

Algorithms and Stability Analysis for Universal-Swarming-Based Content Distribution

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

Abstract—We consider the issues of applying the universal swarming technique to transfer massive content efficiently. In a swarming session, a file is distributed to all the receivers by having all the nodes in the session exchange file chunks. By universal-swarming, not only all the nodes in the session, but also some nodes outside the session may participate in the chunk exchange to speed up the distribution process. We present a universal swarming model where the chunks are distributed along different Steiner trees rooted at the source and covering all the receivers. We assume chunks arrive dynamically at the sources and focus on finding stable universal swarming algorithms. To achieve the maximum throughput, universal swarming usually involves a tree-selection subproblem of finding a min-cost Steiner tree, which is NP-hard. We propose a universal swarming scheme that employs an approximate tree-selection algorithm. We show that it achieves network stability for a reduced throughput region, where the reduction ratio is the same as that of the tree-selection algorithm. We propose a second universal swarming scheme that employs a randomized tree-selection algorithm. It achieves the maximum throughput region, but with a weaker stability result.

I. INTRODUCTION

The Internet is being used to transfer content on a more and more massive scale. A recent innovation for efficient support of content distribution is a technique known as *swarming*, which was initially introduced in peer-to-peer (P2P) networking but is becoming increasingly attractive for infrastructure-based content distribution by service providers. In a swarming session, the file to be distributed is broken into many chunks at the original source, which are then spread out across the peers. Subsequently, the peers exchange the chunks with each other to speed up the distribution process. Many different ways of swarming have been proposed, such as BitTorrent [1], FastReplica [2], Bullet [3], [4], Chunkcast [5] and CoBlitz [6].

The problem addressed in this paper is how to conduct content distribution more efficiently by a class of improved swarming techniques, known as *universal swarming*. We associate with each file to be distributed a *session*, which consists of the source of a file and the receivers who are interested in downloading the file. In normal swarming, chunk exchange is restricted to the nodes of the session. It is well known that, for a swarming session that starts with a small number of nodes, as the size of the session increases, the distribution efficiency (e.g., completion time, average downloading rate) first increases quickly. The reason is that having more nodes means more opportunities that the nodes can help each other

to speed up the distribution process. However, as the session size reaches a threshold, further improvement in distribution efficiency due to the session size increase encounters sharply diminished return. This implies that, if we combine a small resource-poor session with a large resource-rich session, the distribution efficiency of the small session can improve greatly with negligible impact on the larger session.

Hence, in universal swarming, if we focus on a particular file, not only the source and all the receivers participate in the chunk exchange process, some other nodes who are not interested in the file may also participate. We call the latter out-of-session nodes. To illustrate the essence of universal swarming, as well as the main issues, consider the toy example in Fig. 1. The numbers associated with the links are their capacities. Let us consider a particular file whose source is node 1 and whose receivers are nodes 2 and 3. Node 4 is out of the session. Let us focus on a fixed chunk and consider how it can be distributed to the receivers. With some thoughts, it can be seen that the chunk propagates on a tree rooted at the source and covering both receivers. All possible distribution trees are shown in Fig. 2. We notice that a distribution tree may or may not include the out-of-session node, 4. Thus, a distribution tree is a *Steiner* tree rooted at the source covering all the receivers; the out-of-session nodes (e.g., node 4) are the Steiner nodes.

The example shows that universal swarming can be thought as distribution over multiple multicast trees, where each tree is a Steiner tree. With this model, one of the main questions is how to assign the chunks to different distribution trees so as to optimize certain performance objective, such as maximizing the sum of the utility functions of the sessions, or minimizing the time of the slowest session. This is a *rate allocation problem* on the multiple multicast trees. A similar question was first addressed in [7] in the context of non-universal swarming, where a single session is considered and the multicast trees are spanning trees instead of Steiner trees. For universal swarming, the question was addressed in [8].

This paper addresses the *stability problem*. The main question is: Given a set of data rates from the sources, which are possibly the solutions to the aforementioned rate allocation problem, how do we get a universal swarming algorithm so that the network queues will be stable? For the example in Fig. 1, a source rate of 2 is the largest distribution rate that can be supported by the network. When the file chunks arrive

at (or generated by) the source node 1 at a mean rate $2 - 2\epsilon$, where $\epsilon > 0$ is a small number, we can place chunks on the first and the second trees in Fig. 2 at a mean rate $1 - \epsilon$ each. For this example, the solution actually stabilizes the network. But, this conclusion requires technique conditions and is not generally true for more complicated situations.

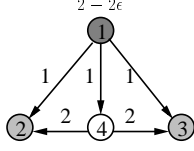


Fig. 1. Node 1 sends the file to node 2 and 3.

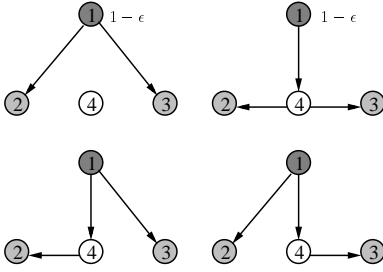


Fig. 2. All possible distribution trees for the example in Fig. 1.

Research on similar stability questions has been very active, but generally, in the context of unicast setting (e.g. [9]–[18]), possibly with multiple paths per connection. The presence of multicast puts our problem in a class of its own in that many earlier stable control algorithms, such as the maximum backpressure-based algorithm [9], [11], and techniques for stability analysis are not directly applicable. The main reason is that, unlike unicast, the flow conservation condition no longer holds under multicast.

Another salient aspect of the universal swarming problem is most related to the problem of link scheduling in wireless networks subject to link interference constraints, which has attracted much attention recently [10], [14], [16]–[30]. In [10], Tassiulas *et al.* showed that the maximum weighted link schedule achieves (i.e., stabilizes) the maximum throughput region. However, such a schedule is in general NP-hard. The universal swarming problem usually involves an NP-hard subproblem of finding a minimum-cost Steiner tree in order to achieve the maximum throughput. This similarity makes many of the concerns and investigative approaches in the wireless link scheduling problem relevant to the universal swarming problem. In [16], [23], Lin *et al.* showed that approximation algorithms for the maximum weighted scheduling problem can be used to stabilize the network for a reduced throughput region. Several researchers studied the maximal scheduling algorithms and how they impact the achievable throughput region [17], [18], [27]–[30]. Tassiulas [19] proposed a randomized scheduling algorithm that achieves the maximum throughput region.

