Optimal Multicast Algorithm for Content Distribution on Two-Layer Networks

Chunglae Cho and Ye Xia Department of Computer and Information Science and Engineering
University of Florida
Gainesville, Florida 32611
Email: {ccho, yx1}@cise.ufl.edu

Abstract

Traditionally, content distribution service is performed by only the content providers without explicit participation of the Internet Service Providers (ISPs). However, as massive content distribution has become an important application on the Internet, the ISPs have been motivated to actively participate in the content distribution service. We consider the problem of content distribution among data servers over a two-layer network consisting of the overlay and underlay networks, where the control variables can be adjusted at both networks. We propose an optimal backpressure algorithm for content distribution over the layered network. The algorithm is well separated into the overlay and underlay operations such that the communication between the two layers only occurs locally at their interface. We show that our algorithm achieves optimality even when the two network layers operate under different timescales and without time synchronization.

I. Introduction

Massive content distribution has become a more and more important application on the Internet. Some of the representative examples are the delivery systems of high quality television programs or video content, file-sharing systems, and e-science networks. In these systems, the content at a source needs to be distributed to a large number of receivers throughout the network. Traditionally, Internet Service Providers (ISPs) and content providers (CPs, including the content distribution service providers) are independent entities. Content distribution service is performed by only the CPs without explicit participation of the ISPs. The ISPs only provide connectivity to transport content. However, as the service proliferates, the ISPs have the motivation to actively participate the service for the following reasons. First, they can improve the utilization of their network resources by cooperating with the CPs. It has been shown that network-oblivious end-to-end content distribution traffic has a significantly negative impact on the network efficiency [1]. Cooperation between the ISPs and CPs not only leads to more efficient network resource utilization but also improves the content distribution performance [2], [3]. Second, the ISPs can generate revenue by hosting the service and distributing content to their customers, or they can gain additional profits by providing high-quality resources to the CPs [4].

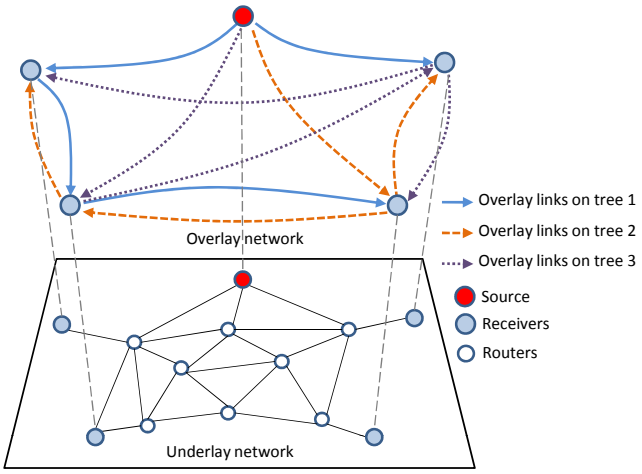


Fig. 1. Two layers of network.

Given the above context, we consider the problem of content distribution among data servers over a two-layer network consisting of the overlay and underlay networks, as shown in Fig. 1. The overlay network is managed by a CP and the underlay physical network is administered by an ISP (or ISPs) [1]. The overlay network consists of data servers and overlay links. The CP manages multiple sessions where each session distributes data from a single data server (source) to a subset of the data servers (receivers) on the overlay network. We assume that the overlay network is highly connected such that multiple distribution trees can be established for each session. The underlay network consists of routers/switches, data servers, and underlay links, which may be either physical links or virtual links.

We study how to distribute content optimally using multiple multicast trees on the overlay network when the control variables (e.g., routing and rate allocations) can be set at both the overlay and underlay networks. Being able to control at both networks amounts to more efficient use of the network resources and improved distribution performance. This is in contrast with the vast majority of the prior work where either there is only one flat network involved or the control can be adjusted only at one layer and the other layer must be held as given. The main challenge is to find a simple optimal algorithm that does not require excessive coordination between the independent operators of the two networks. Hence, our focus is to find an optimal algorithm that can be cleanly partitioned into the overlay part operated by the CP and the underlay part operated by the ISP (or ISPs) with only minimal coordination and without time synchronization. Specifically, we assume that each session has infinite data backlog at its source and is given a small number of trees. The data will be divided and then distributed over the trees to the receivers. The problem we formulate is to find an optimal tree rate allocation subject to the underlay link capacity constraints. The optimization objective is to maximize the sum of the session utilities, each of which is a concave function of the admitted traffic rate.

[1]Although the CP is normally an independent content (or distribution service) provider, an ISP can be both the ISP and the CP.

Our contributions are as follows. First, we develop a simple backpressure algorithm that is ideal for content distribution over a two-layer network. The algorithm is well separated into the overlay and underlay operations such that the communication between the two layers only occurs at the interface between the layers. The algorithm is well distributed and local. Rate control and overlay link scheduling are done locally at each data server and are based on the neighboring queue sizes. Second, we show that our algorithm achieves optimality even when the two network layers operate without time synchronization and/or under different timescales. This is important because in practice the two layers are likely to be operated by independent operators. Third, we analyze the algorithm under a stochastic network setting where the underlay link capacities may vary randomly with some stationary distributions. Using the Lyapunov optimization technique, we prove the optimality of the algorithm in the long-time average sense and the boundedness of the virtual queue sizes. We expect that the real queues are also bounded, although the claim has not been proved yet. We use simulation experiments to support the conjecture. Overall, our algorithm is able to achieve higher application performance and better network resource efficiency than most of the earlier algorithms that do not offer control at both network layers; it is simpler and more practical than the rest of the algorithms that may achieve comparable performance.

A. Related Work

We next contrast our work with the closely related prior studies [5]–[10], which have introduced similar problems and backpressure-based solutions. In [5], the authors present an optimal resource allocation algorithm for overlay multicast service. Their approach is different from ours in the following aspects. First, their problem is a multirate multicast problem with a single session using a single tree, whereas ours is a unirate multicast problem with multiple sessions using multiple trees. Second, they assume that all the operations are run by the CP and there is no participation from an underlay network provider. However, we assume that both parties actively participate in the service, and hence, we wish to divide the operations accordingly and make the interactions between the parties as simple as possible. To achieve the goal, we formulate a very different problem and, as a result, derive a very different algorithm. Lastly, the proof techniques are different. We use the Lyapunov optimization technique to prove the optimality of our algorithm in a stochastic setting, whereas they use the conventional technique for static convex optimization, which cannot be applied to stochastic problems.

None of the studies in [6]–[10] consider the overlay networks, which makes them mostly inapplicable to the two-layer networks. In [6], the authors introduce rate allocation algorithms for multicast sessions in a wireless network. In [7], [9], [10], the authors introduce rate control algorithms for multirate multicast problems with the objective of maximizing the total utility of all the receivers. In [10], the performance objective is the maxmin fairness, and it is not clear how to extend the results to other types of fairness. In [7], [9], the authors assume that the utility functions are strictly concave, which is a key condition used in their proofs. Therefore, their proof techniques cannot be applied to our case with non-strictly concave utility functions.

In [2], the authors consider an architecture called P4P, which allows cooperative traffic control between the content-distribution applications and the network providers. The difference from our approach is that they start

from the ISP's objective, such as minimizing the network congestion. Then, the applications are required to solve the dual problem derived from the ISP's problem. On the other hand, our utility-maximization formulation is more general and it also more directly addresses how to improve the performance of the content distribution service with limited network resources.

There have been game-theoretic studies about the interaction between the overlay content distribution and traffic engineering (TE) at the underlay networks [3], [11], [12]. The perspective is that the ISP and the CP have different objectives and may even compete with each other. Another major difference is that, in those studies, the timescales of the TE and the CP operations are similar, whereas, in our case, the timescale of the TE operations (i.e., adjusting the underlay routes) can be much larger than that of the overlay operations.

Peer-to-peer systems are widely used for content distribution over overlay networks. There have been several proposals [13]–[16] to enhance the end user performance and network resource utilization by traffic localization. The main focus is on selecting peers in close proximity either by using network probing facilities [13] or by interacting with underlay operators [14]–[16]. Since these proposals rely on heuristic methodologies to select proper neighbors, the performance bound and fairness among sessions are not explicitly guaranteed.

The remaining of the paper is organized as follows. Section II describes the network model and the problem formulation. Section III presents the main algorithm and the proof of its optimality. In Section IV, we further evaluate the performance of the algorithm and show the real queues are bounded by simulations. The concluding remarks are in Section V.

II. Model and Problem Formulation

In this section, we introduce a problem formulation with the property that the underlay bandwidth allocation is much separated from the rate allocation on the overlay network. It serves as the basis for reducing the interaction between the overlay and the underlay to local communication at the data servers. Throughout, we consider a discrete-time system.

A. Network Model

We consider a two-layer network consisting of the overlay network and the underlay network. The overlay network is represented by a directed graph $\hat{G} = (\hat{V}, \hat{E})$, where $\hat{V}$ is the set of data servers and $\hat{E}$ is the set of overlay links. The underlay network is represented by a directed graph $G = (V, E)$ where V is the set of nodes, which are data servers and routers/switches, and E is the set of underlay links.

For each underlay link $e \in E$, let $c_e(k)$ denote the capacity of link e at time k, which is generally time-varying due to the background traffic. We assume that, for each $e \in E$, $\{c_e(k)\}_k$ is an i.i.d. random process and with a mean $\bar{c}_e$. In addition, these processes are mutually independent. We also assume that $0 \leq c_e(k) \leq c_e^{max}$, where each c_e^{max} is a positive constant. Let $C_{max} = \max_{e \in E} c_e^{max}$.

An overlay link is an end-to-end unicast connection in the underlay network between two data servers. We consider static, single-path routing for the underlay network in this paper. However, this restriction can be easily

relaxed such that each overlay link can be associated with more than one underlay paths or it may not be associated with any possible paths between the corresponding node pair if hop-by-hop dynamic routing is adopted.

Let S be the set of all multicast sessions in the network. Each session is associated with a source and a set of receivers, which is a subset of $\hat{V}$. We assume that the source of each session has an infinite backlog of data that needs to be transmitted to all the receivers in the session. Each session s is given a set of trees T_s that it uses for data transmission. Without loss of generality, we assume T_s's are disjoint across the sessions. Let T be the union of T_s. Let $\hat{E}_t$ be the set of overlay links on tree t. Let $b(\hat{e})$ and $d(\hat{e})$ denote the transmitting (tail) node and the receiving (head) node of overlay link $\hat{e}$, respectively. Let $\hat{V}_t$ be the set of transmitting nodes $b(\hat{e})$ for all $\hat{e}$ in $\hat{E}_t$; it is the set of nodes on tree t which are not leaves. Let $o(t)$ be the root node of tree t. Let $p(t, \hat{e})$ be the parent link of overlay link $\hat{e}$ on tree t. Let $\Omega(t, \hat{n})$ be the set of child links at node $\hat{n}$ on tree t. Let $\Omega(\hat{n})$ be the set of all outgoing overlay links from node $\hat{n}$.

On every time slot, the source of each session decides how many packets it admits to the network. We denote by $x_s(k)$ the amount of admitted data of session s at time k. We assume that there exists a positive constant X_{max} such that $0 \leq x_s(k) \leq X_{max}$ for all s. The admitted traffic $x_s(k)$ should be transmitted over the trees in T_s for session s. Let $y_t(k)$ be the admitted data that is transmitted over tree t, $t \in T_s$, at time k. We have $x_s(k) = \sum_{t \in T_s} y_t(k)$.

B. Utility Functions

Let $\bar{x}_s(k)$ be the time average of the admitted traffic rate for session s up to time k, i.e.,

$$\bar{x}_s(k) \triangleq \frac{1}{k} \sum_{\kappa=0}^{k-1} x_s(\kappa).$$

Let $\bar{x}_s$ be the limit of $\bar{x}_s(k)$, [2]

$$\bar{x}_s \triangleq \lim_{k \to \infty} \bar{x}_s(k).$$

Each session s is associated with a utility function $U_s(\bar{x}_s)$. We assume that U_s is concave, monotonically increasing, and non-negative over $[0, X_{max}]$. We also assume that it is continuously differentiable, and hence, the derivative of U_s is bounded on $[0, X_{max}]$. Different utility functions can reflect various fairness criteria. For example, setting $U_s(x_s) = \log(x_s + 1)$ for all s leads to the proportional fairness among sessions. On the other hand, if the objective is to maximize the weighted throughput, one can set $U_s(x_s) = \omega_s x_s$ where $\omega_s > 0$ is the weight for session s. More examples can be found in [17], [18].

