PHILOSOPHICAL TRANSACTIONS OF THE ROYAL SOCIETY A

Phil. Trans. R. Soc. A (2008) **366**, 2059–2074
doi:10.1098/rsta.2008.0016
Published online 6 March 2008

Optimal resource allocation for multicast sessions in multi-hop wireless networks

By Loc Bui[1], R. Srikant[1,*] and Alexander Stolyar[2]

[1]*Department of Electrical and Computer Engineering and Coordinated Science Lab, University of Illinois at Urbana-Champaign, IL 61801, USA*
[2]*Bell Labs, Alcatel-Lucent, NJ 07974, USA*

In this paper, we extend recent results on fair and stable resource allocation in wireless networks to include multicast sessions, in particular multi-rate multicast. The solution for multi-rate multicast is based on scheduling virtual (shadow) 'traffic' that 'moves' in reverse direction from destinations to sources. This shadow scheduling algorithm can also be used to control delays in wireless networks.

Keywords: wireless networks; multicast; resource allocation; congestion control; scheduling

1. Introduction

Multicast sessions are expected to be a common form of traffic in emerging mobile *ad hoc* networks (MANETs). However, the recently developed theory for fair resource allocation in MANETs (Lin & Shroff 2004; Eryilmaz & Srikant 2005, 2006; Neely *et al.* 2005; Stolyar 2005, 2006) addresses only the case of unicast flows. Other than developing appropriate notation, it is somewhat straightforward to extend the theory to multicast sessions if one assumes that data are delivered to all the receivers in a multicast group at the same rate. Such a form of multicast is called single-rate multicast. On the other hand, there are many video applications that allow layered transmission so that different receivers can subscribe to different numbers of layers and receive different qualities of the same video, depending on the congestion level in their respective neighbourhoods. Moreover, in wireless networks, due to varying signal strengths at different receivers, it may neither be desirable nor feasible to deliver data at the same rate to all the receivers in a multicast group. Thus, it is important to extend the optimization-based theory to handle multi-rate multicast sessions, i.e. multicast sessions where different receivers are allowed to receive at different rates. Such an extension is not immediate as in the case of single-rate multicast and is the main subject of this paper. We note that the multi-rate multicast problem has been considered in the context of wired networks (Kar *et al.* 2002; Deb & Srikant 2004; Srikant 2004; Kar & Tassiulas 2006). However, those approaches cannot be directly applied to wireless networks.

* Author for correspondence (rsrikant@uiuc.edu).

One contribution of 16 to a Discussion Meeting Issue 'Networks: modelling and control'.

This journal is © 2008 The Royal Society

For ease of exposition, we first present results for the single-rate multicast sessions. We then extend the results to multi-rate multicast sessions. The key idea is to introduce the concept of 'shadow traffic' generated by the receivers and 'moving back' to the sources, and the corresponding 'shadow' (token) queues. The movement of shadow traffic in the reverse direction then determines the real traffic generation (at the sources) and its movement through the network. Scheduling of the shadow traffic in the network is carried out by a back-pressure-type algorithm (Tassiulas & Ephremides 1992); however, this is a non-standard back-pressure algorithm. Finally, we prove that our proposed mechanism to control the real traffic using the shadow traffic is asymptotically optimal.

2. Single-rate multicast

(a) *Network model*

We consider a time-slotted, multi-hop wireless network that is modelled as a graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$ where $\mathcal{N}$ is the set of nodes and $\mathcal{L}$ is the set of directed links. If a link (n, m) is in $\mathcal{L}$, then it is possible to send packets from node n to node m up to a finite capacity c_{nm} and subject to the interference constraints. The interference constraints are specified in terms of maximal subsets of nodes, which can be scheduled simultaneously. These maximal subsets can be specified quite generally as we will see later. Moreover, we assume that when node n sends a packet to node m, each node k such that $(n, k) \in \mathcal{L}$ also hears that packet and can choose to receive it or not. This assumption models the broadcast nature of the wireless medium. We do not consider fading, but we can easily incorporate channel variations in wireless networks in our framework.

Let us consider both types of traffic in the network: unicast and multicast. Unicast traffic is represented by a set of unicast flows; each flow enters the network at its begin node and exits the network at its end node. Multicast traffic is represented by a set of multicast sessions; each session has a source node and a set of destination nodes. We assume that each multicast session has a given directed multicast tree, where the root of the tree is the source node of the multicast session and all destination nodes belong to the tree. For now, let us also assume that all receivers of a single multicast session have to be served at the same data rate (single-rate multicast). We will address multi-rate multicast later.

Let $\mathcal{F}$ be the set of unicast flows and $\mathcal{S}$ be the set of multicast sessions. For each flow $f \in \mathcal{F}$, let b_f, e_f and x_f denote the begin node, the end node and the rate of flow f, respectively. Let D be the set of end nodes for unicast traffic, i.e. $D = \{d \in \mathcal{N} : d = e_f \text{ for some } f \in \mathcal{F}\}$. For each multicast session $s \in \mathcal{S}$, let $T(s)$ be the corresponding multicast tree, $r(s)$ be the root of $T(s)$ (which is also the source node of the multicast session) and $\tilde{T}(s)$ be the subgraph of $T(s)$ obtained by removing all the leaf nodes. For each node $n \in \tilde{T}(s) \backslash \{r(s)\}$, let $p_s(n)$ denote the preceding node (parent node) of n in $T(s)$ and $C_s(n)$ denote the set of succeeding nodes (children nodes) of n in $T(s)$.

We assume that all sources are continuously backlogged. Each node maintains a single queue for those unicast flows that have the same destination, and a separate queue for each multicast session going through it. Let $q_n^d[t]$ denote the

length of the queue at node n and containing packets from unicast flows that are destined to node d at time t. Similarly, let $q_n^s[t]$ denote the length of the queue maintained at node n for multicast session s at time t.

Owing to the broadcast nature of the wireless medium, when node n sends out a packet, every node k such that $(n,k) \in \mathcal{L}$ also hears it. If that packet is a unicast one directed to node m, then the other nodes simply ignore it. Otherwise, if that packet is a multicast packet for multicast session s, then all nodes in $C_s(n)$ will receive it. At this point, let us define $\mu_{n,m}^d$ to be the long-term average transmission rate from node n to node m allocated to destination $d \in D$ and μ_n^s to be the long-term average transmission rate from node n to all nodes in $C_s(n)$ allocated to multicast session s. Also, let μ_n denote the service rate at node n, i.e.

$$\mu_n \triangleq \sum_{d \in D} \sum_{m:(n,m)\in\mathcal{L}} \mu_{n,m}^d + \sum_{s:n\in\tilde{T}(s)} \mu_n^s.$$

We now present the objective and various constraints in the optimization formulation of the resource allocation problem.