In this paper, we develop a universal swarming scheme that employs an approximation algorithm to the tree selection (*a.k.a.* scheduling) problem, which achieves a reduced throughput region. We propose a second universal swarming scheme that utilizes a randomized tree selection algorithm, which achieves the maximum throughput region, but with a weaker stability property. The difference between our problem and the wireless scheduling problem is substantial. We must consider multi-hop, multicast communications, whereas most of the papers on the wireless problem are about unicast communication and many only focus on single-hop traffic. The difficulties with multi-hop communication arise from the fact that the arrival processes to the internal links are usually unknown.

The main results of the paper are summarized as follows.

- We develop a network signaling approach with source traffic regulation to solve the unknown-arrival problem in multi-hop communication. We show that using an approximate tree scheduling algorithm, together with signaling and source regulation, can achieve a reduced throughput region, and that the reduction ratio is equal to the approximation ratio of the tree scheduling algorithm.
- We show that using a randomized tree scheduling algorithm together with signaling can achieve the maximum throughput region, but with a weaker stability property.

The rest of the paper is organized as follows. The models and problem description are given in Section II. The first universal swarming scheme and analysis are presented in Section III. The second universal swarming scheme and analysis are presented in Section IV. The conclusion is in Section V.

II. PROBLEM DESCRIPTION

We consider a time-slotted system where each time slot is of one unit length. Let the network be represented by a directed graph $G = (V, E)$, where V is the set of nodes and E is the set of links. For each link $e \in E$, let c_e denote its capacity, where $c_e > 0$. For ease of presentation, we assume that each session, which distributes a distinct file, has one source, and hence, there is a one-to-one mapping between a session and a source. It is easy to extend what follows to the situation where a session may have multiple sources. Let S denote the set of sources (sessions); for each $s \in S$, let $V_s \subseteq V$ be the set of receivers associated with the source (session) S .

For each source $s \in S$, suppose unit-length packets (or file chunks) arrive at the source according to a random process, which will be distributed over the network to all the receivers, V_s . The motivation for using a source model with dynamic arrivals is that the content may not be a static file or stored locally. The model is general enough to cover realtime content, streaming video with time-varying bandwidth, or non-locally stored static data. Even if the entire file is static and stored at the source, this source model can still be appropriate. One can assume the arrival process is deterministic with a constant arrival rate. Let $A_s(k)$ be the number of packet arrivals on time slot k . Let us make the following assumption on the arrival

processes $\{A(k)\}$ throughout the paper, unless mentioned otherwise. Additional assumptions may be added as needed.

AS 1. For each source $s \in S$, $E[A_s(k)] = \lambda_s$, and $E[(A_s(k))^2] < K_1$ for some K_1 , for all time k . Furthermore, for every pair of sources s and s' , the covariance $\text{Cov}(A_s(k), A_{s'}(k)) < K_2$ for some K_2 , for all k .

Note that the second order statistics of $\{A(k)\}$ have uniform bounds. As discussed in Section I, for the static file case, λ_s could be the solution of an optimization problem.

In this paper, we will present stable universal swarming algorithms to distribute the packets to all the receivers. For each $s \in S$, the packets will be transmitted along various multicast distribution trees rooted at s to the receivers V_s . Both normal or universal swarming can be viewed as distributing the file of each session over multiple multicast trees. An analog to this in the unicast case is data delivery using multiple paths between the sender and receiver. The following distinguishes universal swarming from normal swarming: Each distribution tree in the former is a Steiner tree, where the out-of-session nodes on the tree are the Steiner nodes; in the latter, each distribution tree is a spanning tree consisting of only the source and the receivers.

We will take Neely's definition of stability ([14]; [31], chapter 2) unless mentioned otherwise. For a single-queue process $\{q(k)\}$, define the overflow function

$$g(M) = \limsup_{k \rightarrow \infty} \frac{1}{K} \sum_{k=1}^K P\{q(k) > M\}. \quad (1)$$

Definition 1. The single-queue process $\{q(k)\}$ is stable if $g(M) \rightarrow 0$ as $M \rightarrow \infty$. A network of queues is stable if every queue is stable.

With this definition of network stability, it generally suffices to show that a Lyapunov function of the queues has a negative drift when it becomes large enough. If with additional assumptions, the network queues form an ergodic Markov chain, the same drift condition implies the chain is positive recurrent, or equivalently, has a stationary distribution.

A. Throughput Region

For each source $s \in S$, let the set of candidate distribution trees be denoted by T_s . Let $T = \cup_{s \in S} T_s$. The trees can be enumerated in an arbitrary order as $t_1, t_2, \dots, t_{|T|}$, where $|\cdot|$ denote the cardinality of a set. Throughout this paper, for each $s \in S$, T_s contains all possible distribution trees rooted at the source s unless specified otherwise. Although $|T|$ is finite, it might be exponentially large in the number of links.

To have stability, the mean arrival rates λ cannot be arbitrarily large. The *maximum throughput region* is defined as

$$\Lambda = \{\lambda \geq 0 : \exists \alpha \geq 0 \text{ such that } \sum_{t \in T_s} \alpha_t = 1, \forall s \in S \\ \text{and } \sum_{s \in S} \sum_{\{t \in T_s | e \in t\}} \alpha_t \lambda_s \leq c_e, \forall e \in E\}. \quad (2)$$

Here, α represents how the traffic from the sources is split among the distribution trees. Obviously, Λ serves as an upper bound of the stability region, which consists of all λ that can be stabilized by some algorithm. This is so because, for any non-negative mean rate vector $\lambda \notin \Lambda$, no matter how the traffic is split among the distribution trees, there exists a link e such that the total arrival rate to e is strictly greater than its service rate. In Section III, we will show the interior of Λ is stabilizable.

We also define a γ -reduced throughput region as $\frac{1}{\gamma}\Lambda$, where $\gamma \geq 1$. By saying that the arrival rate vector λ is strictly inside the region $\frac{1}{\gamma}\Lambda$, we mean that there exist some $\epsilon_0 > 0$ and a vector $\alpha \geq 0$ such that $\sum_{t \in T_s} \alpha_t = 1, \forall s \in S$ and $\sum_{s \in S} \sum_{\{t \in T_s | e \in t\}} \alpha_t \lambda_s \leq \frac{1}{\gamma} c_e - \epsilon_0, \forall e \in E$. This is equivalent to

$$c_e \geq \gamma(\epsilon_0 + \sum_{s \in S} \sum_{\{t \in T_s | e \in t\}} \alpha_t \lambda_s), \quad \forall e \in E. \quad (3)$$

Note that when $\gamma = 1$, the γ -reduced throughput region $\frac{1}{\gamma}\Lambda$ is the maximum throughput region Λ .

