Algorithms and Stability Analysis for Content Distribution over Multiple Multicast Trees

Xiaoying Zheng, Chunglae Cho and Ye Xia

***Abstract*— The paper investigates theoretical issues in applying the universal swarming technique to efficient content distribution. 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 improve the distribution performance. 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 throughput region, 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 no more than the approximation ratio of the tree-selection algorithm. We propose a second universal swarming scheme that employs a randomized tree-selection algorithm. It achieves the 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 content distribution is a technique known as *swarming*. 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 FastReplica [1], Bullet [2], [3], Chunkcast [4], BitTorrent [5], and CoBlitz [6].

The swarming technique was originally introduced by the end-user communities for peer-to-peer (P2P) file sharing. The subject of this paper is how to apply swarming to infrastructure-based content distribution. Compared with the dynamic end-user file-sharing situation, the infrastructure networks and the content servers are relatively static (however, the traffic can still be dynamic). In this setting, it is beneficial to view swarming as distribution over multiple multicast trees. This view allows us to pose the question of how to optimally distribute the content. (See [7].) Furthermore, it is often easier to first develop sophisticated algorithms under the static assumption, and then, adapt them to practical situations. Hence, in this paper, swarming is synonymous to distribution over multiple multicast trees.

Xiaoying Zheng, Chunglae Cho and Ye Xia are with the Department of Computer and Information Science and Engineering, University of Florida, Gainesville, FL 32611. Email: {xiazheng, ccho, yx1}@cise.ufl.edu.

This paper concerns 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 traditional swarming, chunk exchange is restricted to the nodes of the session. However, in *universal swarming*, multiple sessions are combined into a single "super session" on a shared overlay network. Universal swarming takes advantages of the heterogenous resource capacities of different sessions, such as the source upload bandwidth, receiver download bandwidth, or aggregate upload bandwidth, and allows the sessions to share each other's resources. The result is that the distribution efficiency of the resource-poor sessions can improve greatly with negligible impact on the resource-rich sessions (see [8]).

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 for which the source is node 1 and the 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.

With this model of multiple multicast trees, 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 distribution time of the slowest session. This is a *rate allocation problem* on the multicast trees. One such question was addressed in [7] in the context of non-universal swarming, where each session's 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.

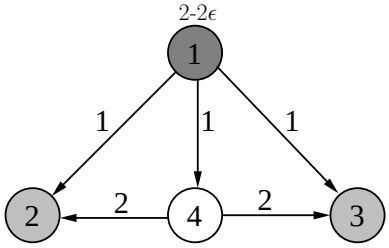

Fig. 1. Node 1 sends the file to nodes 2 and 3.

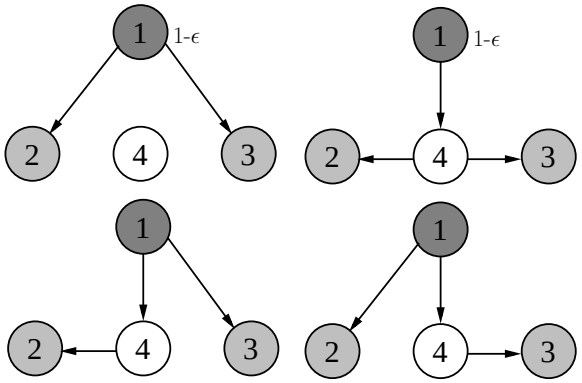

Fig. 2. All possible distribution trees for the example in Fig. 1.