— Objective: $\max \sum_{f\in\mathcal{F}} U_f(x_f) + \sum_{s\in\mathcal{S}} \tilde{U}_s(\tilde{x}_s)$ where x_f and $\tilde{x}_s$ are the arrival rates of flow f and session s, respectively, and $U_f(\bullet)$ and $\tilde{U}_s(\bullet)$ are the associated utility functions for flow f and session s, respectively. All utility functions are assumed to be non-negative, increasing and concave. For the single-rate multicast sessions, the utility is defined with respect to the transmitter and not with respect to each receiver.

— Constraints on the unicast traffic at each node: $\forall d \in D, \forall n \in \mathcal{N}, n \neq d$,

$$\sum_{f\in\mathcal{F}} x_f \mathcal{I}_{\{n=b_f, d=e_f\}} + \sum_{k:(k,n)\in\mathcal{L}} \mu_{k,n}^d \leq \sum_{m:(n,m)\in\mathcal{L}} \mu_{n,m}^d.$$

This constraint is simply a flow-conservation constraint at each node.

— Constraints on the multicast traffic at each node:

$$\tilde{x}_s \leq \mu_{r(s)}^s, \quad \forall\, s \in \mathcal{S},$$

$$\mu_{p_s(n)}^s \leq \mu_n^s, \qquad \forall s \in \mathcal{S}, \qquad \forall n \in \tilde{T}(s)\backslash\{r(s)\}.$$

— Let $\Gamma = \{\gamma^1, \ldots, \gamma^{|\Gamma|}\}$ denote a collection of vectors in the $|\mathcal{N}|$-dimensional non-negative real space, representing the possible set of link rates that can be achieved in a given time slot. The network can choose among one of these vectors in Γ at each time instant. For example, if γ^1 is the chosen set of link rates at a particular time instant, then $\gamma^1(l)$, the l-th component of this vector, denotes the number of packets that can be transferred over link l at that time instant. We assume that zero is always a possible link rate on each link. Also, let $\hat{\Gamma} \triangleq \mathcal{CH}\{\Gamma\}$ denote the convex hull of the set Γ. It is well known that by time sharing between different rate vectors in Γ, any point in $\hat{\Gamma}$ can be attained. Then, the long-term service rate vector $\boldsymbol{\mu}$ has to satisfy the interference constraint: $\boldsymbol{\mu} \in \hat{\Gamma}$, i.e. there exists a $\{\alpha_i\}_{i=1}^{|\Gamma|}$ such that $\sum_{i=1}^{|\Gamma|} \alpha_i \leq 1$ and $\boldsymbol{\mu} = \sum_{i=1}^{|\Gamma|} \alpha_i \gamma^i$.

(*b*) *Optimization problem and Lagrangian decomposition*

Putting all the above elements of the model together, we have the following optimization problem:

$$\max \sum_{f\in\mathcal{F}} U_f(x_f) + \sum_{s\in\mathcal{S}} \tilde{U}_s(\tilde{x}_s), \tag{2.1}$$

$$\text{subject to } \sum_{f\in\mathcal{F}} x_f \mathcal{I}_{\{n=b_f, d=e_f\}} + \mu^d_{\sim,n} \le \mu^d_{n,\sim}, \quad \forall d\in D, \quad \forall n\in\mathcal{N}, \quad n\neq d, \tag{2.2}$$

$$\tilde{x}_s \le \mu^s_{r(s)}, \quad \forall s\in\mathcal{S}, \tag{2.3}$$

$$\mu^s_{p_s(n)} \le \mu^s_n, \quad \forall s\in\mathcal{S}, \quad \forall n\in\tilde{T}(s)\setminus\{r(s)\}, \tag{2.4}$$

$$\mu\in\hat{\Gamma}, \tag{2.5}$$

where

$$\mu^d_{\sim,n} \triangleq \sum_{k:(k,n)\in\mathcal{L}} \mu^d_{k,n}, \mu^d_{n,\sim} \triangleq \sum_{m:(n,m)\in\mathcal{L}} \mu^d_{n,m}.$$

At this point, let us define the *capacity region*, Λ, of the network as the set of flow rates $(\boldsymbol{x},\tilde{\boldsymbol{x}})\ge\mathbf{0}$ for which there exists a set $\mu\ge\mathbf{0}$ that satisfies the constraints (2.2)–(2.5). Then, the above optimization problem is equivalent to the problem $\max_{(\boldsymbol{x},\tilde{\boldsymbol{x}})\in\Lambda}\sum_{f\in\mathcal{F}} U_f(x_f) + \sum_{s\in\mathcal{S}}\tilde{U}_s(\tilde{x}_s)$. Owing to the concavity of $U_f(\bullet)$ and $\tilde{U}_s(\bullet)$ and the convexity of the capacity region Λ there exists a solution to this problem.

We rewrite the problem using Lagrange multipliers

$$\max \sum_{f\in\mathcal{F}} U_f(x_f) + \sum_{s\in\mathcal{S}} \tilde{U}_s(\tilde{x}_s) - \sum_{f\in\mathcal{F}} \lambda^{e_f}_{b_f} x_f + \sum_{d\in D} \sum_{n\in\mathcal{N}:n\neq d} \lambda^d_n\big(-\mu^d_{\sim,n} + \mu^d_{n,\sim}\big)$$
$$- \sum_{s\in\mathcal{S}} \lambda^s_{r(s)}\tilde{x}_s + \sum_{s\in\mathcal{S}} \lambda^s_{r(s)}\mu^s_{r(s)} + \sum_{s\in\mathcal{S}} \sum_{n\in\tilde{T}(s)\setminus\{r(s)\}} \lambda^s_n\Big(-\mu^s_{p_s(n)} + \mu^s_n\Big) \text{ subject to } \boldsymbol{\mu}\in\hat{\Gamma},$$

where $\lambda^{e_f}_{b_f}$, λ^d_n, $\lambda^s_{r(s)}$ and λ^s_n are the Lagrange multipliers corresponding to constraint (2.2)–(2.4), respectively. For notational completeness, let $\lambda^d_d = 0$, and $\lambda^s_m = 0$ if m is a leaf of $T(s)$. When the vector of λ's is specified, the above Lagrangian form of the optimization naturally decomposes into the following two problems. *The congestion control problem*

$$\max_{\boldsymbol{x},\tilde{\boldsymbol{x}}} \sum_{f\in\mathcal{F}} U_f(x_f) - \sum_{f\in\mathcal{F}} \lambda^{e_f}_{b_f} x_f + \sum_{s\in\mathcal{S}} \tilde{U}_s(\tilde{x}_s) - \sum_{s\in\mathcal{S}} \lambda^s_{r(s)}\tilde{x}_s, \tag{2.6}$$