B. The Class of Algorithms: Time Sharing of Trees

Each source has two possible approaches to use the multiple multicast trees. In one approach, the traffic from each source s may be split according to some weights $(\alpha_t)_{t \in T_s}$ and transmitted simultaneously over the trees on every time slot. Alternatively, the distribution can be done by time-sharing of the trees. The algorithms in this paper follow the time-sharing approach. On each time slot k , the source s selects one distribution tree from the set T_s , denoted by $t_s(k)$, according to some tree-scheduling (tree-selection) scheme, and transmits packets only to this tree on time slot k . The time-sharing approach can emulate the first approach in the sense that, when done properly, the fraction of time each distribution tree is used over a long period of time can approximate any weight vector $(\alpha_t)_{t \in T_s}$.

The questions are how to select the distribution tree $t_s(k)$ at each time slot, how many packets are released to the tree $t_s(k)$, whether the network is stable under the proposed algorithm, and how the tree-scheduling scheme impacts the stability region. We will study these questions and present two algorithms in the following sections.

III. SIGNALING, SOURCE TRAFFIC REGULATION AND γ -APPROXIMATED MIN-COST TREE SCHEDULING

A. Signaling Approach

Stability analysis of a multi-hop network is often difficult because the packets travel through the network hop-by-hop, instead of being imposed directly to all links that they will traverse. As a result, the arrival process to each internal link can rarely be described. The frequently-used technique of network signaling can be helpful. In our case, the source s sends control signals to inform all the links on the currently selected tree $t_s(k)$ the number of packets it wishes to transmit over this tree by the end of time slot k . We can imagine each signaling message carries that number of virtual packets with it. We give the signaling messages the highest processing

priority at each node/link, and hence, they experience the minimum delay. We can assume that each signaling message from a source arrives at each hop instantaneously.

Consider a particular time slot k and a particular internal link e on the selected distribution tree. The real packets issued by the source on time slot k will in general be delayed or buffered at upstream hops and will not arrive at link e until later. However, via signaling, link e knows how many packets are transmitted by the source on time slot k . The cumulative number of arrived real packets to each hop must be no more than the cumulative number of arrived virtual packets.

One question is how many virtual packets (real packets as well) are to be released to the network on a time slot. Intuitively, each source s can release all the packets arrived during time slot k , i.e., $A_s(k)$. However, the uncontrolled randomness of $A_s(k)$ causes a trouble in the stability analysis, as we will see later. In our signaling-based algorithm, each source s sets the number of virtual packets to be released at a constant value $\lambda_s + \epsilon_1$ on every time slot k . Here, ϵ_1 is a sufficiently small constant such that $0 < |S|\epsilon_1 < \epsilon_0$. This guarantees the stability of the source regulators, as we will see.

In the algorithm, each link e updates a virtual queue, denoted by $q_e(k)$.

$$q_e(k+1) = [q_e(k) + \sum_{s \in S: e \in t_s(k)} (\lambda_s + \epsilon_1) - c_e]_+. \quad (4)$$

$[\cdot]_+$ is the projection operation onto the non-negative domain. Tree scheduling is based on the virtual queues instead of the real queues.

B. Source Traffic Regulation

Since at each time slot, for any source s , the number of virtual packets signaled by the source is $\lambda_s + \epsilon_1$, the number of real packets transmitted should be no more than $\lambda_s + \epsilon_1$. A regulator is placed at each source s to guarantee this.

A regulator is a traffic shaping device and this technique is also considered in [17]. All the packets arriving at the source s first enter a regulator queue, and will be released to the network later in a controlled fashion. On each time slot k , let $D_s(k)$ denote the number of packets released from the regulator to the distribution tree $t_s(k)$, and let $p_s(k)$ be the regulator queue size at source s . The evolution of the regulator queue is given by

$$p_s(k+1) = p_s(k) + A_s(k) - D_s(k), \quad (5)$$

where

$$D_s(k) = \begin{cases} \lambda + \epsilon_1 & \text{if } p_s(k) \geq \lambda + \epsilon_1; \\ p_s(k) & \text{otherwise.} \end{cases} \quad (6)$$

From (5) and (6), we see that at most $\lambda_s + \epsilon_1$ real packets are released on each time slot, and this rate is slightly higher than the mean packets arrival rate, provided that the regulator has sufficient packets. This guarantees the stability of the regulator, and we will address this in more details in stability analysis.

C. γ -Approximated Min-Cost Tree Scheduling

We can interpret the virtual queue size q_e as the cost of link e . Then, the cost of a tree t is $\sum_{e \in t} q_e$. We propose the γ -approximated min-cost tree scheduling scheme, on each time slot k and for each source s , the selected tree $t_s(k)$ satisfies

$$\sum_{e \in t_s(k)} q_e(k) \leq \gamma \min_{t \in T_s} \sum_{e \in t} q_e(k), \quad (7)$$

where $\gamma \geq 1$. If there are multiple trees satisfying (7), the tie is broken randomly.

The reason that we propose γ -approximated min-cost tree scheduling is straightforward. When $\gamma = 1$, (7) becomes the minimum-cost Steiner tree problem, which is NP-hard. But, the min-cost Steiner tree problem has approximation solutions, which we can use. It will be proven in the following stability analysis that, if we are able to find the minimum-cost Steiner tree on each time slot, we can stabilize the network for the entire interior of the maximum throughput region, Λ ; if we adopt the γ -approximated min-cost tree scheduling, we can stabilize the network for the interior of $\frac{1}{\gamma}\Lambda$.

D. Stability Analysis

The analysis of stability is based on the drift analysis of Lyapunov functions.

1) *Stability of the Regulators:* Define a Lyapunov function of the regulator queues p as

$$L_1(p) = \sum_{s \in S} p_s^2. \quad (8)$$

Lemma 1. *There exists some positive constant M such that for every time slot k and the regulator backlog vector $p(k)$, the Lyapunov drift satisfies*

$$E[L_1(p(k+1)) - L_1(p(k)) | p(k)] \leq -\epsilon_1 \sum_{s \in S} p_s(k), \quad (9)$$

if $\sum_{s \in S} p_s(k) \geq \frac{M}{\epsilon_1}$.

Proof: This is because the mean arrival rate is strictly less than the mean service rate provided the regulator has sufficient packets. The proof is standard and we omit the details. ■

2) *Stability of the Virtual Queues:* Define a Lyapunov function of the virtual queue backlog vector q as

$$L_2(q) = \sum_{e \in E} q_e^2. \quad (10)$$

Let $t(k) = (t_s(k))_{s \in S}$ be the vector of the chosen distribution trees at time k . Note that $t(k)$ is a random vector because there might be multiple solutions to the tree-scheduling problem (7) and the tie is broken randomly.