C. Overlay Transmission Decisions and Link Capacities

For each fixed tree t, overlay data forwarding is decided hop-by-hop by the data servers on the tree. A hop here means an overlay link on the tree. Specifically, on every time slot k, each data server $\hat{n}$ determines the amount of data it intends to transmit, denoted by $r^t_{\hat{e}}(k)$, for every outgoing overlay link $\hat{e} \in \Omega(\hat{n})$ and every tree t such that

[2]We temporarily assume that the limit exists. We shall replace $\lim$ with $\liminf$ or $\limsup$ when the limit does not exist.

$\hat{e} \in \hat{E}_t$. We assume that there is a positive constant R_{max} such that $0 \leq r_{\hat{e}}^t(k) \leq R_{max}$ for all $\hat{e}$ and t. We will call $r_{\hat{e}}^t(k)$ the *overlay transmission rate*. But, note that it is actually the amount of newly scheduled data for tree t at time k to be transmitted over the unicast connection associated with $\hat{e}$. One can imagine that, at time k, $r_{\hat{e}}^t(k)$ amount of data enters a shared queue associated with $\hat{e}$, waiting to be transmitted across $\hat{e}$; this applies to all trees containing $\hat{e}$ as a branch.

The amount of actual transmission on an overlay link at time k depends on the capacity of the overlay link, which is determined by the underlay bandwidth allocation. Let $\nu_{\hat{e}}(k)$ be the capacity of the overlay link $\hat{e}$ at time k. Let $\bar{r}_{\hat{e}}^t$ and $\bar{\nu}_{\hat{e}}$ be the long-time averages for $r_{\hat{e}}^t(k)$ and $\nu_{\hat{e}}(k)$, respectively, similarly defined as for $\bar{x}_s$. For the queue to be stable, it is necessary that the long-time average arrival rate is no more than the long-time average service rate. Hence, we require $\sum_{t:\hat{e}\in\hat{E}_t} \bar{r}_{\hat{e}}^t \leq \bar{\nu}_{\hat{e}}$.

Let Λ_U be the set of all feasible overlay link capacity vectors that can be supported by the underlay network. Λ_U depends on the underlay routing policy. For single-path routing, Λ_U can be written as follows.

$$\Lambda_U \triangleq \{\nu \geq 0 | \sum_{\hat{e}:e\in E_{\hat{e}}} \nu_{\hat{e}} \leq \bar{c}_e, \forall e \in E\}, \tag{1}$$

where $E_{\hat{e}}$ is the set of underlay links corresponding to the overlay link $\hat{e}$ (which is an underlay path). Note that the traffic on an overlay link can be considered as a unicast flow on the underlay network. The constraints in the definition of Λ_U imply that the aggregate amount of unicast flows on an underlay link cannot exceed its capacity in the long-time average sense.

D. Other Notations

Let $\mathcal{I}_{\{c\}}$ be the indicator function such that $\mathcal{I}_{\{c\}} = 1$ if condition c is satified, and $\mathcal{I}_{\{c\}} = 0$ otherwise. Let $[\cdot]^+$ denote the projection onto the non-negative domain. Denote (a_i) to be the vector with entries a_i. Let x, y and r be the vectors $(x_s)_{s\in S}$, $(y_t)_{t\in T}$, and $(r_{\hat{e}}^t)_{t\in T,\hat{e}\in\hat{E}_t}$, respectively. Given a vector $x \in R^n$, let $\|x\|_1$ be the first norm of x, i.e., $\|x\|_1 \triangleq \sum_1^n |x_i|$ where x_i is the i-th component of x. Given a set A, let $|A|$ be the number of elements of the set. Let $\sum_{s,t,\hat{e}}(\cdot)$ be the abbreviation of $\sum_{s\in S}\sum_{t\in T_s}\sum_{\hat{e}\in\hat{E}_t}(\cdot)$.

E. Problem Description

In this subsection, we present the problem formulation, introduce its dual problem, and derive a subgradient algorithm. The subgradient algorithm is used as a reference algorithm to derive our main algorithm in Section III.

Our problem formulation is as follows.

$$\mathbf{P}: \max \sum_{s\in S} U_s(\bar{x}_s)$$

$$\text{s.t. } \bar{r}^t_{\hat{e}} - \bar{r}^t_{p(t,\hat{e})} - \bar{y}_t \mathcal{I}_{\{\hat{e}\in\Omega(t,o(t))\}} \geq 0,$$

$$\forall t \in T, \forall \hat{e} \in \hat{E}_t, \tag{2}$$

$$\sum_{t:\hat{e}\in\hat{E}_t} \bar{r}^t_{\hat{e}} \leq \bar{\nu}_{\hat{e}}, \forall \hat{e} \in \hat{E}, \tag{3}$$

$$\sum_{t\in T_s} \bar{y}_t = \bar{x}_s, \quad \forall s \in S, \tag{4}$$

$$\bar{\nu} \in \Lambda_U, \tag{5}$$

$$0 \leq \bar{x}_s \leq X_{max}, \quad \forall s \in S, \tag{6}$$

$$\bar{y}_t \geq 0, \quad \forall t \in T, \tag{7}$$

$$0 \leq \bar{r}^t_{\hat{e}} \leq R_{max}, \quad \forall t \in T, \forall \hat{e} \in \hat{E}_t. \tag{8}$$

All the variables are the long-time averages. The objective is to maximize the aggregate utility of all multicast sessions. The constraints in (2) are a relaxed form of the tree flow conservation constraints. They imply that for every tree t, the overlay transmission rate on an overlay link $\hat{e}$ on the tree should be no less than the sum of the rate of its parent link $p(t,\hat{e})$ and the exogenous arrival to the tree. We assume that $\bar{r}^t_{p(t,\hat{e})} \equiv 0$ if $p(t,\hat{e})$ is null. The constraints in (3) say that the sum of the overlay transmission rates on an overlay link should not exceed the overlay link capacity assigned by the underlay bandwidth allocation algorithm.

Let $\gamma^t_{\hat{e}}$ and $\lambda_{\hat{e}}$ be the non-negative Lagrange multipliers associated with the constraints (2) and (3), respectively. Then, by relaxing the constraints (2) and (3), we have the following Lagrangian function.

$$\begin{aligned}
&L(x,y,r,\nu;\gamma,\lambda)\\
=&\sum_{s\in S} U_s(x_s)\\
&+\sum_{t\in T}\sum_{\hat{e}\in\hat{E}_t} \gamma^t_{\hat{e}}\big(r^t_{\hat{e}} - r^t_{p(t,\hat{e})} - y_t\mathcal{I}_{\{\hat{e}\in\Omega(t,o(t))\}}\big)\\
&+\sum_{\hat{e}\in\hat{E}} \lambda_{\hat{e}}\big(\nu_{\hat{e}} - \sum_{t:\hat{e}\in\hat{E}_t} r^t_{\hat{e}}\big)\\
=&\sum_{s\in S}\big(U_s(x_s) - \sum_{t\in T_s} y_t \sum_{\hat{e}\in\Omega(t,o(t))} \gamma^t_{\hat{e}}\big)\\
&+\sum_{\hat{e}\in\hat{E}}\sum_{t:\hat{e}\in\hat{E}_t} r^t_{\hat{e}}\big(\gamma^t_{\hat{e}} - \sum_{\hat{e}'\in\Omega(t,d(\hat{e}))} \gamma^t_{\hat{e}'} - \lambda_{\hat{e}}\big) + \sum_{\hat{e}\in\hat{E}} \lambda_{\hat{e}}\nu_{\hat{e}}.
\end{aligned}$$

Let the dual function D be

$$D(\gamma,\lambda) \triangleq \max_{(x,y,r,\nu) \text{ satisfies (4)-(8)}} L(x,y,r,\nu;\gamma,\lambda).$$

Then, the dual problem is:

$$\min\ D(\gamma,\lambda)$$
$$\text{s.t.}\ \ \gamma\geq 0,\ \ \lambda\geq 0.$$

Then, we have a subgradient algorithm as follows.

- On each time slot k, each source s solves the following subproblem.

$$(x_s(k),(y_t(k))_{t\in T_s})\in \arg\max_{\substack{\sum_{t\in T_s} y_t=x_s,\\ 0\leq x_s\leq X_{max},\\ y_t\geq 0}} \Big\{U_s(x_s) - \sum_{t\in T_s} y_t \sum_{\hat{e}\in\Omega(t,o(t))}\gamma^t_{\hat{e}}(k)\Big\}. \tag{9}$$

- On each time slot k, each overlay link $\hat{e}$ solves the following subproblem.

$$((r^t_{\hat{e}}(k))_{t:\hat{e}\in\hat{E}_t})\in \arg\max_{0\leq r^t_{\hat{e}}\leq R_{max}} \Big\{\sum_{t:\hat{e}\in\hat{E}_t} r^t_{\hat{e}}\big(\gamma^t_{\hat{e}}(k) - \sum_{\hat{e}'\in\Omega(t,d(\hat{e}))}\gamma^t_{\hat{e}'}(k)-\lambda_{\hat{e}}(k)\big)\Big\}. \tag{10}$$

- On each time slot k, the underlay bandwidth allocation algorithm solves the following problem, which is to find a rate vector $\nu(k)$ maximizing the weighted throughput.

$$\max\ \sum_{\hat{e}\in\hat{E}}\lambda_{\hat{e}}(k)\nu_{\hat{e}} \tag{11}$$
$$\text{s.t.}\ \ \nu\in\Lambda_U.$$

- On each time slot k, each overlay link $\hat{e}$ updates $\gamma^t_{\hat{e}}$ and $\lambda_{\hat{e}}$ as follows.

$$\gamma^t_{\hat{e}}(k+1)=\Big[\gamma^t_{\hat{e}}(k)-\delta_\gamma\big(r^t_{\hat{e}}(k)-r^t_{p(t,\hat{e})}(k) - y_t(k)\mathcal{I}_{\{\hat{e}\in\Omega(t,o(t))\}}\big)\Big]^+, \tag{12}$$

$$\lambda_{\hat{e}}(k+1)=\Big[\lambda_{\hat{e}}(k)-\delta_\lambda\big(\nu_{\hat{e}}(k)-\sum_{t:\hat{e}\in\hat{E}_t} r^t_{\hat{e}}(k)\big)\Big]^+. \tag{13}$$

Some features of the algorithm are as follows.

- Pros: The operations of the algorithm are well divided between the overlay and underlay networks. The overlay network runs the operations (9), (10), (12), and (13) on the data servers. The underlay network solves the underlay bandwidth allocation problem (11) on the data servers and the routers. The communication between the two networks occurs only at the data servers, which are at the boundaries between the two networks. The underlay network at a data server needs the weights $\lambda_{\hat{e}}(k)$ of the outgoing overlay links from the server; the overlay network at a server needs the overlay link capacities $\nu_{\hat{e}}(k)$ of the neighboring links. This feature is desirable because the operations for the two networks can be run separately by the CP and the ISP, with a small communication cost.

- Cons: The global underlay bandwidth allocation problem (11) needs to be solved completely on every time slot, which may be unrealistic for a large network or a small time-slot size. Moreover, the algorithm requires the mean link capacities $\bar{c}_e$.

III. Asynchronous Algorithm

In this section, we introduce our main algorithm, which allows the two networks to operate asynchronously and possibly on different timescales. It is also more distributed and local. The algorithm is distinguished from the subgradient algorithm in Section II by the following features. First, rather than requiring the underlay network to solve the global underlay bandwidth allocation problem completely on every time slot, the underlay network now only performs very simple computation to update the overlay link rates based on the local overlay and underlay link prices. As a consequence, the overlay link rates $\nu_{\hat{e}}(k)$ are no longer considered as the guaranteed overlay link capacities supported by the underlay network at time k. These rates may sometimes violate the underlay link capacity constraints and queues will form. Second, rather than requiring the overlay and underlay operations to be performed synchronously on every time slot, the two sets of operations can be done asynchronously at different update frequencies. As a result, each network may use rather outdated information produced by the other network. This is more useful in practice since the two networks may be operated by different operators, i.e., the CP and the ISP, and synchronized operations would be difficult to achieve. Third, the new algorithm does not need to have the knowledge of the mean link capacities $\bar{c}_e$.