and *the routing and scheduling problem*

$$\max_{\mu} \sum_{d\in D}\sum_{n\in\mathcal{N}} \lambda^d_n\big(-\mu^d_{\sim,n} + \mu^d_{n,\sim}\big) + \sum_{s\in\mathcal{S}} \lambda^s \mu^s_{r(s)} + \sum_{s\in\mathcal{S}} \sum_{n\in\tilde{T}(s)\setminus\{r(s)\}} \lambda^s_n\Big(-\mu^s_{p_s(n)} + \mu^s_n\Big)$$
$$= \max \sum_{d\in D}\sum_{(n,m)\in\mathcal{L}} \mu^d_{n,m}\big(\lambda^d_n - \lambda^d_m\big) + \sum_{s\in\mathcal{S}}\sum_{n\in\tilde{T}(s)} \mu^s_n\left(\lambda^s_n - \sum_{m\in C_s(n)} \lambda^s_m\right), \tag{2.7}$$
$$\text{subject to } \boldsymbol{\mu}\in\hat{\Gamma}.$$

The goal is to find a dynamic algorithm to compute the Lagrange multipliers. We describe such an algorithm in §2*c*.

(*c*) *Joint congestion control and scheduling algorithm*

(i) *Congestion controller*

At the beginning of time slot t, each flow, say f, and each session, say s, have access to the queue length of their first nodes, i.e. $q_{b_f}^{e_f}[t]$ and $q_{r(s)}^{s}[t]$. Then, motivated by (2.6), the instantaneous mean data rates $x_f[t]$ of flow f and $\tilde{x}_s[t]$ of session s are set as follows:

$$x_f[t] = \arg \max_{0 \le x \le M} \left(U_f(x) - \frac{q_{b_f}^{e_f}[t]}{K} x \right), \tag{2.8}$$

$$\tilde{x}_s[t] = \arg \max_{0 \le x \le M} \left(\tilde{U}_s(x) - \frac{q_{r(s)}^{s}[t]}{K} x \right), \tag{2.9}$$

where M, which is chosen to be large enough, represents the largest possible value of the incoming rates. The positive constant K will be used to guarantee convergence of the achieved rates to the fair allocation. In particular, we are interested in the performance of the system for large K.

Each source has to convert the instantaneous mean data rate determined by the congestion controller into a packet injection rate. While the instantaneous rate could be a non-negative real number, the number of packets injected into the network has to be a non-negative integer that could be determined by some complicated mechanism that converts rates to packets. Instead of precisely modelling this conversion, at each time slot t, the number of arrivals for flow f (session s) is assumed to be a Poisson random variable with mean x_f ($\tilde{x}_s$). This assumption can be easily relaxed in a number of ways without affecting our main conclusions.

(ii) *Back-pressure scheduler*

At time slot t, for each node n, we define the unicast differential backlog for destination node d as $w_n^d[t] = \max_{m:(n,m)\in\mathcal{L}} \left(q_n^d[t] - q_m^d[t]\right)$, and the multicast differential backlog for session s as $w_n^s[t] = \left(q_n^s[t] - \sum_{m\in C_s(n)} q_m^s[t]\right)$. The weight of node n at time slot t is defined as

$$w_n[t] = \max \left\{ \max_{d\in D} w_n^d[t], \max_{s:n\in\tilde{T}(s)} w_n^s[t] \right\}. \tag{2.10}$$

Then, motivated by (2.7), the service rate vector $\xi[t]$ is chosen from Γ so that it satisfies

$$\xi[t] \in \arg \max_{\pi\in\Gamma} \sum_{n\in\mathcal{N}} w_n[t]\pi_n. \tag{2.11}$$

At node n, the commodity (unicast flow or multicast session) whose differential backlog achieves the maximum in 2.10 will be served at rate $\xi_n[t]$. That commodity can be a unicast traffic destined to the destination d over link (n, m)

or a multicast traffic for session s. The rest of the queues at node n are not served at time slot t. We note that the back-pressure scheduler was originally developed in Tassiulas & Ephremides (1992) for the case of inelastic flows.

(d) *Performance analysis*

In this section, we present a result characterizing the performance of the joint congestion control and scheduling mechanism described in §2*c*.

To describe the evolution of the queues in the network, we adopt the following convention: the arrivals to a queue in a time slot are not available for service until the next time slot. If packets traverse from one queue to the next, then the arrivals to the second queue are equal to the number of departures from the first queue in the previous time slot. Thus, the evolution of queue lengths is governed by the following rules.

For each $n \in \mathcal{N}$ and $d \in D$,

$$q_n^d[t+1] = \left(q_n^d[t] - \xi_{n,\sim}^d[t]\right)^+ + \nu_{\sim,n}^d[t] + \sum_{f\in\mathcal{F}} a_f[t]\mathcal{I}_{\{n=b_f, d=e_f\}}, \tag{2.12}$$

and, similarly, for each $s \in \mathcal{S}$ and $n \in \tilde{T}(s)$,

$$q_n^s[t+1] = (q_n^s[t] - \xi_n^s[t])^+ + \tilde{a}_s[t]\mathcal{I}_{\{n=r(s)\}} + \nu_{p_s(n)}^s[t]\mathcal{I}_{\{n\neq r(s)\}}, \tag{2.13}$$

where $\nu_{\sim,n}^d[t]$ is the number of packets for destination d that are routed to node n from other nodes and $\nu_{p_s(n)}^s[t]$ is the number of packets for session s that arrive at node n from the preceding node $p_s(n)$ during time instant t. Also, $a_f[t]$ and $\tilde{a}_s[t]$ are the numbers of exogenous arrivals of flow f and session s, respectively. Note that, in general, $\nu_{\sim,n}^d[t] \leq \xi_{\sim,n}^d[t]$ and $\nu_{p_s(n)}^s[t] \leq \xi_{p_s(n)}^s[t]$, where

$$\xi_{\sim,n}^d \triangleq \sum_{k:(k,n)\in\mathcal{L}} \xi_{k,n}^d, \xi_{n,\sim}^d \triangleq \sum_{m:(n,m)\in\mathcal{L}} \xi_{n,m}^d.$$

Hence, $x_f[t] = \boldsymbol{E}[a_f[t]]$ and $\tilde{x}_s[t] = \boldsymbol{E}[\tilde{a}_s[t]]$. Then, we have the following proposition to establish the performance of the algorithm.