Lemma 2. *If the mean arrival rate vector λ is strictly inside the region $\frac{1}{\gamma}\Lambda$, then, there exist some positive constants M and ϵ for all sample paths of $\{t(k)\}_k$ such that, for every time slot k and virtual queue backlog vector $q(k)$, the Lyapunov drift satisfies*

$$L_2(q(k+1)) - L_2(q(k)) \leq M - 2\epsilon \sum_{e \in E} q_e(k). \quad (11)$$

Proof: The proof is given in Appendix A. ■

Hence, when $\sum_{e \in E} q_e(k) \geq \frac{M}{\epsilon}$, the Lyapunov function has a negative drift under all sample paths of $\{t(k)\}_k$.

$$L_2(q(k+1)) - L_2(q(k)) \leq -\epsilon \sum_{e \in E} q_e(k). \quad (12)$$

Corollary 3. *For each link e , there exists a sufficiently large constant $M_e < \infty$ such that $q_e(k) \leq M_e$.*

Remark: The chosen deterministic arrival rates of the virtual packets guarantee that the virtual queues are bounded. This is an important fact for proving the stability of the real queues. If the sources signal random arrival rates for the virtual packets, the virtual queues can be stable but are not guaranteed to be bounded.

3) *Stability of the Real Queues:* For convenience, let us assume each real packet remembers its distribution tree. This way, the nodes on the tree know when to duplicate the packet. Moreover, each packet at any link also has an unambiguous *hop count*, which is the hop count on its tree path from the source to the current link. With this setup, we can assume the following queueing discipline for the real queues.

AS 2. *At each link e , a packet at a smaller number of hops away from its source will have priority over any packet at a larger number of hops away from its source.*

First we will show some properties of the real packet arrival rate to the intermediate links. Define an indicator function as

$$I(e, t) = \begin{cases} 1 & \text{if } e \in t; \\ 0 & \text{otherwise.} \end{cases}$$

Lemma 4. *For any link $e \in E$, there exists a constant $M_e > 0$ such that for any k_0 and k with $k_0 \leq k$,*

$$\sum_{u=k_0}^k \sum_{s \in S} D_s(u) I(e, t_s(u)) \leq (k - k_0 + 1)c_e + M_e. \quad (13)$$

Proof: See Appendix A. ■

Let $Q_e(k)$ denote the real queue backlog of link e at time slot k . We will show by induction that under the prioritized queueing strategy in AS 2, the real queue backlog is bounded. The proof is adapted from [16].

Theorem 5. *Under the additional assumption AS 2, if the mean arrival rate vector λ is strictly inside the region $\frac{1}{\gamma}\Lambda$, the real queue backlogs are bounded. I.e., there exists some constant $M' > 0$ such that*

$$Q_e(k) \leq M', \quad \forall k, \forall e \in E. \quad (14)$$

Proof: See Appendix A. ■

Theorem 6. *Under the additional assumption AS 2, if the mean arrival rate vector λ is strictly inside the region $\frac{1}{\gamma}\Lambda$, the γ -approximated min-cost tree scheduling scheme stabilizes the network. Furthermore, when $\gamma = 1$, if the mean arrival rate vector λ is strictly inside the maximum throughput region Λ , the min-cost tree scheduling scheme stabilizes the network.*

Proof: From Lemma 1, Lemma 2 (or Corollary 3) and Theorem 5, the regulator queues have negative drifts, the virtual queues and the real queues in the network are all bounded. ■

IV. SIGNALING WITHOUT SOURCE TRAFFIC REGULATION AND RANDOMIZED TREE SCHEDULING

Part of Theorem 6 states that the entire interior of the maximum throughput region Λ can be stabilized, provided one can solve the hard min-cost Steiner tree problem. Theorem 6 also says that the min-cost Steiner tree problem can be replaced with an approximation algorithm, if one is content to have a reduced stability region.

In this section, we will continue to cope with the hard min-cost Steiner tree problem. We will consider an algorithm that randomly samples the trees on each time slot. Selecting trees by random sampling is attractive in practice since the algorithms for doing this tend to be simple and fast. Some practical systems such as BitTorrent [1] already use variants of random sampling.

Our main concern is whether the tree-sampling approach can have any performance guarantee on the stability region. We conjecture it does. The idea is that there is no need to find the min-cost tree on every time slot. It can be sufficient that, as the algorithm goes on, it eventually selects the min-cost tree. In essence, the burden of finding the min-cost tree is amortized over the iteration steps of the algorithm. We will show important steps that may eventually lead to the conclusion that, in contrast to the case with approximation algorithms, the entire interior of Λ is stabilizable. The theoretical development about the algorithm is in part based on [19].

A. Signaling

In this algorithm, the sources still signal the links about the incoming traffic, but they are not regulated. Specifically, the number of virtual packets released by source s on every time slot k is $A_s(k)$ instead of $\lambda_s + \epsilon_1$. For each $e \in E$, the evolution of the virtual queue, $q_e(k)$, is

$$q_e(k+1) = [q_e(k) + \sum_{s \in S: e \in t_s(k)} A_s(k) - (c_e - \epsilon_2)]_+, \quad (15)$$

where $0 < \epsilon_2 < \epsilon_0$. From (15), the virtual queue dynamic is conservative since it does not use the full service capacity. We will later see the reason in stability analysis.

We eliminate the source regulators because, the algorithm would not have any benefit if the regulators are present. Moreover, without the regulators, the operations of the algorithm are much simpler.

B. Randomized Tree Scheduling

Denote the min-cost tree for source s by $\tau_s(q)$ with respect to the link cost vector q . The min-cost tree is expressed formally as

$$\sum_{e \in \tau_s(q)} q_e = \min_{t \in T_s} \sum_{e \in t} q_e. \quad (16)$$

If there are more than one min-cost trees, the tie is broken randomly.

In the randomized tree scheduling scheme, instead of choosing an approximately minimum-cost tree, the sources use some randomized algorithm to select trees, with the requirement that the algorithm can find the min-cost tree with a positive probability. More specifically, let $\hat{t}(k) = (\hat{t}_s(k))_{s \in S}$ be the set of trees selected by the randomized min-cost Steiner tree algorithm on time slot k . The tree set $\hat{t}(k)$ satisfies, for some $\delta > 0$,

$$P\left\{\sum_{e \in \hat{t}_s(k)} q_e(k) = \sum_{e \in \tau_s(k)} q_e(k), \forall s \in S\right\} \geq \delta. \quad (17)$$

We further require that the randomized scheduling scheme always chooses a tree with a lower cost than the previously scheduled tree, with respect to the current link cost vector. Recall that $t_s(k)$ is the scheduled tree at time k . We require that for any source $s \in S$,

$$t_s(k) = \begin{cases} \hat{t}_s(k) & \text{if } \sum_{e \in \hat{t}_s(k)} q_e(k) \leq \sum_{e \in t_s(k-1)} q_e(k); \\ t_s(k-1) & \text{otherwise.} \end{cases} \quad (18)$$

(18) ensures that the randomized tree scheduling scheme always moves toward the cost-reduction direction.