A. Time Structure for Asynchronous Operations

We assume that the overlay and underlay operations are performed asynchronously. For the overlay operations, we consider a set of consecutive overlay time frames, $\{F_0^O, F_1^O, F_2^O, \ldots\}$, where the overlay rate assignments occur at the beginning of every frame. A frame F_i^O consists of a set of consecutive time slots from $b(F_i^O)$ to $b(F_{i+1}^O)-1$, where $b(F_i^O)$ is the first time slot of frame F_i^O. Similarly, we consider a set of consecutive underlay frames, $\{F_0^U, F_1^U, F_2^U, \ldots\}$, for the underlay operations. The underlay rate assignments occur at the beginning of every underlay frame. A frame F_i^U consists of a set of consecutive time slots from $b(F_i^U)$ to $b(F_{i+1}^U)-1$. We assume that the length of a frame is bounded such that every frame consists of at most F_{max} time slots. Let $F^O(k)$ and $F^U(k)$ be the overlay and underlay frames to which the time slot k belongs, respectively. We denote by $k \in F_i^O$ or $k \in F_i^U$ if time slot k belongs to the frame F_i^O or F_i^U, i.e., $b(F_i^O) \leq k < b(F_{i+1}^O)$ or $b(F_i^U) \leq k < b(F_{i+1}^U)$, respectively. We also denote by $b^O(k)$ and $b^U(k)$ the beginning time slots of the overlay frame $F^O(k)$ and the underlay frame $F^U(k)$, respectively. Let $K^O = \{b(F_0^O), b(F_1^O), b(F_2^O), \ldots\}$ be the set of time slots when an overlay frame begins. Similarly, let $K^U = \{b(F_0^U), b(F_1^U), b(F_2^U), \ldots\}$ be the set of time slots when an underlay frame begins.

As illustrated in Fig. 2, sometimes the frame size of the overlay is smaller than that of the underlay, and sometimes vice versa. This general condition includes two special cases: the case where the frame size of the overlay is larger than that of the underlay and the case where the former is smaller than the latter. In these two special cases, we say the two networks operate on different timescales.

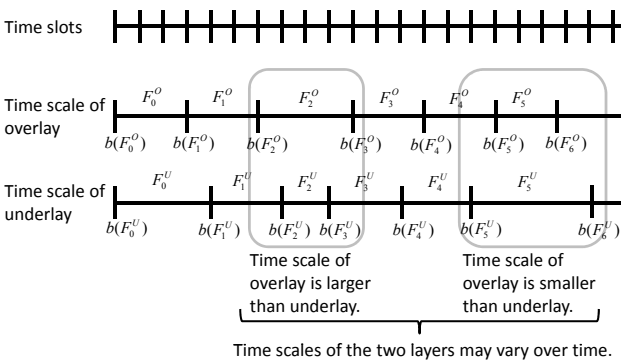


Fig. 2. Time structure for asynchronous overlay and underlay operations.

B. Algorithm

Let $q_{\hat{e}}^t$ and $p_{\hat{e}}$ be the virtual queue sizes corresponding to the constraints (2) and (3), respectively. Let q and p be the corresponding vectors. Let w_e be the virtual queue size corresponding to the constraint $\sum_{\hat{e}:e\in E_{\hat{e}}} \nu_{\hat{e}} \le \bar{c}_e$ in the definition of Λ_U in (1), and let w be the corresponding vector. Let $\delta > 0$ be a parameter, which will be used to adjust the performance tradeoff of the algorithm. Our main algorithm to solve the problem $\mathbf{P}$ is as follows.

Asynchronous Algorithm:

Overlay part:

- On each time slot $k \in K^O$, each source s solves the following subproblem.

$$(x_s(k), (y_t(k))_{t\in T_s}) \in \arg \max_{\substack{\sum_{t\in T_s} y_t = x_s,\\ 0\le x_s \le X_{max},\\ y_t \ge 0}} \Big\{ \frac{1}{\delta} U_s(x_s) - \sum_{t\in T_s} y_t \sum_{\hat{e}\in\Omega(t,o(t))} q_{\hat{e}}^t(k) \Big\}, \text{ for } k \in K^O. \tag{14}$$

- On each time slot $k \in K^O$, each overlay link $\hat{e}$ solves the following subproblem.

$$((r_{\hat{e}}^t(k))_{t:\hat{e}\in\hat{E}_t}) \in \arg \max_{0\le r_{\hat{e}}^t \le R_{max}} \Big\{ \sum_{t:\hat{e}\in\hat{E}_t} r_{\hat{e}}^t \big(q_{\hat{e}}^t(k) - \sum_{\hat{e}'\in\Omega(t,d(\hat{e}))} q_{\hat{e}'}^t(k) - p_{\hat{e}}(k)\big) \Big\}, \text{ for } k \in K^O. \tag{15}$$

- On each time slot $k \notin K^O$, the rate allocation by the overlay part is unchanged, i.e.,

$$x(k) = x(k-1),\ y(k) = y(k-1),\ r(k) = r(k-1), \text{ for } k \notin K^O, \tag{16}$$

Hence, during an overlay frame, the rate allocation by the overlay is the same; that is, for all k in the frame, we have $x(k) = x(b^O(k)), y(k) = y(b^O(k)), r(k) = r(b^O(k))$.

- On each time slot k, each overlay link $\hat{e}$ updates $q^t_{\hat{e}}$ as follows.

$$q^t_{\hat{e}}(k+1) = \Big[q^t_{\hat{e}}(k) - \big(r^t_{\hat{e}}(k) - r^t_{p(t,\hat{e})}(k) - y_t(k)\mathcal{I}_{\{\hat{e}\in\Omega(t,o(t))\}}\big)\Big]^+. \tag{17}$$

  Note that the update of the queue size $q^t_{\hat{e}}$ can be performed locally during an overlay frame once the information of rate allocation $r^t_{p(t,\hat{e})}(b^O(k))$ is transmitted from the parent link at the start of the frame.

Interface between the two parts:

- On every time slot k, each overlay link $\hat{e}$ updates $p_{\hat{e}}$ as follows.

$$p_{\hat{e}}(k+1) = \Big[p_{\hat{e}}(k) - \big(\nu_{\hat{e}}(k) - \sum_{t:\hat{e}\in\hat{E}_t} r^t_{\hat{e}}(k)\big)\Big]^+. \tag{18}$$

  The queue size $p_{\hat{e}}(k)$ is used locally by both the overlay and underlay parts. Since the update of $p_{\hat{e}}(k)$ requires only the local information, there is little burden to update it on every time slot.

Underlay part:

- On each time slot $k \in K^U$, for each overlay link $\hat{e}$, the underlay part updates the overlay link rate $\nu_{\hat{e}}(k)$ as follows.

$$\nu_{\hat{e}}(k) \in \arg\max_{0\le\nu_{\hat{e}}\le\nu_{max}} (p_{\hat{e}}(k) - \sum_{e\in E_{\hat{e}}} w_e(k))\nu_{\hat{e}}, \quad \text{for } k \in K^U. \tag{19}$$

  Note that $\nu_{\hat{e}}(k)$ is bounded by ν_{max} where ν_{max} is a constant.

- On each time slot $k \notin K^U$, the overlay link rates are unchanged, i.e.,

$$\nu(k) = \nu(k-1), \text{ for } k \notin K^U. \tag{20}$$

  Hence, during an underlay frame, the overlay link rates remain constant.

- On each time slot k, for each underlay link e, the underlay part updates the link weight $w_e(k)$ as follows.

$$w_e(k+1) = \Big[w_e(k) - \big(c_e(k) - \sum_{\hat{e}:e\in E_{\hat{e}}} \nu_{\hat{e}}(k)\big)\Big]^+. \tag{21}$$

  Note also that the update of the link weights can be performed locally during an underlay frame once the information about the overlay link rates $\nu(b^U(k))$ is transmitted at the start of the frame.

Remark. In the parts of the algorithm that solve the subproblems (14), (15) and (19), a tie is broken randomly. We assume that the virtual queues are updated on every time slot for ease of presentation. In fact, they only need to be updated once in every frame since the rates are not changed in a frame and the queue information is used in the next frame.

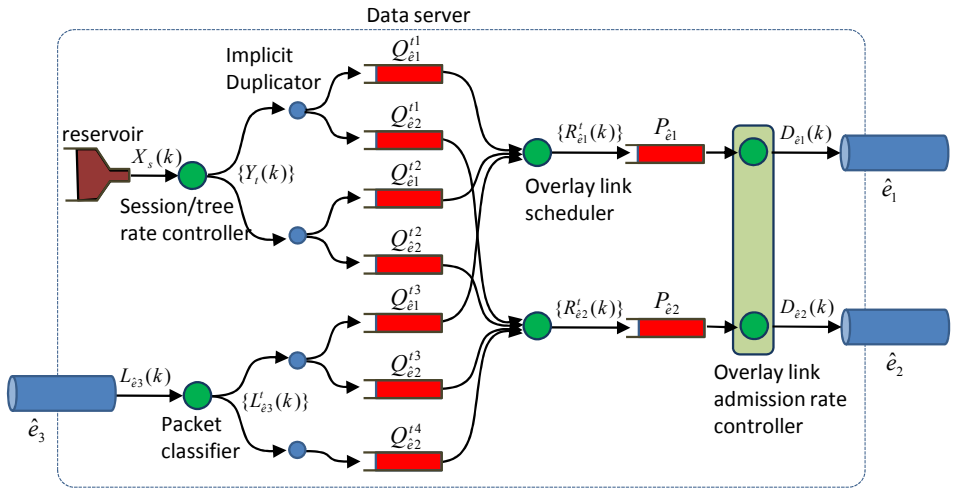


Fig. 3. How the traffic is handled in a data server.

C. Illustrating Example

We next illustrate how the algorithm works with an example. Noting that our algorithm uses the virtual queue sizes $q^t_{\hat{e}}$ and $p_{\hat{e}}$ for control, we pay particular attention to how the corresponding real queues are organized and updated. We assume that all the real queues are FIFO.

Consider a data server with two downstream overlay links and one upstream overlay link. Session s takes it as the source and distributes its content over two trees, t_1 and t_2. The data server is also an intermediate node of another session using trees t_3 and t_4. Fig. 3 shows how the traffic is handled in the data server.

Consider a fixed overlay link $\hat{e}$. Let $Q^t_{\hat{e}}$ be the real queue size of tree t's data waiting for transmission onto $\hat{e}$. After the overlay link scheduler, the packets coming out of each $Q^t_{\hat{e}}$ will be put into the queue $P_{\hat{e}}$ for $\hat{e}$.

We suppose that the trees t_1, t_2 use downstream links $\hat{e}_1$ and $\hat{e}_2$. The session/tree rate controller determines the session rate x_s and the tree rates $\{y_t\}$ for $t \in \{t_1, t_2\}$ according to (14) and (16). Then, it admits $Y_t(k) = y_t(k)$ real packets to the corresponding queues. The amount of admitted real packets for session s at time k is denoted by $X_s(k)$ and we have $X_s(k) = \sum_{t \in T_s} Y_t(k)$. Since both trees use the two downstream links, the admitted packets should be duplicated and put into the queues $Q^t_{\hat{e}_1}, Q^t_{\hat{e}_2}$ where $t \in \{t_1, t_2\}$. In an actual implementation, we do not need to duplicate the packets explicitly; it is enough to maintain pointers to the packets.

Let $L_{\hat{e}}(k)$ be the amount of real packets from an upstream overlay link $\hat{e}$ and arriving at the node $d(\hat{e})$ at time k. Let $L^t_{\hat{e}}(k)$ be the amount of real packets of tree t arriving at the node $d(\hat{e})$ at time k. We have $L_{\hat{e}}(k) = \sum_{t:\hat{e} \in \hat{E}_t} L^t_{\hat{e}}(k)$. Once the packets arrive, a packet classifier examines and distributes them into the appropriate queues.

Let $R^t_{\hat{e}}(k)$ be the amount of real packets transferred from queue $Q^t_{\hat{e}}$ to queue $P_{\hat{e}}$. The overlay link scheduler for each downstream overlay link $\hat{e}$ first determines $\{r^t_{\hat{e}}(k)\}$ for trees t such that $\hat{e} \in \hat{E}_t$, according to (15) and (16). Then, it sets $R^t_{\hat{e}}(k) = \min\{Q^t_{\hat{e}}(k), r^t_{\hat{e}}(k)\}$.

Let $D_{\hat{e}}(k)$ be the amount of real departures from queue $P_{\hat{e}}$ onto the overlay link $\hat{e}$. The overlay link admission

rate controller first determines $\nu_{\hat{e}}(k)$ according to (19) and (20). Then, it sets $D_{\hat{e}}(k) = \min\{P_{\hat{e}}(k), \nu_{\hat{e}}(k)\}$.