Proposition 2.1. *Under the joint congestion control and scheduling algorithm described in §2c, the queues in the network are stable (i.e. the Markov chain of queue occupancies is positive recurrent), and the source rates satisfy*

$$\sum_{f\in\mathcal{F}} U_f(\bar{x}_f) + \sum_{s\in\mathcal{S}} \tilde{U}_s(\bar{\tilde{x}}_s) \geq \sum_{f\in\mathcal{F}} U_f(x_f^\star) + \sum_{s\in\mathcal{S}} \tilde{U}_s(\tilde{x}_s^\star) - \frac{B}{K},$$

where

$$\bar{x}_f = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \boldsymbol{E}[x_f[t]], \quad \bar{\tilde{x}}_s = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \boldsymbol{E}[\tilde{x}_s[t]],$$

$(\boldsymbol{x}^\star, \tilde{\boldsymbol{x}}^\star)$ is an optimal solution to (2.1) and $B < \infty$ is some constant.

Proof. The proof is similar to the proofs in Stolyar (2005), Neely *et al.* (2005) and Eryilmaz & Srikant (2006) and, hence, is omitted. ∎

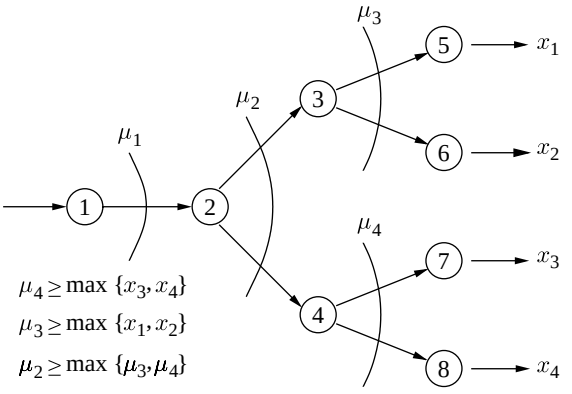

Figure 1. An example multi-rate multicast tree. The service rate offered at node 2 (which is μ_2) is at least the maximum among rates offered by its succeeding nodes, i.e. $\mu_2 \geq \max\{\mu_3, \mu_4\}$. Similar situations occur for nodes 3 and 4.

Hence, by tuning the parameter K, the algorithm can achieve a network utility that is arbitrarily close to optimal, while stabilizing the network queues.

We note that the algorithm (2.8)–(2.13) looks very similar to a dual algorithm to solve the optimization problem (2.1)–(2.5). However, there is a crucial difference: in (2.12) and (2.13), if one were to implement the dual algorithm from optimization theory, then $\nu^d_{\sim,n}[t]$ and $\nu^s_{p_s(n)}[t]$ have to be replaced by $\xi^d_{\sim,n}[t]$ and $\xi^s_{p_s(n)}[t]$, respectively. Thus, one cannot directly appeal to convergence results from optimization theory even if the arrival processes are assumed to be deterministic. However, the results in Stolyar (2005), Neely *et al.* (2005) and Eryilmaz & Srikant (2006) account for these differences from the dual algorithm and are applicable quite generally even with stochastic arrivals and channel variations.

3. Multi-rate multicast

In this section, we address the multi-rate multicast problem. In particular, we allow the receivers of a multicast session to be served at different rates. Recall that due to the broadcast assumption, whenever node n sends out a multicast packet for a multicast session s, all nodes in $C_s(n)$ will receive it. If a node is allowed to drop some packets before relaying the remaining packets along the multicast tree, then multi-rate multicast becomes feasible. To facilitate multi-rate multicast, each node has to offer the service rate that is at least the maximum among rates offered by its succeeding nodes (see figure 1 for an example of a multi-rate multicast tree). This leads to a totally different set of constraints compared with those in the case of single-rate multicast, as we will see shortly.

At this point, we have to introduce more notation. For each multicast session s, let $D(s)$ be the set of the destination nodes of s. Also, for each destination node $d \in D(s)$, let $\tilde{x}^d_s$ denote the average rate at which session s's traffic reaches d and $U_{d,s}(\tilde{x}^d_s)$ be the utility function associated with it. Again,

we assume that the utility functions are non-negative, increasing and concave. The objective of the resource allocation problem is to solve the following optimization problem:

$$\max \sum_{f\in\mathcal{F}} U_f(x_f) + \sum_{s\in\mathcal{S}} \sum_{d\in D(s)} \tilde{U}_{d,s}(\tilde{x}_s^d),$$

$$\text{subject to } \sum_{f\in\mathcal{F}} x_f \mathcal{I}_{\{n=b_f, d=e_f\}} + \mu_{\sim,n}^d \le \mu_{n,\sim}^d, \quad \forall\, d\in D, \quad \forall\, n\in\mathcal{N}, \quad n\neq d, \tag{3.1}$$

$$\tilde{x}_s^d \le \mu_{p_s(d)}^s, \quad \forall s\in\mathcal{S}, \quad \forall d\in D(s), \tag{3.2}$$

$$\mu_n^s \le \mu_{p_s(n)}^s, \quad \forall s\in\mathcal{S}, \quad \forall n\in\tilde{T}(s)\backslash\{r(s)\}, \tag{3.3}$$

$$\boldsymbol{\mu}\in\hat{\Gamma}. \tag{3.4}$$

Note that the constraints on unicast traffic (3.1) and the interference constraint (3.4) are the same as in the case of single-rate multicast. On the other hand, the constraints on multicast traffic (3.2) and (3.3) are different, due to the fact that each node has to offer the service rate that is at least the maximum among rates offered by its succeeding nodes. These constraints are somewhat problematic and necessitate a fresh look at solving the utility maximization problem for networks with the multi-rate multicast flows. To understand the source of difficulty, note that the constraints indicate that the data rate into a node ($\mu_{p_s(n)}^s$) should be greater than or equal to the data rate exiting that node (μ_n^s). Clearly, such a condition would not be desirable for queue-length stability if we were to maintain a queue at node n with arrival rate $\mu_{p_s(n)}^s$ and departure rate μ_n^s. For this reason, we will introduce a fictitious queue called *a shadow queue* that operates in the *reverse* direction, i.e. it pretends to store packets from multicast destinations to the source. The above constraints will ensure the stability of the shadow queues. The contents of the shadow queues will be used as tokens to inject traffic in the forward direction.