There are many possible randomized selection algorithms that satisfy (17) and (18). For instance, one algorithm might be to modify the current tree by randomly adding or deleting edges subject to the tree requirement. The selection of the edges can be biased toward lower-cost ones for addition and higher-cost ones for deletion. In this paper, we will not dwell on finding specific algorithms but will focus on the stability issue of the whole algorithm class.

C. Stability Analysis

We will show that, if the mean arrival rate vector λ strictly inside the maximum throughput region Λ , the randomized tree scheduling scheme is able to stabilize all the virtual queues; with additional assumptions, the cumulative arrival of the real packets by any time slot is strictly less than the cumulative link service rate for every link. Due to space limitation, we will state some of the results without proofs. They can be found in the extended version of this paper.

1) *Stability of the Virtual Queues:* The virtual queues $q(k)$ will be considered as the link costs. Let $t(k)$ be the vector of chosen trees. The analysis of stability is based on the drift analysis of an appropriate Lyapunov function of $x = (q, t)$. Define a Lyapunov function

$$L(x) = L_1(x) + L_2(x),$$

where

$$L_1(x) = \sum_{e \in E} q_e^2, \quad L_2(x) = \left(\sum_{s \in S} \lambda_s \left(\sum_{e \in t_s} q_e - \sum_{e \in \tau_s(q)} q_e \right) \right)^2.$$

The proofs for the following three main lemmas parallel the development in [19], although the details are different and technical. We omit them for brevity.

Lemma 7. *If the mean arrival rate vector λ is strictly inside the maximum throughput region Λ , there exist some positive constants M_1 and ϵ such that*

$$E[L_1(x(k+1)) - L_1(x(k)) | x(k)] \leq M_1 + 2\sqrt{L_2(x(k))} - 2\epsilon \sum_{e \in E} q_e(k). \quad (19)$$

Lemma 8. *If the arrival rate vector λ is strictly inside the maximum throughput region Λ , there exist some positive constants M_2 and M_3 such that*

$$E[L_2(x(k+1)) - L_2(x(k)) | x(k)] \leq M_2 + M_3\sqrt{L_2(x(k))} - \delta L_2(x(k)). \quad (20)$$

Lemma 9. *If the arrival rate vector λ is strictly inside the maximum throughput region Λ , there exist some positive constants M and ϵ such that, if $L(x(k)) \geq M$,*

$$E[L(x(k+1)) - L(x(k)) | x(k)] \leq -\epsilon\sqrt{L_1(x(k))}. \quad (21)$$

Theorem 10. *If the mean arrival rate vector λ is strictly inside the maximum throughput region Λ , the randomized tree scheduling scheme stabilizes the virtual queues.*

Proof: This is a corollary from Lemma 9. ■

2) *Stability of the Real Queues:* We have partial results about the stability of the real queues under additional conditions. We assume the following assumption in this subsection.

AS 3. *The processes $\{A_s(k)\}_k$ for different s are independent from each other. For each $s \in S$, $\{A_s(k)\}_k$ is IID. In addition, at every time slot k , there is a nonzero probability that no packet arrives at the sources, i.e., $P\{A_s(k) = 0, \forall s \in S\} > 0$.*

We will show that for any link e , its average traffic intensity (load), ρ_e , satisfies $\rho_e < 1$, where ρ_e is the ratio of the average packet arrival rate and service rate. First, stronger stability conclusions can be said about the virtual queues.

Theorem 11. *Suppose the mean arrival rates vector λ is strictly inside the maximum throughput region Λ , and assumptions AS 1 and AS 3 hold.*

- *The process $\{q(k), t(k)\}_{k=0}^{\infty}$ is an aperiodic and irreducible Markov chain with a stationary distribution. Moreover, let \hat{q} be the virtual queues under the stationary distribution. Then, $E[\hat{q}_e] < \infty$.*
- *The strong law of large numbers holds: For each initial condition, and for all $e \in E$,*

$$\lim_{k \rightarrow \infty} \frac{\sum_{u=0}^k q_e(u)}{k+1} = E[\hat{q}_e], \text{ almost surely.} \quad (22)$$

- *The mean ergodic theorem holds: For each initial condition, and for all $e \in E$,*

$$\lim_{k \rightarrow \infty} E[q_e(k)] \rightarrow E[\hat{q}_e]. \quad (23)$$

Proof: See Appendix B. ■

Theorem 12. For any link $e \in E$,

$$\limsup_{k \rightarrow \infty} \frac{1}{k+1} \sum_{u=0}^k \sum_{s \in S} A_s(u) I(e, t_s(u)) \leq c_e - \epsilon_2, \quad (24)$$

and

$$\limsup_{k \rightarrow \infty} E \left[\frac{1}{k+1} \sum_{u=0}^k \sum_{s \in S} A_s(u) I(e, t_s(u)) \right] \leq c_e - \epsilon_2. \quad (25)$$

Proof: See Appendix B. ■

Let $a_e(k)$ denote the number of packet arrivals at the real queue of link e on any time slot k .

Corollary 13. For any link $e \in E$, the average traffic intensity (or load) $\rho_e < 1$, where ρ_e is defined as

$$\rho_e = \limsup_{k \rightarrow \infty} \frac{\sum_{u=0}^k a_e(u)}{(k+1)c_e}.$$

Proof: For any time slot k , the number of cumulative arrivals at the real queue is no more than the number of cumulative arrivals at the virtual queue, i.e.,

$$\sum_{u=0}^k a_e(u) \leq \sum_{u=0}^k \sum_{s \in S} A_s(u) I(e, t_s(u)).$$

Then, $\rho_e < 1$ follows from Theorem 12. ■

Remark: From Corollary 13, the conservative service of the virtual queue (i.e., service rate is $c_e - \epsilon_2$) guarantees $\rho_e < 1$.

Under the randomized tree scheduling scheme, the virtual queues are stable and the real traffic intensity $\rho_e < 1$ for any link e . But, we do not know whether the real queues are stable or not [13]. We expect that in practice, they are almost always stable. We suspect that under more assumptions on the traffic arrival process and the queueing discipline for the real queues, the real queues can be proven to be stable. Currently, the question remains open.