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-weight link schedule achieves (i.e., stabilizes) the entire interior of the throughput region[1], where the weights are the queue size differences, or the backpressure. However, finding such a schedule is in general an NP-hard problem. The universal swarming problem usually involves an NP-hard subproblem in order to achieve the entire throughput region, which is to find a minimum-cost Steiner tree. 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-weight scheduling problem can be used to stabilize a portion of the throughput region. Some researchers considered maximal scheduling algorithms and studied what their stability regions are [17], [18], [27]–[30]. Tassiulas [19] proposed a randomized scheduling algorithm that achieves the entire 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 rate region equal to the throughput region reduced by a constant factor γ, $\gamma \geq 1$. We show that γ is no more than the approximation ratio of the tree scheduling algorithm. The scheme requires network signaling and source traffic regulation. We propose a second universal swarming scheme that utilizes a randomized tree selection algorithm, which achieves the entire 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 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 rest of the paper is organized as follows. The models and the problem description are given in Section II. The first universal swarming scheme and the analysis are presented in Section III. The second scheme and the analysis are presented in Section IV. The detailed proofs are omitted and can be found in [31]. The conclusion is in Section V.

II. Problem Description

We consider a time-slotted system where each time slot has a duration of one time unit. 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 but without loss of generality, 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. 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

[1]Subsequently, when we say an algorithm achieves or stabilizes a region, we mean the interior of the region.

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 $0 < K_1 < \infty$, 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 $0 < K_2 < \infty$, 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 a rate allocation 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. An analog to this in the unicast case is data delivery using multiple paths between the sender and the receiver.

We will take Neely's definition of stability ([14]; [32], chapter 2) unless mentioned otherwise. For a single-queue process $\{q(k)\}$, let us define the overflow function:

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

Definition 1: The single-queue process $\{q(k)\}$ is *stable* if $g(M) \to 0$ as $M \to \infty$. 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 $t_1, t_2, \cdots, t_{|T|}$, 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 very large.

The *throughput region* is defined as

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

Here, α represents how the traffic from the sources is split among the distribution trees. Obviously, Λ contains the stability region, i.e., all λ that can be stabilized by some algorithms. 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

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

Note that the region of rate vectors that are strictly inside the region $\frac{1}{\gamma}\Lambda$ contains the interior of $\frac{1}{\gamma}\Lambda$.

B. The Class of Algorithms: Time Sharing of Trees

Each source has at least two possible approaches to use the 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 over 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 $(\alpha_t)_{t\in T_s}$.

In addition to selecting the distribution tree $t_s(k)$ at each time slot, an algorithm also needs to decide how many packets are released to the tree. We will present two algorithms in the following sections. The key question is what portion of the rate region is stabilizable by each algorithm.

III. Signaling, Source Traffic Regulation and γ-approximation 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 be difficult to describe. The frequently-used technique of network signaling can be helpful. In our case, each 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 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 difficulty 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 (its use is also considered in [17]). All the packets arriving at 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

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

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

C. γ-Approximation 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-approximation 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 arbitrarily.

The rationale for this tree-scheduling scheme is straightforward. When $\gamma = 1$, the tree-scheduling scheme solves the min-cost Steiner tree problem, which is NP-hard. But, the min-cost Steiner tree problem has approximation solutions, which we can use. In [33], a family of approximation algorithms for the directed Steiner tree problem is proposed, which achieves an $O(\log^2 k)$ approximation ratio in quasi-polynomial time, where k is the number of receivers. 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 interior of the entire 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 $0 < M_1 < \infty$ 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 $\sum_{s\in S} p_s(k) \geq \frac{M_1}{\epsilon_1}$.

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. We allow $t(k)$ to be a random vector. For instance, this will be the case when there are multiple trees satisfying (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 $0 < M_2 < \infty$ 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

$$L_2(q(k+1)) - L_2(q(k)) \leq M_2 - 2\epsilon \sum_{e\in E} q_e(k), \quad (11)$$

where $\epsilon = \gamma\epsilon_0 - |S|\epsilon_1 > 0$.

Hence, when $\sum_{e\in E} q_e(k) \geq \frac{M_2}{\epsilon}$, 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 with a smaller hop count has priority over any packet with a larger hop count.

First, we will show some properties of the real packet arrival rates to the intermediate links. Define an indicator function $I(e,t)$, where e is a link and t is a tree.