Let us define the *capacity region*, Ω, of the network as the set of flow rates $(\boldsymbol{x},\tilde{\boldsymbol{x}})\ge\mathbf{0}$ for which there exists a set $\boldsymbol{\mu}\ge\mathbf{0}$ that satisfies the constraints (3.1)–(3.4). Similar to the case of single rate, we can rewrite the optimization problem using Lagrange multipliers and decompose the Lagrangian form into the following two problems. *The congestion control problem*

$$\max_x \sum_{f\in\mathcal{F}} U_f(x_f) - \sum_{f\in\mathcal{F}} \lambda_{b_f}^{e_f} x_f + \sum_{s\in\mathcal{S}} \sum_{d\in D(s)} \tilde{U}_{d,s}(\tilde{x}_s^d) - \sum_{s\in\mathcal{S}} \sum_{d\in D(s)} \lambda_d^s \tilde{x}_s^d, \tag{3.5}$$

and *the routing and scheduling problem*

$$\max \sum_{d\in D} \sum_{(n,m)\in\mathcal{L}} \mu_{n,m}^d (\lambda_n^d - \lambda_m^d) + \sum_{s\in\mathcal{S}} \sum_{n\in\tilde{T}(s)} \mu_n^s \left(\sum_{m\in C_s(n)} \lambda_m^s - \lambda_n^s \right), \tag{3.6}$$

$$\text{subject to} \quad \boldsymbol{\mu}\in\hat{\Gamma},$$

where $\lambda_{b_f}^{e_f}$, λ_n^d, λ_s^d and λ_n^s are the Lagrange multipliers corresponding to constraint (3.1)–(3.3), respectively. For notational convenience, let $\lambda_d^d = 0$ and $\lambda_{r(s)}^s = 0$.

Again, our goal is to design a dynamic algorithm to compute the Lagrange multipliers.

(*a*) *Shadow algorithm for multi-rate multicast*

In this section, we present an algorithm to solve the above optimization problem. Since the service rates of the destinations in a multicast session are different, we cannot directly apply the congestion control algorithm that adjusts only the source node's rate, as in the case of single-rate multicast. Instead, in this case, the congestion control problem (3.5) suggests that the rate control should be handled at each receiver of each multicast session. We introduce a shadow joint congestion control and scheduling algorithm for this purpose. In the shadow algorithm, one pretends the direction of the traffic is from the multicast receivers to the source. Each node maintains a shadow queue that contains fictitious packets as though traffic is moving in the reverse direction, from destinations to sources of multicast flows. When a packet moves from a shadow queue to the next, then this packet is interpreted as a token to send a packet in the forward direction, which is the true direction. If time is slotted, one can easily implement such a mechanism by using a small fraction at the beginning (or end) of the slot to exchange shadow queue lengths, which is used for the updates of shadow queue lengths. The movement of shadow packets is determined by running a back-pressure algorithm on the shadow queues for the multicast flows and real queues for the unicast flows.

Note that these concepts of shadow queues and shadow traffic can be used for both multicast and unicast. However, for a resource allocation point of view, we do not need to deploy it for unicast (except for the reason of reducing delay, which we will discuss later in §4*b*). Therefore, let us just consider the case where shadow traffic is used only for multicast and real traffic for unicast traffic.

We now describe the shadow joint congestion control and scheduling algorithm. Suppose that n is a node that belongs to multicast tree $T(s)$ of session s and $C_s(n)$ is the set of succeeding nodes of n in $T(s)$. The shadow traffic will move from all nodes in $C_s(n)$ to node n. Let $\tilde{q}_s^m$ denote the shadow queue length for session s at node m. The source node $r(s)$, which is the destination for shadow traffic, has the shadow queue length always equal to zero. The shadow algorithm consists of the following parts.

(i) *The congestion control algorithm* (*for unicast traffic*)

At the beginning of time slot t, each flow f injects traffic into its corresponding entrance queue $q_{b_f}^{e_f}$ according to the following data rates:

$$x_f[t] = \arg \max_{0 \le x \le M} \left(U_f(x) - \frac{q_{b_f}^{e_f}[t]}{K} x \right).$$

(ii) *The shadow congestion control algorithm* (*for multicast traffic*)

At the beginning of time slot t, for each session s, each destination node $d \in D(s)$ injects shadow traffic into its corresponding 'entrance' shadow queue $\tilde{q}_d^s$ according to the following data rates:

$$\tilde{x}_s^d[t] = \arg \max_{0 \le x \le M} \left(U_{d,s}(x) - \frac{\tilde{q}_d^s[t]}{K} x \right).$$

As in the case of single-rate multicast, we will denote the number of shadow packets generated at destination d of session s at time t by $\tilde{a}_s^d[t]$, where $\tilde{a}_s^d[t]$ is a Poisson random variable with mean $\tilde{x}_s^d[t]$.

(iii) *The shadow back-pressure scheduling algorithm*

The back-pressure algorithm is run jointly for the shadow multicast traffic and the real unicast traffic. In particular, at time slot t, for each node n, we define the unicast differential backlog for destination node d as $w_n^d[t] = \max_{m:(n,m)\in\mathcal{L}} \left(q_n^d[t] - q_m^d[t]\right)$, and the shadow multicast differential backlog for session s as $w_n^s[t] = \left(\sum_{m\in C_s(n)} \tilde{q}_m^s[t] - \tilde{q}_n^s[t]\right)$. The weight of node n at time slot t is defined as

$$w_n[t] = \max\left\{\max_{d\in D} w_n^d[t], \max_{s:n\in\tilde{T}(s)} w_n^s[t]\right\}. \tag{3.7}$$

Then, the service rate vector $\boldsymbol{\xi}[t]$ is chosen from Γ such that

$$\boldsymbol{\xi}[t] \in \arg\max_{\boldsymbol{\pi}\in\Gamma} \sum_{n\in\mathcal{N}} w_n[t]\boldsymbol{\pi}_n. \tag{3.8}$$

At node n, the commodity whose differential backlog achieves the maximum in (3.7) will be served at rate $\boldsymbol{\xi}_n[t]$. That commodity can be real unicast traffic destined to destination d over link (n, m) or shadow multicast traffic for session s. The rest of the queues at node n are not served at time slot t.

Since the maximization in (3.8) may have many solutions, we assume that those solutions are indexed and scheduler always chooses the least index solution.

Note that the real unicast traffic is served at node n as usual. However, if, for example, k shadow packets of session s are served at node n in the links between $C_s(n)$ and n, then each shadow queue $\tilde{q}_m^s$ of each node $m \in C_s(n)$ is reduced by $\min\{k, \tilde{q}_m^s\}$, and shadow queue q_n^s is increased by $\min\{k, \max_{m\in C_s(n)} \tilde{q}_m^s\}$. In other words, all queues $\tilde{q}_m^s$, $m \in C_s(n)$ are served simultaneously, but the number of shadow packets arriving to $\tilde{q}_n^s$ is the maximum of what was served from each shadow input.

(iv) *Transform the shadow multicast traffic to the real multicast traffic*

At any time slot, each node will try to send the same amount of real packets as the number of shadow packets that it virtually received at that time slot. However, the actual number of packets sent could be fewer because the node may not have enough real packets to send. Let $\nu_n^s[t]$ denote the number of shadow packets of session s that node n virtually receives at time t and $q_n^s[t]$ denote the queue length for session s's real packets at node n at time t. Then, the actual number of real packets sent by node n at time t is $\min\{\nu_n^s[t], q_n^s[t]\}$.