V. CONCLUSIONS

We study the problem of how to schedule the distribution of the packets under the dynamic arrival scenario. In order to take advantage of the universal swarming technique, the packets are distributed along multiple multicast trees. To achieve the maximum throughput region, we encounter a min-cost Steiner tree problem, which is NP-hard. Multi-hop traffic is another difficulty for finding stable universal swarming algorithms. We propose a γ -approximated min-cost tree scheduling algorithm with network signaling and source regulators. It guarantees network stability in a reduced throughput region, where the reduction ratio is the same as the approximation ratio of the solution to the min-cost tree problem. We further develop a randomized tree scheduling scheme with network signaling. It achieves the maximum throughput region and stabilizes the virtual queues. Moreover, the average traffic intensity at each link is strictly less one. However, whether or not the real queues are stable remains an open question.

APPENDIX A PROOFS FOR SECTION III

Proof of Lemma 2: Take any sample path of the process $\{t(k)\}_k$. Under this sample path, we have the following.

$$\begin{aligned} & L_2(q(k+1)) - L_2(q(k)) \\ &= \sum_{e \in E} (q_e^2(k+1) - q_e^2(k)) \\ &= \sum_{e \in E} \left(([q_e(k) + \sum_{s \in S: e \in t_s(k)} (\lambda_s + \epsilon_1) - c_e]_+)^2 - q_e^2(k) \right) \\ &\leq \sum_{e \in E} \left((q_e(k) + \sum_{s \in S: e \in t_s(k)} (\lambda_s + \epsilon_1) - c_e)^2 - q_e^2(k) \right) \\ &= \sum_{e \in E} \left(2q_e(k) \left(\sum_{s \in S: e \in t_s(k)} (\lambda_s + \epsilon_1) - c_e \right) + c_e^2 \right. \\ &\quad \left. + \left(\sum_{s \in S: e \in t_s(k)} (\lambda_s + \epsilon_1) \right)^2 - 2c_e \sum_{s \in S: e \in t_s(k)} (\lambda_s + \epsilon_1) \right) \\ &\leq M + 2 \sum_{e \in E} q_e(k) \left(\sum_{s \in S: e \in t_s(k)} \lambda_s + |S| \epsilon_1 - c_e \right) \end{aligned}$$

The first inequality holds due to the non-expansion property of the projection operation. The second inequality holds because the second and third terms in the previous equality are bounded, and the fourth term is non-positive. Note that M does not depend on the sample path of $\{t(k)\}_k$.

Since λ is strictly in the reduced throughput region, by (3), we can continue with the following.

$$\begin{aligned} & L_2(q(k+1)) - L_2(q(k)) \\ &\leq M + 2 \sum_{e \in E} q_e(k) \left(\sum_{s \in S: e \in t_s(k)} \lambda_s + |S| \epsilon_1 - \gamma \epsilon_0 \right. \\ &\quad \left. - \gamma \sum_{s \in S} \sum_{t \in T_s: e \in t} \alpha_t \lambda_s \right) \\ &= M - 2(\gamma \epsilon_0 - |S| \epsilon_1) \sum_{e \in E} q_e(k) + 2 \sum_{s \in S} \lambda_s \left(\sum_{e \in t_s(k)} q_e(k) \right. \\ &\quad \left. - \sum_{t \in T_s} \alpha_t \left(\gamma \sum_{e \in t} q_e(k) \right) \right) \\ &\leq M - 2(\gamma \epsilon_0 - |S| \epsilon_1) \sum_{e \in E} q_e(k) \\ &= M - 2\epsilon \sum_{e \in E} q_e(k). \end{aligned}$$

The first equality holds by rearranging the terms and changing the order of summation. The second inequality holds because the third term in previous equality is non-positive due to the tree-selection algorithm (7). In the last equality, we define $\epsilon = \gamma \epsilon_0 - |S| \epsilon_1 > 0$. Note that $\gamma \epsilon_0 - |S| \epsilon_1 > 0$ since $\epsilon_0 > |S| \epsilon_1$ and $\gamma \geq 1$. ■

Proof of Lemma 4: According to (4),

$$q_e(k+1) \geq q_e(k) + \sum_{s \in S} (\lambda_s + \epsilon_1) I(e, t_s(k)) - c_e, \quad \forall e \in E. \quad (26)$$

From the evolution of the regulator backlog (6), we have

$$D_s(k) \leq \lambda_s + \epsilon_1, \quad \forall k. \quad (27)$$

Substituting (27) into (26) yields

$$q_e(k+1) \geq q_e(k) + \sum_{s \in S} D_s(k) I(e, t_s(k)) - c_e, \forall e \in E. \quad (28)$$

Summing the above inequality from time slots k_0 to k , we have

$$\begin{aligned} & \sum_{u=k_0}^k \sum_{s \in S} D_s(u) I(e, t_s(u)) \\ & \leq (k - k_0 + 1)c_e + q_e(k+1) - q_e(k_0) \\ & \leq (k - k_0 + 1)c_e + M_e. \end{aligned}$$

The last inequality is due to the non-negativity of $q(k)$ and Corollary 3. \blacksquare

Proof of Theorem 5: Recall that every packet at any link has a hop count from the source, which is the hop count on its tree path to the link. If a packet has a hop count h when it arrives at a link, we say it belongs to the h^{th} -hop traffic. Let $Q_e(k, h)$ denote the queue backlog at link e at time k contributed by all first-hop through h^{th} -hop traffic. We assume $Q_e(k, 0) = 0$ for ease of presentation.

For each $e \in E$, let $\bar{h}_e \leq |V| - 1$ be the largest hop count of any packet in e . Let $\bar{h} = \max_{e \in E} \bar{h}_e$. We claim that for all $h = 1, \dots, \bar{h}$,

$$Q_e(k, h) \leq M_h, \quad \forall k, \forall e \in E, \quad (29)$$

where the constants satisfy $M_1 \leq M_2 \leq \dots \leq M_{\bar{h}} < \infty$. We prove this claim by induction.

Base: When $h = 1$, note that $Q_e(k, 1) \leq q_e(k)$ and $q_e(k) \leq M_e$ by Corollary 3. Let $M_1 = \max_{e \in E} M_e$. Then, (29) holds for $h = 1$ and for all k .

Assume (29) holds for $1, \dots, h-1$ and for all k .