Then, the real queues $Q^t_{\hat{e}}$ and $P_{\hat{e}}$ evolve as follows.

$$Q^t_{\hat{e}}(k+1) = Q^t_{\hat{e}}(k) + L^t_{p(t,\hat{e})}(k) + Y_t(k) - R^t_{\hat{e}}(k),$$
$$P_{\hat{e}}(k+1) = P_{\hat{e}}(k) + \sum_{t:\hat{e}\in\hat{E}_t} R^t_{\hat{e}}(k) - D_{\hat{e}}(k).$$

The traffic leaving the queue $P_{\hat{e}}$ traverses through the underlay links corresponding to the overlay link $\hat{e}$. We assume that every underlay link has a single real queue. Let W_e be the real queue size at the underlay link e. Let $A_e(k)$ be the amount of real packets arriving at link e and $D_e(k)$ be the amount of departures from the link. Note that $D_e(k)$ = $\min\{c_e(k), W_e(k)\}$. Then, the queue W_e evolves as

$$W_e(k+1) = W_e(k) + A_e(k) - D_e(k).$$

D. Performance Analysis

In this section, we show the optimality and virtual queue stability of the algorithm using the Lyapunov optimization technique introduced in [19]–[21]. All the proofs are in Appendix A - D.

We first establish the bounds for a few terms. These are used in the proofs of the following lemmas. For every $\hat{e}$, the one time slot change of $q^t_{\hat{e}}(k)$ is bounded since the overlay transmission rate $r^t_{\hat{e}}(k)$ is bounded by R_{max} and the tree rate $y_t(k)$ is bounded by X_{max}. Define Q_{max} to be a constant such that for all time slots k,

$$\|q(k+1) - q(k)\|_1 \triangleq \sum_{s,t,\hat{e}} |q^t_{\hat{e}}(k+1) - q^t_{\hat{e}}(k)| \le Q_{max}. \tag{22}$$

Similarly, for every e, the one time slot change of $w_e(k)$ is bounded since $c_e(k)$ is bounded by c_e^{max} and $\sum_{\hat{e}:e\in E_{\hat{e}}} \nu_{\hat{e}}(k)$ is also bounded by $|\hat{E}|\nu_{max}$. Define W_{max} to be a constant such that for all time slots k,

$$\|w(k+1) - w(k)\|_1 \triangleq \sum_{e} |w_e(k+1) - w_e(k)| \le W_{max}. \tag{23}$$

Let us denote by $(x^*(k), y^*(k))$ an optimal solution to the following problem given the queue sizes $q(k)$ at time k,

$$(x^*(k), y^*(k)) \in \arg \max_{\substack{\sum_{t\in T_s} y_t = x_s,\\ 0\le x_s \le X_{max},\\ y_t \ge 0}} \Big\{ \sum_{s\in S} \Big(\frac{1}{\delta} U_s(x_s) - \sum_{t\in T_s} \sum_{\hat{e}\in\Omega(t,o(t))} q^t_{\hat{e}}(k) y_t \Big) \Big\}.$$

Lemma 1: Under the algorithm, there exists a constant $B_\Psi > 0$ such that for all time slots k,

$$\sum_{s\in S} \Big(\frac{1}{\delta} U_s(x_s(k)) - \sum_{t\in T_s} \sum_{\hat{e}\in\Omega(t,o(t))} q^t_{\hat{e}}(k) y_t(k) \Big)$$
$$\ge \sum_{s\in S} \Big(\frac{1}{\delta} U_s(x^*_s(k)) - \sum_{t\in T_s} \sum_{\hat{e}\in\Omega(t,o(t))} q^t_{\hat{e}}(k) y^*_t(k) \Big) - B_\Psi.$$

Let us denote by $r^*(k)$ an optimal solution to the following problem given the queue sizes $q(k)$ and $p(k)$ at time k,

$$r^*(k) \in \arg\max_{0\le r^t_{\hat{e}}\le R_{max}} \Big\{ \sum_{s,t,\hat{e}} \big(q^t_{\hat{e}}(k) - \sum_{\hat{e}'\in\Omega(t,d(\hat{e}))} q^t_{\hat{e}'}(k) - p_{\hat{e}}(k)\big) r^t_{\hat{e}} \Big\}.$$

Lemma 2: Under the algorithm, there exists a constant $B_\Phi > 0$ such that for all time slots $k > 0$,

$$\sum_{s,t,\hat{e}} \big(q^t_{\hat{e}}(k) - \sum_{\hat{e}'\in\Omega(t,d(\hat{e}))} q^t_{\hat{e}'}(k) - p_{\hat{e}}(k)\big) r^t_{\hat{e}}(k)$$
$$\ge \sum_{s,t,\hat{e}} \big(q^t_{\hat{e}}(k) - \sum_{\hat{e}'\in\Omega(t,d(\hat{e}))} q^t_{\hat{e}'}(k) - p_{\hat{e}}(k)\big) r^{*t}_{\hat{e}}(k) - B_\Phi.$$

Let $\nu^*(k)$ be an optimal solution to the following problem given the queue sizes $p(k)$ and $w(k)$ at time k,

$$\nu^*(k) \in \arg\max_{0\le\nu_{\hat{e}}\le\nu_{max}} \sum_{\hat{e}\in\hat{E}} \big(p_{\hat{e}}(k) - \sum_{e\in E_{\hat{e}}} w_e(k)\big)\nu_{\hat{e}}.$$

Lemma 3: Under the algorithm, there exists a constant $B_\Theta > 0$ such that for all time slots $k > 0$,

$$\sum_{\hat{e}\in\hat{E}} (p_{\hat{e}}(k) - \sum_{e\in E_{\hat{e}}} w_e(k))\nu_{\hat{e}}(k)$$
$$\ge \sum_{\hat{e}\in\hat{E}} (p_{\hat{e}}(k) - \sum_{e\in E_{\hat{e}}} w_e(k))\nu^*_{\hat{e}}(k) - B_\Theta.$$

Remark. The above lemmas imply that our algorithms (14)-(21) are constant approximation solutions to the corresponding subproblems on every time slot. The difference between the optimal value and the approximation increases with the maximum frame size.

To analyze the performance bound of our algorithm, we consider the following ϵ-tightened problem $\mathbf{P}(\epsilon)$.

$$\mathbf{P}(\epsilon): \max \sum_{s\in S} U_s(x_s)$$
$$\text{s.t. } r^t_{\hat{e}} - r^t_{p(t,\hat{e})} - y_t \mathcal{I}_{\{\hat{e}\in\Omega(t,o(t))\}} \ge \epsilon,$$
$$\forall t\in T, \forall \hat{e}\in\hat{E}_t, \tag{24}$$
$$\sum_{t:\hat{e}\in\hat{E}_t} r^t_{\hat{e}} \le \nu_{\hat{e}} - \epsilon, \forall \hat{e}\in\hat{E}, \tag{25}$$
$$\sum_{\hat{e}:e\in E_{\hat{e}}} \nu_{\hat{e}} \le \bar{c}_e - \epsilon, \forall e\in E, \tag{26}$$
$$\nu_{\hat{e}} \ge 0, \forall \hat{e}\in\hat{E},$$
$$\text{and the constraints (4), (6) - (8),}$$

where $0 < \epsilon < Y$ and Y is the largest value of ϵ such that the problem $\mathbf{P}(\epsilon)$ has a feasible solution. It is easy to see that any feasible solution of problem $\mathbf{P}(\epsilon)$ is feasible to problem $\mathbf{P}$.

Let (x^*, y^*, r^*, ν^*) and $(x^*(\epsilon), y^*(\epsilon), r^*(\epsilon), \nu^*(\epsilon))$ be some optimal solutions of problem $\mathbf{P}$ and $\mathbf{P}(\epsilon)$, respectively. Note that $(x^*(\epsilon), y^*(\epsilon), r^*(\epsilon), \nu^*(\epsilon))$ is feasible to problem $\mathbf{P}$. We use it in the optimality proof by comparing it with our solution.

Define U_{max} to be

$$U_{max} \triangleq \sum_{s \in S} \max_{0 \leq x_s \leq X_{max}} U_s(x_s).$$

Note that U_{max} is well defined. Let f^* and $f^*(\epsilon)$ be the optimal objective values of problem $\mathbf{P}$ and $\mathbf{P}(\epsilon)$, respectively. Let us also define $z(k)$ and $\|z\|_1(k)$ as follows.

$$z(k) \triangleq (q(k), p(k), w(k)),$$
$$\|z\|_1(k) \triangleq \sum_{s,t,\hat{e}} q^t_{\hat{e}}(k) + \sum_{\hat{e}} p_{\hat{e}}(k) + \sum_{e} w_e(k).$$

Then, we have the following theorem.

Theorem 4: For any parameter $\delta > 0$, the algorithm satisfies the following performance bounds.

$$\liminf_{k \to \infty} \sum_{s \in S} U_s\left(\frac{1}{k} \sum_{\kappa=0}^{k-1} \mathbb{E}\{x_s(\kappa)\}\right) \geq f^* - \frac{B_{\mathcal{A}} \delta}{2}, \tag{27}$$

$$\limsup_{k \to \infty} \frac{1}{k} \sum_{\kappa=0}^{k-1} \mathbb{E}\{\|z(\kappa)\|_1\} \leq \frac{B_{\mathcal{A}} + 2U_{max}/\delta}{2Y}, \tag{28}$$

where $B_{\mathcal{A}} = B + 2(B_\Psi + B_\Phi + B_\Theta)$, and B and Y are some constants.

Theorem 4 implies that the parameter δ can be used as a knob to tradeoff the optimization performance and the virtual queue size bounds. By choosing δ to be sufficiently small, the long-time average of the solution given by the algorithm can produce an objective value arbitrarily close to the optimal value of problem $\mathbf{P}$. However, the bound on the long-time average of the virtual queue sizes increases as δ decreases. The constant $B_{\mathcal{A}}$ includes the constants B_Ψ, B_Φ, and B_Θ due to the asynchronous operations of the algorithm by the two network layers. Hence, the larger the maximum frame size is, which implies a higher degree of asynchrony, the less tight the performance bounds are. Note that the boundedness of the virtual queue sizes does not directly imply network stability. We suspect that the real queue sizes in the network are also bounded and we will show simulation results to support the conjecture in Section IV.

IV. Simulation Experiments

In this section, we show simulation-based performance evaluation of the algorithm in Section III. We examine the rate optimality and the real queue boundedness of the algorithm.

For the stability of the real queues, we consider the following slightly modified system. When a source s injects real packets into a tree $t \in T_s$ at time k, we require that the source sends packets at a slightly smaller rate than the virtual tree rate $y_t(k)$ determined by our main algorithm. Specifically, the source injects $Y_t(k)$ real packets where $Y_t(k)$ is a random variable with mean $\beta y_t(k)$ for some β slightly less than 1. We call β the real traffic intensity. If

$y_t(k) = 0$, then $Y_t(k) = 0$. We also require that $Y_t(k)$ is bounded above by X_{max}. Using the concept of the real traffic intensity to show network stability can be seen in the literature [6], [22].

We wish to show that all the real queues $Q^t_{\hat{e}}$ and $P_{\hat{e}}$ in the data servers and the queues W_e at the underlay links are bounded. The aggregate queue size at time k is defined to be the sum of all the real queue sizes $\sum_{s,t,\hat{e}} Q^t_{\hat{e}}(k) + \sum_{\hat{e}} P_{\hat{e}}(k) + \sum_e W_e(k)$.

A. Simulation Setup

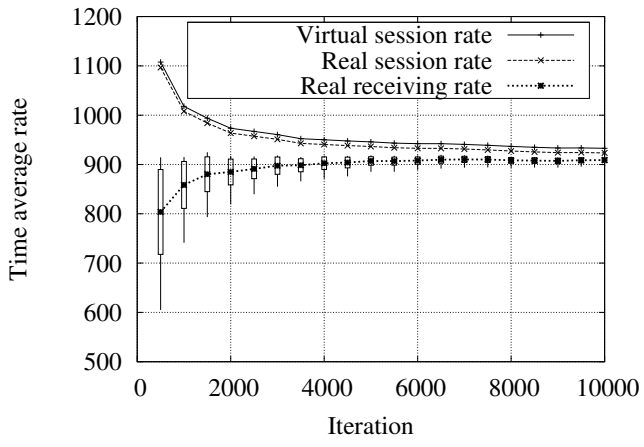
We use a real ISP network topology obtained from the Rocketfuel project [23], which consists of 41 routers and 136 underlay links. We attach 100 data servers randomly to the network as follows. Assuming we have five sessions, we assign 20 data servers per session. Then, for each session, we attach its 20 data servers randomly to the network such that no data servers in the session are attached to the same router. Most of the underlay links except some critical links have an average link capacity 1000 each. By critical links, we mean roughly the links that easily become the bottleneck if they do not have sufficient capacities. We assign an average link capacity 3000 to each critical link. We also assign a relatively large average link capacity 200000 to each of the links between a data server and its neighboring router so that they do not become the bottleneck. The underlay link capacities $c_e(k)$ vary randomly from time slot to time slot. They are i.i.d random processes with the uniform distributions and they are mutually independent.