Similarly, at any time slot, each node will receive the same number of real packets as the number of shadow packets that it virtually sent at that time slot. For example, suppose that, in a given slot, k shadow packets of session s are moved from $C_s(n)$ to n, but node m ($m \in C_s(n)$) sends only k' ($k' < k$) shadow packets to n (this can happen due to the manner in which shadow traffic is served as described above). Then, node n will send k real packets of flow s to each node in $C_s(n)$ in that time slot, but node m only accepts k' real packets and drops the rest $k - k'$. Also, when k shadow packets arrive at source node $r(s)$, the source will inject k 'new' real packets to the multicast tree $T(s)$.

We have that, for a given session s and a given node $n \in T(s)$, the update rule of the shadow queue $(\tilde{q}_n^s[t])$ is as follows:

$$\tilde{q}_n^s[t+1] = \Big(\tilde{q}_n^s[t] - \xi_{p_s(n)}^s[t]\Big)^+ + \max\big\{\tilde{a}_s^n[t]\mathcal{I}_{\{n\in D(s)\}}, \nu_n^s[t]\big\}, \tag{3.9}$$

where $\tilde{a}_s^d[t]$ is the number of exogenous shadow arrivals of session s at destination $d \in D(s)$; $\xi_n^s[t]$ is the amount of shadow packets that are *scheduled* to be served over links between $C_s(n)$ and n (defined in (3.8)); and $\nu_n^s[t]$ is the amount of shadow packets that are *actually* served over links between $C_s(n)$ and n. Note that

$$\nu_n^s[t] = \min\Big\{\xi_n^s[t], \max_{m\in C_s(n)} \tilde{q}_m^s\Big\}.$$

In equation (3.9), the number of packets that the shadow queue $\tilde{q}_n^s$ receives at time t is $\max\{\tilde{a}_s^n[t]\mathcal{I}_{\{n\in D(s)\}}, \nu_n^s[t]\}$ since we allow the case when some destinations are not leaves of the tree. Also, the number of packets that the shadow queue $\tilde{q}_s^n$ sends at time t is $\min\{\tilde{q}_n^s[t], \xi_{p_s(n)}^s[t]\}$. Therefore, the update rule for the real queue $(q_n^s[t])$ is as follows:

$$\begin{aligned} q_n^s[t+1] = {} & \big(q_n^s[t] - \max\big\{\tilde{a}_s^n[t]\mathcal{I}_{\{n\in D(s)\}}, \nu_n^s[t]\big\}\big)^+ \\ & + \min\Big\{\tilde{q}_n^s[t], \xi_{p_s(n)}^s[t], q_{p_s(n)}^s[t]\Big\}. \end{aligned} \tag{3.10}$$

Intuitively, the shadow traffic can be viewed as the calculated capacity for the real traffic.

(*b*) *Stability and performance*

We note that the shadow congestion control and scheduling algorithms actually mimic the one described in §2*c*. Therefore, the proof of proposition 2.1 can be used to show that the shadow congestion control and scheduling algorithm achieves a network utility that is arbitrarily close to the optimal one, and also stabilizes the real unicast queues and *the shadow multicast queues*. In particular, the following proposition can be stated.

Proposition 3.1. *Under the shadow congestion control and scheduling algorithms described in* §3*a, the source rates satisfy*

$$\sum_{f\in\mathcal{F}} U_f(\bar{x}_f) + \sum_{s\in\mathcal{S}}\sum_{d\in D(s)} \tilde{U}_{d,s}\Big(\bar{\tilde{x}}_s^d\Big) \geq \sum_{f\in\mathcal{F}} U_f\big(x_f^\star\big) + \sum_{s\in\mathcal{S}}\sum_{d\in D(s)} \tilde{U}_s(\tilde{x}_s^{\star\, d}) - \frac{B}{K},$$

where

$$\bar{x}_f = \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1} \boldsymbol{E}[x_f[t]], \quad \bar{\tilde{x}}_s^d = \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1} \boldsymbol{E}\big[\tilde{x}_s^d[t]\big]$$

and $B<\infty$ is some constant. Also, all real unicast queues and shadow multicast queues are stable (*i.e. the Markov chain of queue occupancies is positive recurrent*).

Proof. The proof of this proposition is similar to the proof of proposition 2.1. ■

Now we address the queue-length stability of the real queues. Recall that at any time slot, each node will send (or receive) at most as many real packets as what it virtually received (or sent) at that time slot. Further, if a node receives more real packets than what it virtually sent, it will drop the excess amount. Thus, it is important to understand whether this packet-dropping mechanism can stabilize the real queues.

Let us introduce some more notations. Consider a multicast tree $T(s)$ with the root $r(s)$. For any node $n \in T(s)$, let $A_s(n)$ be the set of nodes in the path from n to $r(s)$ including n. Also, let l_n^s be the level of node n in $T(s)$, i.e. $l_n^s = |A_s(n)| - 1$ where $|S|$ denotes the cardinality of set S.

For each $s \in \mathcal{S}$ and $n \in T(s)$, let us define that $\alpha_n^s[t]$ and $\eta_n^s[t]$ are the actual numbers of real packets arriving to and departing from the real queue q_n^s at time t, respectively. Similarly, let $\tilde{\alpha}_n^s[t]$ and $\tilde{\eta}_n^s[t]$ be the actual numbers of shadow packets arriving to and departing from the shadow queue $\tilde{q}_n^s$ at time t, respectively. Comparing with (3.9), we have

$$\tilde{\alpha}_n^s[t] = \max\{\tilde{a}_s^n[t]\mathcal{I}_{\{n \in D(s)\}}, \nu_n^s[t]\},$$

$$\tilde{\eta}_n^s[t] = \min\left\{\tilde{q}_n^s[t], \xi^s_{p_s(n)}[t]\right\}.$$

Also, the update rule (3.10) becomes

$$q_n^s[t+1] = (q_n^s[t] - \tilde{\alpha}_n^s[t])^+ + \min\left\{\tilde{\eta}_n^s[t], q^s_{p_s(n)}[t]\right\}.$$

In other words, the shadow packets are the available tokens for sending real packets. Note that the tokens are *instantaneous permits* to send packets. If the tokens are not used within the same time slot that they are generated, then they are lost. These tokens cannot be stored by the receiving node for future use. This is different from the usual sense in which tokens are interpreted in communication networks.

To prove the stability of the real queues, we consider a *modified system* as follows: instead of sending up to the number of available tokens, each node sends a slightly smaller number of real packets. More precisely, suppose that node n received $k > 0$ shadow packets in a time slot, then it will try to send k real packets (or all the packets in the queue if the queue length is smaller than k) with probability $(1 - \epsilon/k)$, or simply drop k packets from its queue with probability ϵ/k. Thus, the traffic is *thinned* as it proceeds along a multicast tree.