Induction on h : Let $x_e(k, h)$ denote the number of packets arriving at link e on time slot k that belong to the first-hop through h^{th} -hop traffic. Then during any interval of time, k_0 through k , the total number of arrivals is no more than the sum of the number of packets released by the sources during this interval that travel through link e and the backlogged first through $(h-1)^{\text{th}}$ -hop packets in the network at time k_0 , i.e.,

$$\begin{aligned} & \sum_{u=k_0}^k x_e(u, h) \\ & \leq \sum_{u=k_0}^k \sum_{s \in S} D_s(u) I(e, t_s(u)) + \sum_{e' \in E} Q_{e'}(k_0, h-1) \\ & \leq \sum_{u=k_0}^k \sum_{s \in S} D_s(u) I(e, t_s(u)) + |E| \cdot M_{h-1}. \end{aligned} \quad (30)$$

The second inequality holds due to the induction hypothesis.

By Loynes' formula, for all k ,

$$Q_e(k, h) = \max_{0 \leq k_0 \leq k} \left\{ \sum_{u=k_0}^k x_e(u, h) - c_e(k - k_0 + 1) \right\}$$

$$\begin{aligned} & \leq \max_{0 \leq k_0 \leq k} \left\{ \sum_{u=k_0}^k \sum_{s \in S} D_s(u) I(e, t_s(u)) \right. \\ & \quad \left. + |E| \cdot M_{h-1} - c_e(k - k_0 + 1) \right\} \\ & \leq |E| \cdot M_{h-1} + \max_{0 \leq k_0 \leq k} \{M_e\} \\ & = M_e + |E| \cdot M_{h-1}. \end{aligned} \quad (31)$$

The first inequality is because of (30). The second inequality is by Lemma 4. We can define $M_h = M_e + |E| \cdot M_{h-1}$.

Finally, note that the overall queue backlog $Q_e(k)$ at link e is equal to $Q_e(k, \bar{h}_e)$, where $\bar{h}_e \leq \bar{h}$. Hence, $Q_e(k) \leq M_{\bar{h}_e} \leq M_{\bar{h}}$ for all k . \blacksquare

APPENDIX B PROOFS FOR SECTION IV

Proof of Theorem 11: The proof follows from Theorem 8.0.3 in [32]. Denote by \mathcal{X} the state space of the Markov process $\{x(k)\} = \{q(k), t(k)\}$. Define a finite subset of the state space $\hat{\mathcal{X}} = \{x \in \mathcal{X} : \sum_{e \in E} q_e \leq \frac{M}{\epsilon}\}$, for M and ϵ as specified in Lemma 9. Note that by assumption AS 3, the process $\{x(k)\}$ is an x^* -irreducible Markov chain with $x^* = (q^*, t^*)$, where $q^* = \bar{0}$ and t^* is a fixed set of minimum-cost trees under the link cost vector $q^* = \bar{0}$;¹ and it is also aperiodic on the countable state space \mathcal{X} . Assumption AS 3 says that on every time slot, there is some nonzero probability that no new arrivals enter the sources. The system will empty after a finite number of such successive ‘‘no arrival’’ slots, an event that has a positive probability. This implies the process $\{x(k)\}$ is an x^* -irreducible Markov chain (i.e., $\sum_{k=0}^{\infty} P^k(x, x^*) > 0, x \in \hat{\mathcal{X}}$). Aperiodicity follows from $P(x^*, x^*) = P\{A(k) = 0\} \cdot P\{t^* \text{ is chosen}\} > 0$.

By Lemma 9, the chain $\{x(k)\}$ satisfies the Foster-Lyapunov drift conditions [32]. Now all the conditions of Theorem 8.0.3 in [32] are met. Hence, Theorem 11 holds. \blacksquare

Proof of Theorem 12: According to (15),

$$q_e(k+1) \geq q_e(k) + \sum_{s \in S} A_s(k) I(e, t_s(k)) - (c_e - \epsilon_2), \forall e \in E.$$

Summing the above inequality from time slots 0 to k , we have

$$\begin{aligned} & \sum_{u=0}^k \sum_{s \in S} A_s(u) I(e, t_s(u)) \\ & \leq (c_e - \epsilon_2)(k+1) + q_e(k+1) - q_e(0) \\ & \leq (c_e - \epsilon_2)(k+1) + q_e(k+1). \end{aligned} \quad (32)$$

Dividing both sides of (32) by $k+1$ yields

$$\frac{\sum_{u=0}^k \sum_{s \in S} A_s(u) I(e, t_s(u))}{k+1}$$

¹There might be multiple sets of the minimum-cost trees. Let t^* be an arbitrary one of them. Since the tie is broken uniformly at random, there is a non-zero probability that we choose the set t^* .

$$\begin{aligned}
&\leq c_e - \epsilon_2 + \frac{q_e(k+1)}{k+1} \\
&= c_e - \epsilon_2 + \frac{\sum_{u=0}^{k+1} q_e(u) - \sum_{u=0}^k q_e(u)}{k+1} \\
&= c_e - \epsilon_2 + \frac{\sum_{u=0}^{k+1} q_e(u)}{k+2} \cdot \frac{k+2}{k+1} - \frac{\sum_{u=0}^k q_e(u)}{k+1}.
\end{aligned}$$

Taking the limit on both sides of the above inequality as k goes to infinity yields,

$$\begin{aligned}
&\limsup_{k \rightarrow \infty} \frac{\sum_{u=0}^k \sum_{s \in S} A_s(u) I(e, t_s(u))}{k+1} \\
&\leq c_e - \epsilon_2 + \lim_{k \rightarrow \infty} \frac{\sum_{u=0}^{k+1} q_e(u)}{k+2} \cdot \lim_{k \rightarrow \infty} \frac{k+2}{k+1} \\
&\quad - \lim_{k \rightarrow \infty} \frac{\sum_{u=0}^k q_e(u)}{k+1} = c_e - \epsilon_2.
\end{aligned}$$

The last equality holds because $\lim_{k \rightarrow \infty} \frac{\sum_{u=0}^{k+1} q_e(u)}{k+2} = \lim_{k \rightarrow \infty} \frac{\sum_{u=0}^k q_e(u)}{k+1} = \bar{q}_e$ by Theorem 11. Hence (24) holds.