We use the following heuristic to obtain the set of multicast trees for each session. We run the algorithm in [24], which computes the optimal set of trees (approximately), as well as their rate allocation and costs. The number of trees returned by that algorithm is very large[3]. We choose to use only 10 trees for each session in the increasing order of the tree cost. We choose $U_s(x_s) = \log(x_s + 1)$ as the utility function.

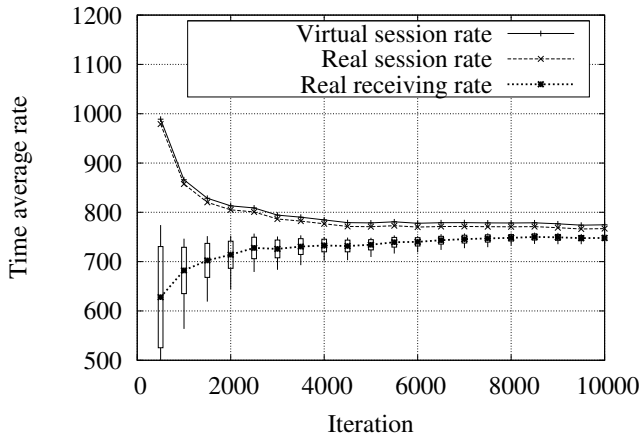
B. Single Session Case

In this subsection, we run the algorithm with only single session consisting of one source and 19 receivers and examine the performance of the algorithm. First, we show how the algorithm works when the overlay and underlay networks operate synchronously. In this experiment, we set both the overlay and underlay frame sizes to be 1. The value of δ is 10^{-6} and β is 0.99. Fig. 4(a) shows the convergence of the time average rates at every 500 iterations. The achieved session rate is 919, which is quite close to the rate 967 obtained by the algorithm in [24]. Note that our algorithm uses only 10 trees whereas the algorithm in [24] uses 22320 trees. We also plot the time average of the receiving rate in Fig. 4(a). Before reaching the steady state, the session rate can be different from each of the receiving rates because some packets are temporarily queued in the network. Each point on the dotted line represents the average across the receivers of the time average receiving rates at the corresponding iteration. The box centered around each point marks one standard deviation above or below the average rate, and the end points of the vertical line on each point represent the maximum and the minimum (with respect to the rate samples

[3]The number of trees returned by that algorithm is 22320 in the single session case and it is 70595 in the multiple session case.



(a) Synchronous case



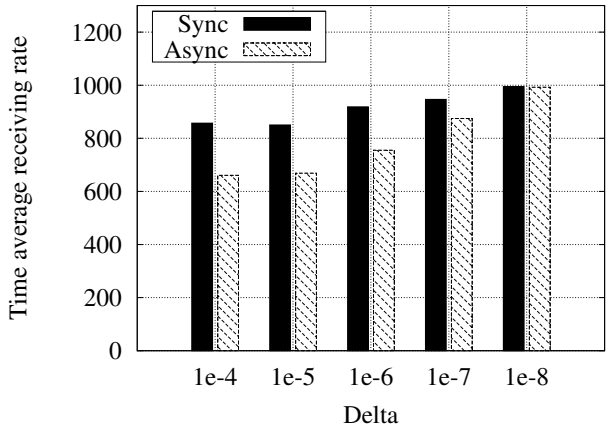
(b) Asynchronous case

Fig. 4. Time average rates.

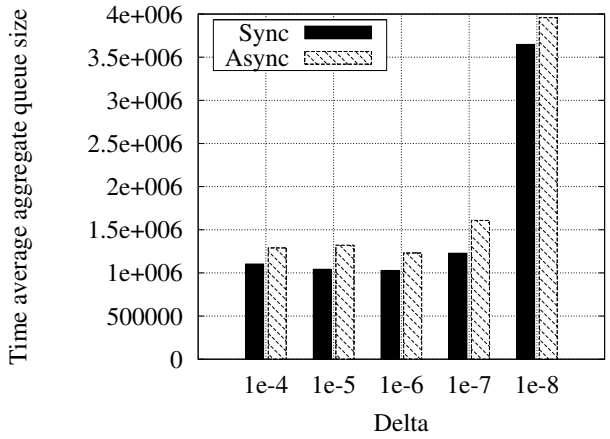
collected across the receivers at the corresponding iteration). One can see that the average of the time average receiving rates eventually approaches the time average session rate. This suggests that the queues in the network will not grow indefinitely. Moreover, the standard deviation of the time average receiving rates also decreases as convergence takes place.

Next, we run the algorithm asynchronously to show how the performance is affected by the asynchronous operations. In this experiment, the overlay frame sizes vary randomly between 1 and 2 and the underlay frame sizes vary between 2 and 3. All the other parameters are the same as the synchronous case. Fig. 4(b) shows the convergence of the time average rates. The achieved session rate is 756, which is 82% of the synchronous case. The performance degradation is because the algorithm often uses outdated information for control in the asynchronous operations.

Fig. 5 shows how the parameter δ affects the algorithm performance. In Fig. 5(a), we show the time average



(a) Effect on the achieved rate



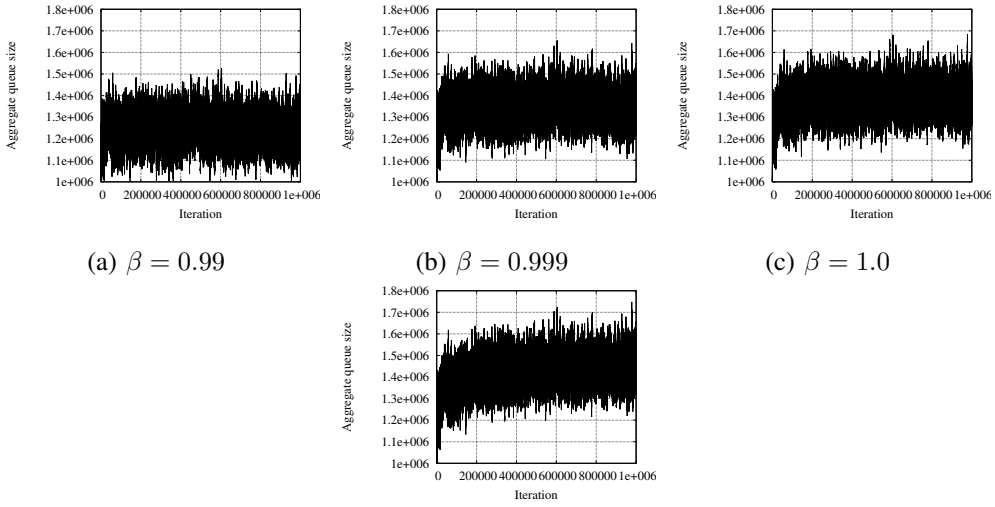
(b) Effect on the time average aggregate queue size

Fig. 5. Effect of δ on the achieved rate and on the time average aggregate queue size.

receiving rates at the 2×10^5th iteration for both the synchronous and asynchronous cases, with δ varying between 10^{-4} and 10^{-8}. As δ decreases, the achieved receiving rate tends to increase. However, the queue sizes in the network also tend to increase, as shown in Fig. 5(b). The figures also show that, as δ decreases, the achieved rate of the asynchronous case approaches that of the synchronous case.

We next examine the aggregate real queue size under the algorithm. In the experiment, we test four different real traffic intensities: $\beta = \{0.99, 0.999, 1.0, 1.001\}$. We run asynchronous operations and we use $\delta = 10^{-5}$. Fig. 6 shows the trajectory of the aggregate real queue size up to 10^6th iteration. The figure indicates that the real queues are stable if β is less than 1[4]. When the queues are stable, the upper bound of the aggregate queue size tends to increase as β increases.

[4]In this experiment, the aggregate real queue size looks stable even when $\beta = 1.0$. However, it is not always true as shown in the following subsection IV-D.



(a) $\beta = 0.99$ (b) $\beta = 0.999$ (c) $\beta = 1.0$

(d) $\beta = 1.001$

Fig. 6. Trajectory of the aggregate real queue size.

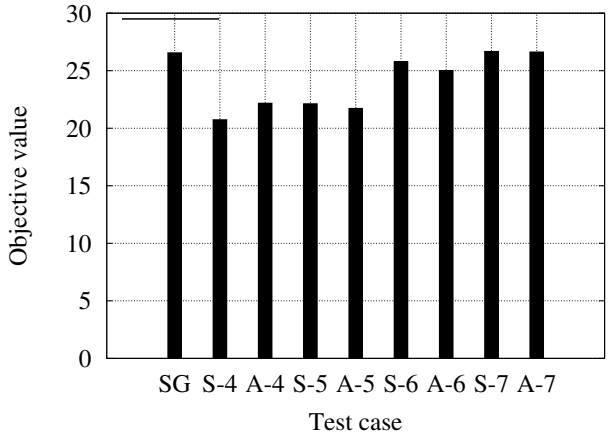
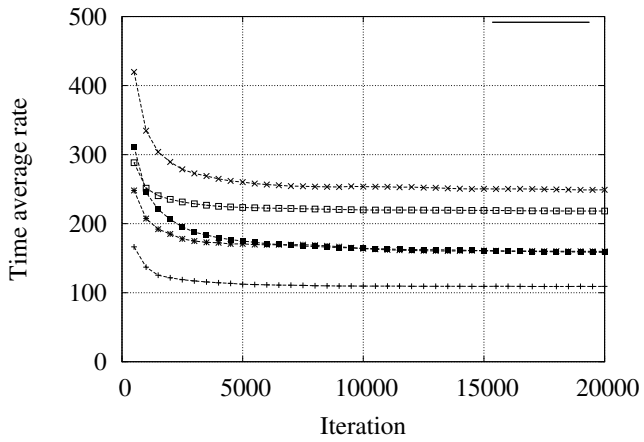


Fig. 7. Objective values of the cases of multiple sessions.

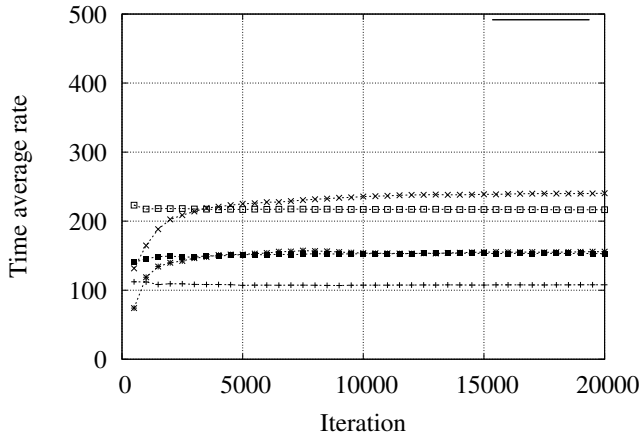
C. Case of Multiple Sessions

Now we consider the case of multiple sessions. We have five sessions and each session consists of one source and 19 receivers.

Fig. 7 plots the objective values obtained from various test cases with multiple sessions. Test case SG is the case of running the subgradient algorithm used in [24]. The achieved objective value in SG is 26.6, which is considered as the maximum objective value which can be obtained by our algorithm. Test cases with prefix S are run under the synchronous time scales, and those with prefix A are run under the asynchronous time scales. The suffix number indicates the value of parameter δ. For example, for the test case $S-4$, we set δ to be 10^{-4}. The figure shows that as δ decreases, our algorithm approaches the optimal objective value.



(a) Real session rates

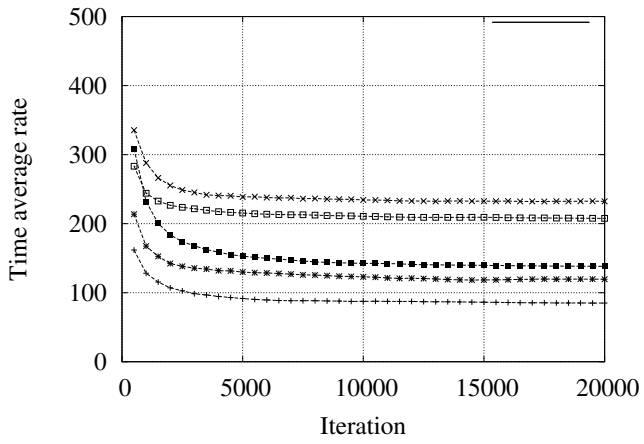


(b) Real receiving rates

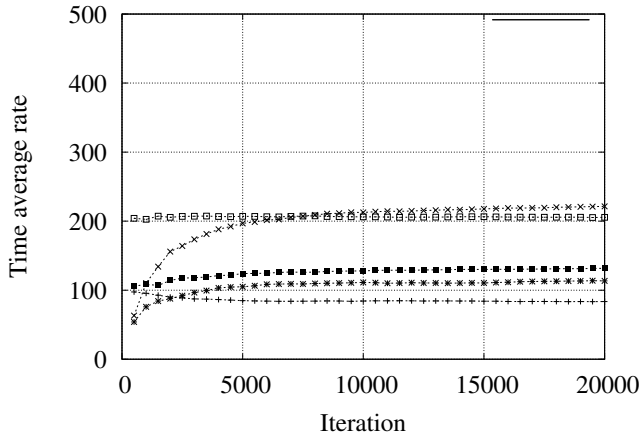
Fig. 8. Time average rates of multiple sessions under synchronous time scales.