Note that as ϵ goes to zero, this *modified system* will emulate the *original system*. Next, we will prove the stability of the *modified system*.

Proposition 3.2. *For the modified system, the Markov chain* $(\boldsymbol{q}[t], \tilde{\boldsymbol{q}}[t])$ *is positive recurrent and*

$$\boldsymbol{E}(\|(\boldsymbol{q}[\infty], \tilde{\boldsymbol{q}}[\infty])\|) < \infty, \tag{3.11}$$

where the time ∞ *indicates the stationary regime. Further,*

$$\boldsymbol{E}(\tilde{\alpha}_n^s[\infty]) - l_n^s \epsilon \le \boldsymbol{E}(\eta_n^s[\infty]) \le \boldsymbol{E}(\tilde{\alpha}_n^s[\infty]), \tag{3.12}$$

where ϵ *is the parameter of the modified system that was described before the statement of the proposition.*

Proof. It is easy to see that $(\boldsymbol{q}[t], \tilde{\boldsymbol{q}}[t])$ forms a Markov chain that is irreducible and aperiodic.

Recall that for the shadow queue of any node $n \in T(s)$ of session $s \in \mathcal{S}$, we have

$$\tilde{q}_n^s[t] = \tilde{q}_n^s[0] + \sum_{k=0}^{t-1} \tilde{\alpha}_n^s[k] - \sum_{k=0}^{t-1} \tilde{\eta}_n^s[k], \quad \forall t > 0.$$

Since $\tilde{q}_n^s$ is positive recurrent, by the ergodic theorem (Norris 1998),

$$\lim_{T\to\infty} \frac{1}{N} \sum_{k=0}^{N-1} \tilde{\eta}_n^s[k] = \boldsymbol{E}[\tilde{\eta}_n^s[\infty]] \quad \text{almost surely}$$

$$\lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N-1} \tilde{\alpha}_n^s[k] = \boldsymbol{E}[\tilde{\alpha}_n^s[\infty]] \quad \text{almost surely}$$

where the time ∞ indicates *stationary regime*. Further, the stability of the shadow queues implies

$$\boldsymbol{E}[\tilde{\eta}_n^s[\infty]] = \boldsymbol{E}[\tilde{\alpha}_n^s[\infty]]. \tag{3.13}$$

Now, let the modified real queueing system be

$$q_n^s[t+1] = (q_n^s[t] - \hat{\alpha}_n^s[t])^+ + \hat{\eta}_n^s[t],$$

where $\hat{\alpha}_n^s$ and $\hat{\eta}_n^s$ are the 'service' and arrival processes, respectively, for each real queue. By construction, we have that for every t,

$$\hat{\alpha}_n^s[t] = \tilde{\alpha}_n^s[t], \tag{3.14}$$

$$\boldsymbol{E}[\hat{\eta}_n^s[t] | \tilde{\eta}_n^s[t]] \le \tilde{\eta}_n^s[t] - \epsilon', \quad \text{when} \quad \tilde{\eta}_n^s[t] > 0, \tag{3.15}$$

where $\epsilon' > 0$ is a constant depending on ϵ and $\epsilon' \to 0$ as long as $\epsilon \to 0$. These estimates, along with (3.13), show that each real queue will have a negative average drift, as long as it is large enough. This allows us to establish the positive recurrence, as well as (3.11) (we omit details). The estimate (3.12) follows from the fact that at any time, the maximum possible expected number of real packets that can be dropped by any given node due to thinning is ϵ. ■

The above proposition shows that the rate at which real packets reach each destination is close to the token generation rate at that destination. Note that the above proposition is also valid if 'thinning' is done only at the source and not at every intermediate node. In this case, one can show that the arrival rate is less than that of the available service rate at each queue by an induction argument starting at the source of each multicast tree.

(*c*) *The finite-buffer case*

In §3*b*, we showed the stability of the multicast congestion control algorithm by thinning the traffic slightly and transmitting at a rate slightly smaller than the token generation rate. In this section, we consider the case of a network with finite-buffer queues. For such a network, stability is not an issue. However, we still have to prove that the rate at which real packets reach each destination is close to the solution obtained from the network utility maximization problem, at least when the buffer size at each queue is large. For simplicity, we will assume that the shadow algorithm is used by all flows, unicast or multicast.

Suppose that each queue has a maximum queue length $H < \infty$, i.e. packets that arrive when the queue length reaches H will be dropped. The real packet transmission scheme for this system is as follows: when a node receives a certain number of tokens for a flow, the node transmits as many real packets as the number of tokens received if that many real packets are available in that flow's queue; else it transmits all the real packets in the queue. In particular, no thinning is used as in the case of the infinite-buffer queue. We will use a coupling argument to relate this finite-buffer model to the infinite-buffer model considered earlier to establish our result.

Let us consider the following queueing systems. All of them are coupled, i.e. their evolution processes are constructed on the same probability space in a natural manner.

SYSTEM 1 (*S1*). The *infinite-buffer system* considered in §3*b* with thinning parameter ϵ.

— From proposition 3.2, we know that $\boldsymbol{E}[\|(\boldsymbol{q}[\infty], \tilde{\boldsymbol{q}}[\infty])\|] < \infty$. Let us call this fact 1.
— From proposition 3.2, we also know that, for any receiver, the real received rate is close to its token injection rate which is close to the optimal solution (fact 2).

SYSTEM 2 (*S2*). The queueing dynamics of this system are the same as those of S1 with two modifications. At any time t, after completion of all arrivals and departures that occur in the transition from time $t-1$ to time t, any packet in any queue that has H or more packets ahead of it in the queue is coloured red. We will also give high priority to uncoloured packets in our queueing network. In other words, when a new uncoloured packet arrives at a queue, this packet is inserted in front of all the coloured packets. (If after completion of all arrivals and departures, this packet will have H or more packets in front of it, it will be coloured.) Note that this does not alter the total queue length at any queue, it only ensures that coloured packets will be served last in a queue. It follows from fact 1 that, in the stationary regime, the fraction of coloured packets can be made arbitrarily small by making H large (fact 3). (More precisely, the average rate at which packets are being coloured in the network will be small.) It then follows from facts 2 and 3 that, when H is sufficiently large, in S2 the received rate of uncoloured real packets at any receiver is close to its token injection rate, which in turn is close to the optimal solution (fact 4).

SYSTEM 3 (*S3*). This queueing system is similar to S2, but the real traffic is not thinned, i.e. we let ϵ be equal to zero. Instead, those packets which would have been 'thinned out' at a source or any intermediate node are also coloured red. As in S2, the priority scheme for this queueing system gives the highest priority to uncoloured packets—they are always in front of the coloured packets in any queue. Note that this queueing system may be unstable, but its stability does not matter to us. Clearly, fact 4 holds for S3 as well as for S2: when H is large enough, the received rate of uncoloured real packets at any receiver is close to its token injection rate which is close to the optimal solution (fact 5).