Now taking the expectation on the both sides of (32) yields

$$\begin{aligned}
E\left[\sum_{u=0}^k \sum_{s \in S} A_s(u) I(e, t_s(u))\right] &\leq (c_e - \epsilon_2)(k+1) + E[q_e(k+1)] \\
&\leq (c_e - \epsilon_2)(k+1) + M_e,
\end{aligned}$$

where $E[q_e(k+1)] \leq M_e < \infty$ is some positive finite number. Because Theorem 11 says, $\lim_{k \rightarrow \infty} E[q_e(k)] \rightarrow E[\hat{q}_e]$ and $E[\hat{q}_e] < \infty$, such M_e exists. Dividing both sides of the above inequality by $k+1$ and taking the limit, we have,

$$\begin{aligned}
&\limsup_{k \rightarrow \infty} E\left[\frac{1}{k+1} \sum_{u=0}^k \sum_{s \in S} A_s(u) I(e, t_s(u))\right] \\
&\leq c_e - \epsilon_2 + \lim_{k \rightarrow \infty} \frac{M_e}{k+1} = c_e - \epsilon_2.
\end{aligned}$$

REFERENCES

- [1] BitTorrent Website, <http://www.bittorrent.com/>.
- [2] J. Lee and G. de Veciana, "On application-level load balancing in FastReplica," *Computer Communications*, vol. 30, no. 17, pp. 3218–3231, November 2007.
- [3] D. Kostić, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: high bandwidth data dissemination using an overlay mesh," in *Proceedings of 19th ACM Symposium on Operating Systems Principles (SOSP '03)*, October 2003.
- [4] D. Kostić, R. Braud, C. Killian, E. Vandekieft, J. W. Anderson, A. C. Snoeren, and A. Vahdat, "Maintaining high bandwidth under dynamic network conditions," in *Proceedings of USENIX Annual Technical Conference*, 2005.
- [5] B.-G. Chun, P. Wu, H. Weatherspoon, and J. Kubiatowicz, "ChunkCast: An anycast service for large content distribution," in *Proceedings of the International Workshop on Peer-to-Peer Systems (IPTPS)*, February 2006.
- [6] K. Park and V. S. Pai, "Scale and performance in the CoBlitz large-file distribution service," in *Proceedings of the 3rd USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI)*, San Jose, CA, May 2006.
- [7] X. Zheng, C. Cho, and Y. Xia, "Optimal peer-to-peer technique for massive content distribution," in *Proceedings of INFOCOM*, 2008.
- [8] —, "Content distribution by multiple multicast trees and intersession cooperation: Optimal algorithms and approximations," in *Manuscript*, 2008.
- [9] B. Awerbuch and F. T. Leighton, "A simple local-control approximation algorithm for multicommodity flow," in *Proceedings of the IEEE Symposium on Theory of Computing*, 1993, pp. 459–468.
- [10] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Transactions on Automatic Control*, vol. 37, no. 12, December 1992.
- [11] L. Tassiulas, "Adaptive back-pressure congestion control based on local information," *IEEE Transactions on Automatic Control*, vol. 40, p. 236250, 1995.
- [12] A. Eryilmaz and R. Srikant, "Fair resource allocation in wireless networks using queue-length based scheduling and congestion control," in *Proceedings of the IEEE Infocom 2005*, Miami, USA, March 2005.
- [13] J. G. Dai, "On positive harris recurrence of multiclass queueing networks: A unified approach via fluid limit models," *Annals of Applied Probability*, vol. 5, pp. 49–77, 1995.
- [14] M. J. Neely, "Energy optimal control for time wireless networks," *IEEE Transactions on Information Theory*, vol. 52, no. 7, July 2006.
- [15] L. Georgiadis, M. J. Neely, and L. Tassiulas, "Resource allocation and cross-layer control in wireless networks," *Found. Trends Netw.*, vol. 1, no. 1, pp. 1–144, 2006.
- [16] X. Lin, N. B. Shroff, and R. Srikant, "The impact of imperfect scheduling on cross-layer rate control in wireless networks," *IEEE/ACM Transaction on Networking*, vol. 14, no. 2, pp. 302–315, April 2006.
- [17] X. Wu, R. Srikant, and J. R. Perkins, "Scheduling efficiency of distributed greedy scheduling algorithms in wireless networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 6, pp. 595–605, 2007.
- [18] C. Joo, X. Lin, and N. B. Shroff, "Understanding the capacity region of the greedy maximal scheduling algorithm in multi-hop wireless networks," in *Proceedings of INFOCOM*, April 2008.
- [19] L. Tassiulas, "Linear complexity algorithms for maximum throughput in radio networks and input queued switches," in *Proceedings of IEEE INFOCOM*, 1998.
- [20] P. Bjorklund, P. Varbrand, and D. Yuan, "Resource optimization of spatial TDMA in ad hoc radio networks: a column generation approach," in *Proceedings of INFOCOM*, 2003.
- [21] M. Johansson and L. Xiao, "Cross-layer optimization of wireless networks using nonlinear column generation," *IEEE Transaction on Wireless Communications*, vol. 5, no. 2, pp. 435–445, Feb. 2006.
- [22] S. Bohacek and P. Wang, "Toward tractable computation of the capacity of multihop wireless networks," in *Proceedings of INFOCOM*, 2007.
- [23] X. Lin, N. B. Shroff, and R. Srikant, "A tutorial on cross-layer optimization in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1452–1463, Aug. 2006.
- [24] L. Chen, S. H. Low, and J. C. Doyle, "Joint congestion control and media access control design for ad hoc wireless networks," in *Proceedings of the IEEE Infocom 2005*, Miami, USA, March 2005.
- [25] L. Chen, S. H. Low, M. Chiang, and J. C. Doyle, "Cross-layer congestion control, routing and scheduling design in ad hoc wireless networks," in *Proceedings of the IEEE Infocom 2006*, Barcelona, Spain, April 2006.
- [26] J. Yuan, Z. Li, W. Yu, and B. Li, "A cross-layer optimization framework for multicast in multi-hop wireless networks," in *Wireless Internet, 2005. Proceedings. First International Conference on*, July 2005, pp. 47–54.
- [27] A. Dimakis and J. Walrand, "Sufficient conditions for stability of longest-queue-first scheduling: second-order properties using fluid limits," *Advances in Applied Probability*, vol. 38, no. 2, 2006.
- [28] G. Sharma, N. B. Shroff, and R. R. Mazumdar, "Joint congestion control and distributed scheduling for throughput guarantees in wireless networks," in *Proceedings of INFOCOM*, May 2007.
- [29] A. Gupta, X. Lin, and R. Srikant, "Low-complexity distributed scheduling algorithms for wireless networks," in *Proceedings of INFOCOM*, 2007.
- [30] C. Joo, X. Lin, and N. B. Shroff, "Performance limits of greedy maximal matching in multi-hop wireless networks," in *Decision and Control, 2007 46th IEEE Conference on*, 2007.
- [31] M. J. Neely, "Dynamic power allocation and routing for satellite and wireless networks with time varying channels," Ph.D. dissertation, Massachusetts Institute of Technology, 2003.
- [32] S. Meyn, *Control Techniques For Complex Networks*. Cambridge University Press, 2008.