Fig. 8 and Fig. 9 show how the session rates and receiving rates of the five sessions converge under the synchronous and asynchronous time scales, respectively. The value of δ is 10^{-6} and β is 0.99. The figures show that the asynchrony degrades the achieved session rates as in the single session case. Under the synchronous time scales, the achieved rate of the session experiencing the highest rate is 245 and the rate of the session experiencing the lowest rate is 107 (43% of the highest rate). Under the algorithm in [24], all the sessions have nearly the same achieved rate 200. It implies that under the above setup, our algorithm results in less fair rate allocation than the algorithm in [24]. Under the asynchronous time scales, the achieved rate of the session experiencing the highest rate is 229 and the rate of the session experiencing the lowest rate is 85 (37% of the highest rate). It implies that asynchrony also affects the fairness.

Fig. 10 plots how the rate allocation is achieved with various parameter δ under either synchronous or asynchronous times scales. The figure indicates that as δ decreases, the overall achieved rate increases and the fairness



(a) Real session rates



(b) Real receiving rates

Fig. 9. Time average rates of multiple sessions under asynchronous time scales.

among sessions improves as well. However, the improvement is obtained with the increase of the aggregate queue size as shown in Fig. 11.

We also examine the aggregate real queue size with the multiple sessions. Fig. 12 shows the trajectory of the aggregate queue size up to 10^6th iteration. As in the single session case, the figure also indicates that the real queues are stable if β is less than 1.

D. Example of Instable Real Queues When $\beta = 1.0$

In the above experiments, the real queues look stable even when the real traffic intensity β is 1.0. However, we have found that in some experiments, it is not true. For example, in the following experiment, the aggregate real queue size is not stable when β is 1.0. We run the algorithm with a single session consisting of single source and 9 receivers. Most of the parameters are the same as the single session case in subsection IV-B. Fig. 13 shows

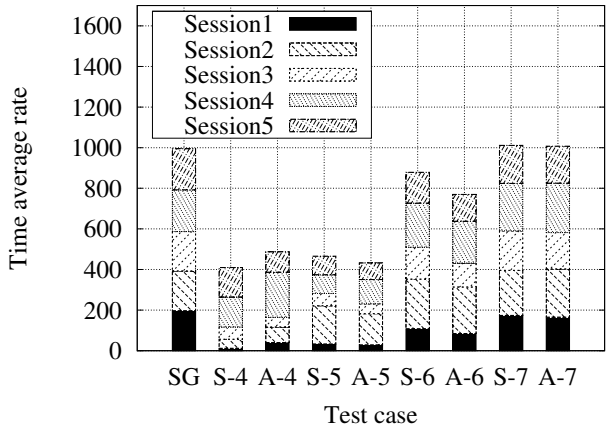


Fig. 10. Effect of δ on rate allocation.

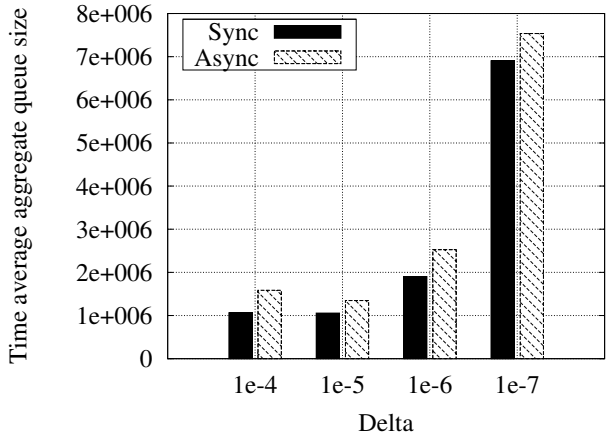
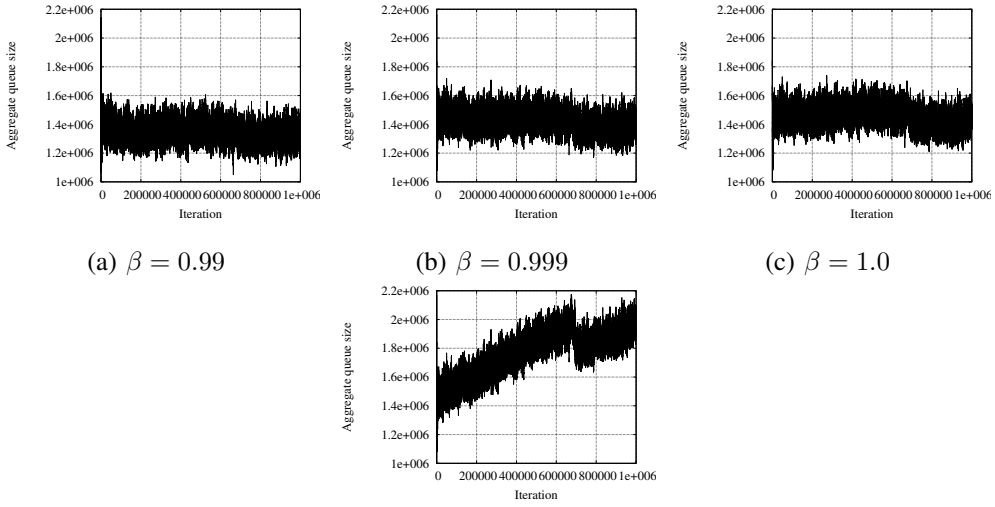


Fig. 11. Effect of δ on aggregate queue size.

that, in this test case, the aggregate real queue size continues to increase when $\beta = 1.0$ whereas it is stable when $\beta = 0.999$. It implies that the real traffic intensity 1.0 does not necessarily guarantee the real queue stability.

V. Concluding Remarks

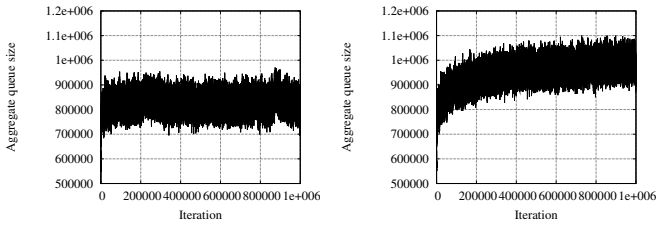
We have introduced a distributed backpressure algorithm for content distribution over a two-layer network, where the control variables can be adjusted at both layers. The algorithm can achieve optimal distribution performance with efficient use of the network resources. In the meantime, the algorithm is well divided between the overlay and underlay networks with minimal interaction and the two networks can operate asynchronously or on different timescales. We have proved its optimality using the Lyapunov optimization technique and showed network stability by simulations. The formal proof of the real queue boundedness is left to future research. We can extend our algorithm to allow more underlay controls, such as multi-path routing or dynamic routing for the underlay network,



(a) $\beta = 0.99$ (b) $\beta = 0.999$ (c) $\beta = 1.0$

(d) $\beta = 1.001$

Fig. 12. Trajectory of the aggregate real queue size.



(a) $\beta = 0.999$ (b) $\beta = 1.0$

Fig. 13. Example of instable aggregate real queue size when $\beta = 1.0$.

which can further improve the application performance and/or resource utilization.

APPENDIX A
PROOF OF LEMMA 1

Proof: Due to (22) and the boundedness of an overlay frame size, we have

$$\|q(k) - q(b^O(k))\|_1 \leq F_{max} Q_{max}, \tag{29}$$

for any k.

Let us define a function $D_\Psi(q)$ to be

$$D_\Psi(q) \triangleq \max_{\substack{\sum_{t \in T_s} y_t = x_s, \\ 0 \leq x_s \leq X_{max}, \\ y_t \geq 0}} \sum_s \left(\frac{1}{\delta} U_s(x_s) - \sum_t \sum_{\hat{e}} q_{\hat{e}}^t y_t \right).$$

Note that $D_\Psi(q)$ is a continuous function. Then, for any q' such that $\|q'-q\|_1 \le F_{max}Q_{max}$, there exists a constant B'_Ψ such that

$$|D_\Psi(q) - D_\Psi(q')| \le B'_\Psi. \tag{30}$$

Note also that

$$\begin{aligned}(x(k), y(k)) \in \ \arg \max_{\substack{\sum_{t\in T_s} y_t = x_s,\\ 0\le x_s \le X_{max},\\ y_t \ge 0}} \Big\{ \sum_{s\in S} \Big(\frac{1}{\delta} U_s(x_s) \\ - \sum_{t\in T_s} \sum_{\hat{e}\in\Omega(t,o(t))} q^t_{\hat{e}}(b^O(k)) y_t \Big) \Big\}. \end{aligned} \tag{31}$$

Then, we have

$$\begin{aligned}
&\sum_{s\in S} \Big(\frac{1}{\delta} U_s(x_s(k)) - \sum_{t\in T_s} \sum_{\hat{e}\in\Omega(t,o(t))} q^t_{\hat{e}}(k) y_t(k) \Big) \\
&= \sum_{s\in S} \Big(\frac{1}{\delta} U_s(x_s(k)) - \sum_{t\in T_s} \sum_{\hat{e}\in\Omega(t,o(t))} q^t_{\hat{e}}(b^O(k)) y_t(k) \Big) \\
&\quad - \sum_{s,t,\hat{e}} \big(q^t_{\hat{e}}(k) - q^t_{\hat{e}}(b^O(k)) \big) y_t(k) \\
&\ge D_\Psi(q(b^O(k))) - F_{max}Q_{max}X_{max} \\
&\ge D_\Psi(q(k)) - B'_\Psi - F_{max}Q_{max}X_{max},
\end{aligned}$$

where the first inequality is due to (29), (31) and the boundedness of $y_t(k)$, and the second inequality is due to (30). Letting $B_\Psi \triangleq B'_\Psi + F_{max}Q_{max}X_{max}$, the lemma holds. ■

APPENDIX B
PROOF OF LEMMA 2

Proof: Let $h^t_{\hat{e}}(k) \triangleq q^t_{\hat{e}}(k) - \sum_{\hat{e}'\in\Omega(t,d(\hat{e}))} q^t_{\hat{e}'}(k) - p_{\hat{e}}(k)$. For any time slots k, k' such that $|k-k'| \le F_{max}$,

$$\begin{aligned}
&\Big| \sum_{s,t,\hat{e}} h^t_{\hat{e}}(k) - \sum_{s,t,\hat{e}} h^t_{\hat{e}}(k') \Big| \\
&= \Big| \sum_{s,t,\hat{e}} \big(q^t_{\hat{e}}(k) - \sum_{\hat{e}'\in\Omega(t,d(\hat{e}))} q^t_{\hat{e}'}(k) - p_{\hat{e}}(k) \big) \\
&\qquad - \sum_{s,t,\hat{e}} \big(q^t_{\hat{e}}(k') - \sum_{\hat{e}'\in\Omega(t,d(\hat{e}))} q^t_{\hat{e}'}(k') - p_{\hat{e}}(k') \big) \Big| \\
&\le \Big| \sum_{s,t,\hat{e}} \big(q^t_{\hat{e}}(k) - q^t_{\hat{e}}(k') \big) \Big| \\
&\qquad + \Big| \sum_{s,t,\hat{e}} \sum_{\hat{e}'\in\Omega(t,d(\hat{e}))} \big(q^t_{\hat{e}'}(k) - q^t_{\hat{e}'}(k') \big) \Big| \\
&\qquad + \Big| \sum_{s,t,\hat{e}} \big(p_{\hat{e}}(k) - p_{\hat{e}}(k') \big) \Big| \\
&\le F_{max}Q_{max} + F_{max}|\hat{E}|Q_{max} + F_{max}|T|P_{max}.
\end{aligned}$$

