000.
[25] Min Sik Kim, Taekhyun Kim, YongJune Shin, Simon S. Lam, Edward J. Powers, "A wavelet-based approach to detect shared congestion", SIGCOMM 2004,Aug. 2004, Portland, Oregon, USA. S.
[26] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller, "Construction of an efficient overlay multicast infrastructure for realtime applications," in Infocom'03, San Francisco, CA, USA, Apr. 2003.
[27] S. Shi and J.Turner, "Routing in overlay multicast networks," in Infocom'02, New York, NY, USA, June 2002.
[28] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in ACM SIGCOMM 2002, Pittsburgh, PA, USA, Aug. 2002.

APPENDIX

A summary of the notations used in the model can be found in Table III.

Example to illustrate the proposed network model

We illustrate our distributed network model with a detailed example in Fig. 7. For simplicity, no TCP cross traffic is introduced in this example.

The full matrix of link capacity constraints in the example is given in inequality (25). 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.

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} \leq \begin{pmatrix} 6 \\ 3 \\ 8 \\ 8 \\ 15 \\ 10 \\ 10 \\ 2 \\ 2 \end{pmatrix} \tag{25}$$

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. 7(c). Hence, the available bandwidth constraint at the bottleneck link, *i.e.*, the inequality (1), becomes:

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} \leq \begin{pmatrix} 6 \\ 8 \\ 2 \\ 2 \end{pmatrix} \tag{26}$$

By assumption 4, only x_1 and x_2, x_4 and x_5 may share bottlenecks. As shown in Eq. (2), inequality (26) can be decomposed into:

$$\begin{cases} \begin{pmatrix} 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leq 6 \\ \begin{pmatrix} 1 \end{pmatrix} \begin{pmatrix} x_3 \end{pmatrix} \leq \begin{pmatrix} 1 \end{pmatrix} \begin{pmatrix} 8 \end{pmatrix} \\ \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_4 \\ x_5 \end{pmatrix} \leq \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 2 \\ 2 \end{pmatrix} \end{cases}$$

$$\Leftrightarrow \begin{cases} x_1 + x_2 \leq 6 \\ x_3 \leq 8 \\ x_4 \leq 2 \\ x_5 \leq 2 \end{cases} \tag{27}$$

Now, we find that the inequality (27) from our model is much simpler than the full link capacity constraint inequality (25). The decomposed inequality (27) turns our proposed algorithms into fully distributed algorithms with the lowest message complexity.

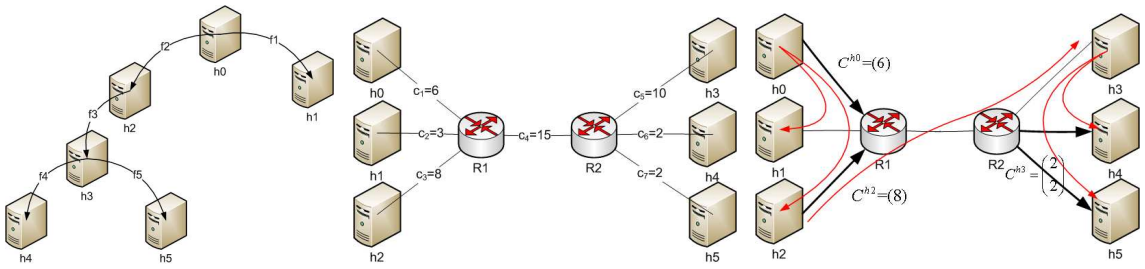
In this example, inequality (3) becomes:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} \leq 0 \tag{28}$$

TABLE III

SUMMARY OF NOTATIONS IN THE MODEL

Notation	Definition
$h \in H = \{h_0, h_1, \ldots, h_n\}$	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 child of h
$f \in F = \{f_1, f_2, \ldots, f_n\}$	Unicast flow in ALM channel
f_i (or f_{h_i}) $\rightarrow h_i$	Flow f_i terminated at h_i
f_{h_i} (or f_i)	Flow terminated at h_i
$x = (x_i or x_{f_i}, f_i \in F)$	Flow rate set of $f_i \in F$
$l \in \Gamma = 1, 2, \ldots, L$	Bottleneck Link l
$c_l \in C$, $l \in \Gamma$	Available bandwidth for the channel of bottleneck link l
$f_{h_i}(or f_i) \rightarrow f'_h \in F'_h$	f'_{h_i} is a child flow of f_{h_i}
F'_h	Set of flow 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
C^h	Vector of 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



(a) Application-layer multicast tree (b) Physical network topology (c) Available bandwidth constraints

Fig. 7. Illustrating 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 $U_f(x_f) = \ln(x_f)$. 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.