SYSTEM 4 (*S4*). Same as S3, except that any packet in any queue which is above threshold H will be *dropped*. In other words, S4 is the *true system* with finite buffers.

Theorem 3.3. *The processes of uncoloured packets in S4 and S3* (*and S2*) *are identical, i.e. the queue lengths at any node at any time are equal in both systems.*

Proof. The proof follows from the construction. ■

Corollary 3.4. *In the system with finite buffers, when the buffer length H is sufficiently large, the real received rate at any receiver is close to its token injection rate. By proposition 3.1, if K is large, then the token injection rate is close to the optimal solution.*

4. Extensions

(*a*) *Energy constraint and minimum arrival rate constraint*

We note that our approach can be extended to scenarios when there are other constraints on the network. The required modifications to handle additional constraints are not different from that in the case of unicast flows and so we only briefly comment on them below.

Consider a network where each node may have an average power constraint. Suppose that when node n transmits at an instantaneous rate $\xi_n[t]$ in time slot t, it consumes an amount of energy equal to $g_n(\xi_n[t])$ per time slot, where g_n depends upon the PHY layer characteristics. Suppose that the battery power of node n is constrained to be less that P_n, i.e. the long-term time average of $g_n(\xi)$ has to be less than or equal to P_n. The Lagrange multiplier for this constraint can be viewed as a fictitious queue: 'packets' arrive at this fictitious queue during each transmission from the node, with the number of arriving packets equal to the energy consumed during the transmission. Packets are drained from this queue at rate P_n. Thus, if the fictitious queue is stable, then it means that the average energy per unit time consumed by the node is less than or equal to the average power constraint. The idea of using fictitious queues to impose energy constraints is by Neely (2005) and Stolyar (2005, 2006). The fictitious energy queues should also be further incorporated into the back-pressure algorithm for resource allocation as specified in Stolyar (2005, 2006).

Another constraint that arises in applications such as video is a minimum arrival rate constraint: the flow rate x may be required to be greater than or equal to some minimum level $x_{\min}$. Then, we can introduce another fictitious queue for that flow. This fictitious queue has an arrival rate $x_{\min}$ and a service rate x. In other words, at each time slot, there is a constant arrival rate to this queue equal to $x_{\min}$; whenever x packets are sent by the flow, we remove the same number of x packets from this queue. These fictitious queues can then be stabilized by adding them to our framework.

(*b*) *Delay reduction using the shadow algorithm*

One can see that the shadow traffic acts like the capacity for the real traffic. In other words, the shadow algorithm creates 'pipes' for the real traffic. Since the capacity information is available, we can effectively reduce the delay of real traffic by letting each flow send traffic at a rate less than its available capacity as in the thinning algorithm in §3*b*. By increasing ϵ, one can decrease the rate of real traffic significantly below the available capacity, thus providing a trade-off between throughput and delay. This idea can also be used to reduce the delay for unicast traffic as well.

(c) Implementation issues

The disadvantage of the back-pressure algorithm is the implementation complexity. However, approximations of the back-pressure algorithm, by modifying the 802.11 protocol, have been proposed and implemented by Akyol *et al.* (in press).

5. Conclusion

We study the resource allocation problem for multicast sessions in multi-hop wireless networks. First, we extend the results from the existing theory for unicast flows to also consider the single-rate multicast sessions. However, such an extension is not straightforward in the case of multi-rate multicast. Therefore, we introduce the concept of shadow queues and propose a shadow algorithm that can achieve the optimal solution for multi-rate multicast. It turns out that our approach can also be used to exercise some degree of control over QoS (packet delays) delivered to the users of the network.

This research was supported by the DARPA CBMANET project, NSF grants CCF 06-34891, CNS 07-21286 and a Vodafone Fellowship.

References

Akyol, U., Andrews, M., Gupta, P., Hobby, J., Saniee, I. & Stolyar, A. In press. Joint scheduling and congestion control in mobile ad-hoc networks. In *Proc. IEEE INFOCOM.*

Deb, S. & Srikant, R. 2004 Congestion control for fair resource allocation in networks with multicast flows. *IEEE/ACM Trans. Netw.* **12**, 274–285. (doi:10.1109/TNET.2004.826293)

Eryilmaz, A. & Srikant, R. 2005 Fair resource allocation in wireless networks using queue-length based scheduling and congestion control. In *Proc. IEEE INFOCOM*, pp. 1794–1803. (doi:10.1109/INFCOM.2005.1498459)

Eryilmaz, A. & Srikant, R. 2006 Joint congestion control, routing and MAC for stability and fairness in wireless networks. *IEEE J. Select. Areas Commun.* **24**, 1514–1524. (doi:10.1109/JSAC.2006.879361)

Kar, K. & Tassiulas, L. 2006 Layered multicast rate control based on lagrangian relaxation and dynamic programming. *IEEE J. Select. Areas Commun.* **24**, 1464–1474. (doi:10.1109/JSAC.2006.879353)

Kar, K., Sarkar, S. & Tassiulas, L. 2002 A scalable low-overhead rate control algorithm for multirate multicast sessions. *IEEE J. Select. Areas Commun.* **24**, 1541–1557. (doi:10.1109/JSAC.2002.803988)

Lin, X. & Shroff, N. 2004 Joint rate control and scheduling in multihop wireless networks. In *Proc. IEEE Conf. on Decision and Control*, pp. 1484–1489.

Neely, M. J., Modiano, E. & Li, C.-P. 2005 Fairness and optimal stochastic control for heterogeneous networks. In *Proc. IEEE INFOCOM*, pp. 1723–1734. (doi:10.1109/INFCOM.2005.1498453)

Neely, M. J. 2005 Energy optimal control for time-varying wireless networks. In *Proc. IEEE INFOCOM*, pp. 572–583. (doi:10.1109/TIT.2006.876219)

Norris, J. R. 1998 *Markov chains.* Cambridge, UK: Cambridge University Press.

Srikant, R. 2004 *The mathematics of Internet congestion control.* Boston, MA: Birkhauser.

Stolyar, A. 2005 Maximizing queueing network utility subject to stability: greedy primal-dual algorithm. *Queueing Syst.* **50**, 401–457. (doi:10.1007/s11134-005-1450-0)

Stolyar, A. 2006 Greedy primal-dual algorithm for dynamic resource allocation in complex networks. *Queueing Syst.* **54**, 203–220. (doi:10.1007/s11134-006-0067-2)

Tassiulas, L. & Ephremides, A. 1992 Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Trans. Autom. Control.* **37**, 1936–1948. (doi:10.1109/9.182479)