Let $H_{max} \triangleq F_{max}Q_{max} + F_{max}|\hat{E}|Q_{max} + F_{max}|T|P_{max}$. Then, we have

$$\begin{aligned}
&\sum_{s,t,\hat{e}} h^t_{\hat{e}}(k) r^t_{\hat{e}}(k) \\
&= \sum_{s,t,\hat{e}} h^t_{\hat{e}}(b^O(k)) r^t_{\hat{e}}(k) - \sum_{s,t,\hat{e}} \big(h^t_{\hat{e}}(b^O(k)) - h^t_{\hat{e}}(k)\big) r^t_{\hat{e}}(k) \\
&\geq \sum_{s,t,\hat{e}} h^t_{\hat{e}}(b^O(k)) r^t_{\hat{e}}(k) - H_{max}R_{max} \\
&\geq \sum_{s,t,\hat{e}} h^t_{\hat{e}}(b^O(k)) r^{*t}_{\hat{e}}(k) - H_{max}R_{max} \\
&= \sum_{s,t,\hat{e}} h^t_{\hat{e}}(k) r^{*t}_{\hat{e}}(k) - \sum_{s,t,\hat{e}} \big(h^t_{\hat{e}}(k) - h^t_{\hat{e}}(b^O(k))\big) r^t_{\hat{e}}(k) \\
&\quad - H_{max}R_{max} \\
&\geq \sum_{s,t,\hat{e}} h^t_{\hat{e}}(k) r^{*t}_{\hat{e}}(k) - 2H_{max}R_{max}.
\end{aligned}$$

Letting $B_\Phi \triangleq 2H_{max}R_{max}$, the lemma holds. ■

APPENDIX C
PROOF OF LEMMA 3

Proof: Since, for any time slot k, $\nu_{\hat{e}}(k) = \nu_{\hat{e}}(b^U(k))$, we have, by (19),

$$\nu(k) \in \arg \max_{0 \leq \nu_{\hat{e}} \leq \nu_{max}} \sum_{\hat{e} \in \hat{E}} \big(p_{\hat{e}}(b^U(k)) - \sum_{e \in E_{\hat{e}}} w_e(b^U(k))\big)\nu_{\hat{e}}. \tag{32}$$

Due to (22), (23), and the boundedness of a frame length, for any ν such that $0 \leq \nu_{\hat{e}} \leq \nu_{max}$, we have

$$\begin{aligned}
&\Big| \sum_{\hat{e} \in \hat{E}} (p_{\hat{e}}(b^U(k)) - \sum_{e \in E_{\hat{e}}} w_e(b^U(k)))\nu_{\hat{e}} \\
&\qquad - \sum_{\hat{e} \in \hat{E}} (p_{\hat{e}}(k) - \sum_{e \in E_{\hat{e}}} w_e(k))\nu_{\hat{e}} \Big| \\
&= \Big| \sum_{\hat{e} \in \hat{E}} (p_{\hat{e}}(b^U(k)) - p_{\hat{e}}(k))\nu_{\hat{e}} \\
&\qquad + \sum_{\hat{e} \in \hat{E}} \sum_{e \in E_{\hat{e}}} (w_e(k) - w_e(b^U(k)))\nu_{\hat{e}} \Big| \\
&\leq F_{max}(P_{max} + |\hat{E}|W_{max})\nu_{max}.
\end{aligned} \tag{33}$$

Then, under the algorithm, we have

$$\begin{aligned}
&\sum_{\hat{e}\in\hat{E}}(p_{\hat{e}}(k)-\sum_{e\in E_{\hat{e}}}w_e(k))\nu_{\hat{e}}(k)\\
=&\sum_{\hat{e}\in\hat{E}}\big(p_{\hat{e}}(b^U(k))-\sum_{e\in E_{\hat{e}}}w_e(b^U(k))\big)\nu_{\hat{e}}(k)\\
&-\sum_{\hat{e}\in\hat{E}}\big(p_{\hat{e}}(b^U(k))-p_{\hat{e}}(k)\big)\nu_{\hat{e}}(k)\\
&-\sum_{\hat{e}\in\hat{E}}\big(\sum_{e\in E_{\hat{e}}}w_e(k)-\sum_{e\in E_{\hat{e}}}w_e(b^U(k))\big)\nu_{\hat{e}}(k)\\
\geq&\sum_{\hat{e}\in\hat{E}}\big(p_{\hat{e}}(b^U(k))-\sum_{e\in E_{\hat{e}}}w_e(b^U(k))\big)\nu_{\hat{e}}(k)\\
&-F_{max}(P_{max}+|\hat{E}|W_{max})\nu_{max},\\
\geq&\sum_{\hat{e}\in\hat{E}}\big(p_{\hat{e}}(b^U(k))-\sum_{e\in E_{\hat{e}}}w_e(b^U(k))\big)\nu^*_{\hat{e}}(k)\\
&-F_{max}(P_{max}+|\hat{E}|W_{max})\nu_{max},
\end{aligned}\tag{34}$$

where the first inequality is due to (33) and the second one is from (32). Further, we have

$$\begin{aligned}
\text{RHS of (34)}=&\sum_{\hat{e}\in\hat{E}}\big(p_{\hat{e}}(k)-\sum_{e\in E_{\hat{e}}}w_e(k)\big)\nu^*_{\hat{e}}(k)\\
&-\sum_{\hat{e}\in\hat{E}}\big(p_{\hat{e}}(k)-p_{\hat{e}}(b^U(k))\big)\nu^*_{\hat{e}}(k)\\
&-\sum_{\hat{e}\in\hat{E}}\big(\sum_{e\in E_{\hat{e}}}w_e(b^U(k))-\sum_{e\in E_{\hat{e}}}w_e(k)\big)\nu^*_{\hat{e}}(k)\\
&-F_{max}(P_{max}+|\hat{E}|W_{max})\nu_{max}\\
\geq&\sum_{\hat{e}\in\hat{E}}\big(p_{\hat{e}}(k)-\sum_{e\in E_{\hat{e}}}w_e(k)\big)\nu^*_{\hat{e}}(k)\\
&-2F_{max}(P_{max}+|\hat{E}|W_{max})\nu_{max},
\end{aligned}$$

where the inequality is due to (33). Letting $B_\Theta \triangleq 2F_{max}(P_{max}+|\hat{E}|W_{max})\nu_{max}$, the lemma holds. ■

APPENDIX D
PROOF OF THEOREM 4

Proof: Define the Lyapunov function $V(k)$ and the Lyapunov drift $\Delta(k)$ as follows:

$$\begin{aligned}
V(k)&\triangleq\sum_{s,t,\hat{e}}(q^t_{\hat{e}}(k))^2+\sum_{\hat{e}}(p_{\hat{e}}(k))^2+\sum_e(w_e(k))^2,\\
\Delta(k)&\triangleq\mathbb{E}\{V(k+1)-V(k)|z(k)\}.
\end{aligned}$$

By squaring both sides of the queue evolution equation in (17) followed by simple manipulations, we have

$$\begin{aligned}
&(q^t_{\hat{e}})^2(k+1) - (q^t_{\hat{e}})^2(k) \\
&\leq (r^t_{\hat{e}}(k) - r^t_{p(t,\hat{e})}(k) - y_t(k)\mathcal{I}_{\{\hat{e}\in\Omega(t,o(t))\}})^2 \\
&\quad - 2q^t_{\hat{e}}(k)(r^t_{\hat{e}}(k) - r^t_{p(t,\hat{e})}(k) - y_t(k)\mathcal{I}_{\{\hat{e}\in\Omega(t,o(t))\}}).
\end{aligned}$$

Summing the inequality over all $t \in T$ and all $\hat{e} \in \hat{E}_t$ and taking conditional expectations, we have

$$\begin{aligned}
&\mathbb{E}\big\{ \sum_{s,t,\hat{e}} (q^t_{\hat{e}}(k+1))^2 - \sum_{s,t,\hat{e}} (q^t_{\hat{e}}(k))^2 | z(k)\big\} \\
&\leq B_1 - 2\sum_{s,t,\hat{e}} q^t_{\hat{e}}(k)\mathbb{E}\big\{ r^t_{\hat{e}}(k) - r^t_{p(t,\hat{e})}(k) \\
&\qquad - y_t(k)\mathcal{I}_{\{\hat{e}\in\Omega(t,o(t))\}} | z(k)\big\}. \qquad (35)
\end{aligned}$$

where $B_1 \triangleq \sum_{s,t,\hat{e}} (R_{max} + X_{max})^2$.

Similarly, from the queue evolution equation in (18), we get

$$\begin{aligned}
&\mathbb{E}\big\{ \sum_{\hat{e}} (p_{\hat{e}}(k+1))^2 - \sum_{\hat{e}} (p_{\hat{e}}(k))^2 | z(k)\big\} \\
&\leq B_2 - 2\sum_{\hat{e}} p_{\hat{e}}(k)\mathbb{E}\big\{ \nu_{\hat{e}}(k) - \sum_{t:\hat{e}\in\hat{E}_t} r^t_{\hat{e}}(k) | z(k)\big\}, \qquad (36)
\end{aligned}$$

where $B_2 \triangleq \sum_{\hat{e}} (\max\{|T|R_{max}, \nu_{max}\})^2$.

Similarly, from the queue evolution equation in (21), we get

$$\begin{aligned}
&\mathbb{E}\big\{ \sum_{e} (w_e(k+1))^2 - \sum_{e} (w_e(k))^2 | z(k)\big\} \\
&\leq B_3 - 2\sum_{e} w_e(k)\mathbb{E}\{ c_e(k) - \sum_{\hat{e}:e\in E_{\hat{e}}} \nu_{\hat{e}}(k) | z(k)\} \\
&= B_3 - 2\sum_{e} w_e(k)\big(\bar{c}_e - \sum_{\hat{e}:e\in E_{\hat{e}}} \mathbb{E}\{\nu_{\hat{e}}(k) | z(k)\}\big), \qquad (37)
\end{aligned}$$

where $B_3 \triangleq \sum_e (\max\{C_{max}, \nu_{max}\})^2$.

Combining the above three inequalities (35), (36), (37), adding the term $-2/\delta \sum_{s\in S} \mathbb{E}\{U_s(x_s(k)) | z(k)\}$ to the

both sides of the inequality, and rearranging the terms, we have

$$\begin{aligned}
&\Delta(k) - \frac{2}{\delta}\sum_{s\in S}\mathbb{E}\{U_s(x_s(k))|z(k)\} \\
\leq\ & B_1 + B_2 + B_3 \\
&- 2\sum_{s\in S}\Big(\mathbb{E}\{\frac{1}{\delta}U_s(x_s(k))|z(k)\} \\
&- \sum_{t\in T_s}\sum_{\hat{e}\in\Omega(t,o(t))} q_{\hat{e}}^t(k)\mathbb{E}\{y_t(k)|z(k)\}\Big) \\
&- 2\sum_{s,t,\hat{e}}\Big(\big(q_{\hat{e}}^t(k) - \sum_{\hat{e}'\in\Omega(t,d(\hat{e}))} q_{\hat{e}'}^t(k) - p_{\hat{e}}(k)\big) \\
&\times \mathbb{E}\{r_{\hat{e}}^t(k)|z(k)\}\Big) \\
&- 2\sum_{\hat{e}}\big(p_{\hat{e}}(k) - \sum_{e\in E_{\hat{e}}} w_e(k)\big)\mathbb{E}\{\nu_{\hat{e}}(k)|z(k)\} \\
&- 2\sum_{e} w_e(k)\bar{c}_e. \qquad (38)
\end{aligned}$$

Define functions $\Psi(k)$, $\Phi(k)$ and $\Theta(k)$ as follows.

$$\begin{aligned}
\Psi(k) \triangleq\ & \sum_{s\in S}\mathbb{E}\{\frac{1}{\delta}U_s(x_s(k))|z(k)\} \\
&- \sum_{t\in T_s}\sum_{\hat{e}\in\Omega(t,o(t))} q_{\hat{e}}^t(k)\mathbb{E}\{y_t(k)|z(k)\}, \\
\Phi(k) \triangleq\ & \sum_{s,t,\hat{e}}(q_{\hat{e}}^t(k) - \sum_{\hat{e}'\in\Omega(t,d(\hat{e}))} q_{\hat{e}'}^t(k) - p_{\hat{e}}(k)) \\
&\times \mathbb{E}\{r_{\hat{e}}^t(k)|z(k)\}, \\
\Theta(k) \triangleq\ & \sum_{\hat{e}}(p_{\hat{e}}(k) - \sum_{e\in E_{\hat{e}}} w_e(k))\mathbb{E}\{\nu_{\hat{e}}(k)|z(k)\} \\
&+ \sum_{e} w_e(k)\bar{c}_e.
\end{aligned}$$

Let $\Psi^A(k), \Phi^A(k)$ and $\Theta^A(k)$ be the values of $\Psi(k), \Phi(k)$ and $\Theta(k)$ under the algorithm (14)-(21).

Let $\Psi^*(k)$ be the maximum of $\Psi(k)$ where $(x(k), y(k))$ satisfies the constraints (4), (6), (7) of problem $\mathbf{P}$. Then, using Lemma 1 and the fact that $(x^*(\epsilon), y^*(\epsilon), r^*(\epsilon), \nu^*(\epsilon))$ is feasible to problem $\mathbf{P}$ and $\mathbf{P}(\epsilon)$, we have

$$\begin{aligned}
\Psi^A(k) \geq\ & \Psi^*(k) - B_\Psi \\
\geq\ & \sum_{s\in S}\Big(\frac{1}{\delta}U_s(x_s^*(\epsilon)) - \sum_{t\in T_s}\sum_{\hat{e}\in\Omega(t,o(t))} q_{\hat{e}}^t(k)y_t^*(\epsilon)\Big) \\
&- B_\Psi.
\end{aligned}$$

Let $\Phi^*(k)$ be the maximum of $\Phi(k)$ where $r(k)$ satisfies the constraints (8) of problem $\mathbf{P}$. Then, using Lemma

2 and the feasibility of $(x^*(\epsilon), y^*(\epsilon), r^*(\epsilon), \nu^*(\epsilon))$ to problem $\mathbf{P}$ and $\mathbf{P}(\epsilon)$, we have

$$\begin{aligned}
\Phi^A(k) &\geq \Phi^*(k) - B_\Phi \\
&\geq \sum_{s,t,\hat{e}} (q^t_{\hat{e}}(k) - \sum_{\hat{e}' \in \Omega(t,d(\hat{e}))} q^t_{\hat{e}'}(k) - p_{\hat{e}}(k)) r^{t*}_{\hat{e}}(\epsilon) \\
&\quad - B_\Phi \\
&= \sum_{s,t,\hat{e}} q^t_{\hat{e}}(k)(r^{t*}_{\hat{e}}(\epsilon) - r^{t*}_{p(t,\hat{e})}(\epsilon)) - \sum_{\hat{e}} p_{\hat{e}}(k) r^{t*}_{\hat{e}}(\epsilon) \\
&\quad - B_\Phi \\
&\geq \sum_{s,t,\hat{e}} q^t_{\hat{e}}(k)(y^*_t(\epsilon)\mathcal{I}_{\{\hat{e} \in \Omega(t,o(t))\}} + \epsilon) \\
&\quad - \sum_{\hat{e}} p_{\hat{e}}(k) r^{t*}_{\hat{e}}(\epsilon) - B_\Phi.
\end{aligned}$$

Let $\Theta^*(k)$ be the maximum of $\Theta(k)$ where $\nu(k)$ satisfies the constraints $0 \leq \nu_{\hat{e}} \leq \nu_{max}$ for all $\hat{e}$ in $\hat{E}$. Then, using Lemma 3 and the feasibility of $(x^*(\epsilon), y^*(\epsilon), r^*(\epsilon), \nu^*(\epsilon))$ to problem $\mathbf{P}$ and $\mathbf{P}(\epsilon)$, we have

$$\begin{aligned}
\Theta^A(k) &\geq \Theta^*(k) - B_\Theta \\
&\geq \sum_{\hat{e}} (p_{\hat{e}}(k) - \sum_{e \in E_{\hat{e}}} w_e(k)) \nu^*_{\hat{e}}(\epsilon) + \sum_e w_e(k) \bar{c}_e \\
&\quad - B_\Theta \\
&= \sum_{\hat{e}} p_{\hat{e}}(k) \nu^*_{\hat{e}}(\epsilon) + \sum_{e \in E_{\hat{e}}} w_e(k)(\bar{c}_e - \sum_{\hat{e}: e \in E_{\hat{e}}} \nu^*_{\hat{e}}(\epsilon)) \\
&\quad - B_\Theta \\
&\geq \sum_{\hat{e}} p_{\hat{e}}(k) \nu^*_{\hat{e}}(\epsilon) + \epsilon \sum_{e \in E_{\hat{e}}} w_e(k) - B_\Theta.
\end{aligned}$$

From the above inequalities, we have

$$\begin{aligned}
&\Psi^A(k) + \Phi^A(k) + \Theta^A(k) \\
&\geq \sum_{s \in S} \Big(\frac{1}{\delta} U_s(x^*_s(\epsilon)) - \sum_{t \in T_s} \sum_{\hat{e} \in \Omega(t,o(t))} y^*_t(\epsilon) q^t_{\hat{e}}(k) \Big) \\
&\quad + \sum_{s,t,\hat{e}} q^t_{\hat{e}}(k)(y^*_t(\epsilon)\mathcal{I}_{\{\hat{e} \in \Omega(t,o(t))\}} + \epsilon) \\
&\quad + \sum_{\hat{e}} p_{\hat{e}}(k)(\nu^*_{\hat{e}}(\epsilon) - r^{t*}_{\hat{e}}(\epsilon)) \\
&\quad + \epsilon \sum_{e \in E_{\hat{e}}} w_e(k) - (B_\Psi + B_\Phi + B_\Theta) \\
&\geq \sum_{s \in S} \frac{1}{\delta} U_s(x^*_s(\epsilon)) + \epsilon \sum_{s,t,\hat{e}} q^t_{\hat{e}}(k) + \epsilon \sum_{\hat{e}} p_{\hat{e}}(k) \\
&\quad + \epsilon \sum_{e \in E_{\hat{e}}} w_e(k) - (B_\Psi + B_\Phi + B_\Theta),
\end{aligned}$$

where the second inequality is due to the cancelation of the common terms and the feasibility of $(x^*(\epsilon), y^*(\epsilon), r^*(\epsilon), \nu^*(\epsilon))$ to problem $\mathbf{P}(\epsilon)$.

Let $B_{\mathcal{A}} = B_1 + B_2 + B_3 + 2(B_\Psi + B_\Phi + B_\Theta)$. Applying the above inequalities to the drift expression in (38), we get

$$\begin{aligned} &\Delta(k) - \frac{2}{\delta}\sum_{s\in S}\mathbb{E}\{U_s(x_s(k))|z(k)\} \\ &\leq B_{\mathcal{A}} - \frac{2}{\delta}f^*(\epsilon) - 2\epsilon\|z\|_1(k). \end{aligned} \tag{39}$$

From (39), taking the expectation over the distribution of $z(k)$ and summing the inequality over $k \in \{0, 1, ..., K-1\}$, we get

$$\begin{aligned} &\mathbb{E}\{V(K) - V(0)\} - \frac{2}{\delta}\sum_{\kappa=0}^{K-1}\sum_{s\in S}\mathbb{E}\{U_s(x_s(\kappa))\} \\ &\leq KB_{\mathcal{A}} - \frac{2K}{\delta}f^*(\epsilon) - 2\epsilon\sum_{\kappa=0}^{K-1}\mathbb{E}\{\|z\|_1(\kappa)\}. \end{aligned} \tag{40}$$

Using the fact that removing the terms $\mathbb{E}\{V(K)\}$ and $-2\epsilon\sum_{\kappa=0}^{K-1}\mathbb{E}\{\|z\|_1(\kappa)\}$ preserves the inequality, rearranging (40), and taking the $\liminf$ as $K \to \infty$, we get

$$\liminf_{K\to\infty}\frac{1}{K}\sum_{\kappa=0}^{K-1}\sum_{s\in S}\mathbb{E}\{U_s(x_s(\kappa))\} \geq f^*(\epsilon) - \frac{B_{\mathcal{A}}\delta}{2}.$$

Using Jensen's inequality and letting $\epsilon \to 0$, we get (27).

On the other hand, from (40), using the definition of U_{max} and the fact that removing the terms $\mathbb{E}\{V(K)\}$ and $-2Kf^*(\epsilon)/\delta$ preserves the inequality, rearranging (40), and taking the $\limsup$ as $K \to \infty$, we get

$$\limsup_{K\to\infty}\frac{1}{K}\sum_{\kappa=0}^{K-1}\mathbb{E}\{\|z\|_1(\kappa)\} \leq \frac{B_{\mathcal{A}} + 2U_{max}/\delta}{2\epsilon}.$$

Letting $\epsilon \to Y$, we get (28). ■

REFERENCES

[1] T. Karagiannis, P. Rodriguez, and K. Papagiannaki, "Should Internet service providers fear peer-assisted content distribution?" in *Proc. ACM SIGCOMM conference on Internet Measurement*, 2005, pp. 6–6.

[2] H. Xie, Y. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz, "P4P: provider portal for applications," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 351–362, 2008.

[3] W. Jiang, R. Zhang-Shen, J. Rexford, and M. Chiang, "Cooperative content distribution and traffic engineering in an ISP network," in *Proc. ACM SIGMETRICS*, 2009, pp. 239–250.

[4] N. Chowdhury and R. Boutaba, "Network virtualization: state of the art and research challenges," *IEEE Communications Magazine*, vol. 47, no. 7, pp. 20–26, 2009.

[5] Y. Cui, Y. Xue, and K. Nahrstedt, "Optimal resource allocation in overlay multicast," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 8, pp. 808–823, 2006.

[6] L. Bui, R. Srikant, and A. Stolyar, "Optimal resource allocation for multicast sessions in multi-hop wireless networks," *Philosophical Ttransactions of the Royal Society A*, vol. 366, no. 1872, pp. 2059–2074, 2008.

[7] K. Kar, S. Sarkar, and L. Tassiulas, "A scalable low-overhead rate control algorithm for multirate multicast sessions," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 8, pp. 1541–1557, 2002.

[8] S. Sarkar and L. Tassiulas, "A framework for routing and congestion control for multicast information flows," *IEEE Transactions on Information Theory*, vol. 48, no. 10, pp. 2690–2708, 2002.

[9] K. Kar, S. Sarkar, and L. Tassiulas, "Optimization based rate control for multirate multicast sessions," in *Proc. IEEE INFOCOM*, 2001, pp. 123–132.

[10] S. Sarkar and L. Tassiulas, "Back pressure based multicast scheduling for fair bandwidth allocation," in *Proc. IEEE INFOCOM*, vol. 2, 2001, pp. 1123–1132.

[11] Y. Liu, H. Zhang, W. Gong, and D. Towsley, "On the interaction between overlay routing and underlay routing," in *Proc. IEEE INFOCOM*, 2005, pp. 2543–2553.

[12] D. DiPalantino and R. Johari, "Traffic engineering vs. content distribution: a game theoretic perspective," in *Proc. IEEE INFOCOM*, 2009, pp. 540–548.

[13] S. Ren, E. Tan, T. Luo, S. Chen, L. Guo, and X. Zhang, "TopBT: a topology-aware and infrastructure-independent BitTorrent client," in *Proc. IEEE INFOCOM*, 2010, pp. 1–9.

[14] D. Choffnes and F. Bustamante, "Taming the torrent: a practical approach to reducing cross-ISP traffic in peer-to-peer systems," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 363–374, 2008.

[15] V. Aggarwal, A. Feldmann, and C. Scheideler, "Can ISPs and P2P users cooperate for improved performance?" *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 3, pp. 29–40, 2007.

[16] R. Bindal, P. Cao, W. Chan, J. Medved, G. Suwala, T. Bates, and A. Zhang, "Improving traffic locality in BitTorrent via biased neighbor selection," in *IEEE International Conference on Distributed Computing Systems*, 2006, pp. 66–66.

[17] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 556–567, 2000.

[18] R. Srikant, *The mathematics of Internet congestion control*. Birkhauser, 2004.

[19] M. Neely, E. Modiano, and C. Rohrs, "Dynamic power allocation and routing for time-varying wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 1, pp. 89–103, 2005.

[20] L. Georgiadis, M. Neely, and L. Tassiulas, *Resource allocation and cross layer control in wireless networks*. Now Publishers Inc., 2006.

[21] M. Neely, E. Modiano, and C. Li, "Fairness and optimal stochastic control for heterogeneous networks," *IEEE/ACM Transactions on Networking*, vol. 16, no. 2, pp. 396–409, 2008.

[22] L. Bui, R. Srikant, and A. Stolyar, "Novel architectures and algorithms for delay reduction in back-pressure scheduling and routing," in *Proc. IEEE INFOCOM*, 2009, pp. 2936–2940.

[23] *Rocketfuel: an ISP topology mapping engine*, University of Washington, http://www.cs.washington.edu/research/networking/rocketfuel/.

[24] X. Zheng, C. Cho, and Y. Xia, "Content distribution by multiple multicast trees and intersession cooperation: optimal algorithms and approximations," in *Proc. IEEE Conference on Decision and Control*, 2009, pp. 5857–5862.