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

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

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

Let $Q_e(k)$ denote the real queue backlog of link e at time slot k. We can show by induction that under the prioritized queueing strategy in AS 2, the real queue backlogs are 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 $0 < M' < \infty$ such that

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

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-approximation min-cost tree scheduling scheme stabilizes the network. Furthermore, when $\gamma = 1$, if the mean arrival rate vector λ is strictly inside the throughput region Λ, the min-cost tree scheduling scheme stabilizes the network.

IV. Randomized Tree Scheduling

Part of Theorem 6 states that the entire interior of the throughput region Λ can be stabilized, provided one can solve the hard min-cost Steiner tree problem. If approximation algorithms are used for the Steiner tree problem, Theorem 6 claims that a reduced rate region is stabilizable. In this section, we will continue to cope with the hard Steiner tree problem. Instead of approximation algorithms, we will consider an algorithm that randomly samples the trees at 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 [5] already use variants of random sampling.

Our main concern is whether the tree-sampling approach has any performance guarantee with respect to stability. We conjecture it does. 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 the stability analysis.

B. Randomized Tree Scheduling

Denote the min-cost tree for source s by $\tau_s(q)$ with respect to the link cost vector q. We have,

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

If multiple min-cost trees exist, the tie is broken arbitrarily.

In this scheme, the sources use some randomized algorithm to select trees, with the requirement that the algorithm can find the min-cost trees with a positive probability. More specifically, let $\hat{t}(k) = (\hat{t}_s(k))_{s\in S}$ be the trees selected by the randomized algorithm on time slot k. The following condition is satisfied for some $\delta > 0$,

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

We further require that the randomized tree-selection algorithm always chooses a tree with a lower cost than the previously selected 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 $s \in S$,

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

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 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 accumulation of the link service rate for every link.

1) Stability of the Virtual Queues: The virtual queue sizes $q(k)$ are considered as the link costs. Let $t(k)$ be the vector of chosen trees. Define a Lyapunov function of $x = (q, t)$:

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

where

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

The proof for the following lemma parallels the development in [19], although the details are different and technical.

Lemma 7: If the arrival rate vector λ is strictly inside the throughput region Λ, there exist some positive constants $M < \infty$ and ϵ such that, if $L(x(k)) \geq M$,

$$E[L(x(k+1)) - L(x(k))|x(k)] \leq -\epsilon \sqrt{L_1(x(k))}. \quad (19)$$

Theorem 8: If the mean arrival rate vector λ is strictly inside the throughput region Λ, the randomized tree scheduling scheme stabilizes the virtual queues.

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. At every time k, there is a nonzero probability that no packet arrives at the sources, i.e., $\mathrm{P}\{A_s(k) = 0, \forall s \in S\} > 0$.

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 link rate. First, stronger stability conclusions can be said about the virtual queues.

Theorem 9: Suppose the mean arrival rate vector λ is strictly inside the 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 $e \in E$,

$$\lim_{k \to \infty} \frac{\sum_{u=0}^{k} q_e(u)}{k+1} = E[\hat{q}_e], \text{ almost surely.} \quad (20)$$

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

$$\lim_{k \to \infty} E[q_e(k)] = E[\hat{q}_e]. \quad (21)$$

Theorem 10: For any link $e \in E$,

$$\limsup_{k \to \infty} \frac{1}{k+1} \sum_{u=0}^{k} \sum_{s \in S} A_s(u) I(e, t_s(u)) \leq c_e - \epsilon_2, \quad (22)$$

$$\limsup_{k \to \infty} E[\frac{1}{k+1} \sum_{u=0}^{k} \sum_{s \in S} A_s(u) I(e, t_s(u))] \leq c_e - \epsilon_2. \quad (23)$$

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

Corollary 11: For any link $e \in E$, the average traffic intensity (or load) $\rho_e < 1$, where ρ_e is defined as

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

Remark: The service rate of the virtual queue of link e, which 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.

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 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-approximation 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 no more than the approximation ratio of the algorithm for the min-cost tree problem. We further develop a randomized tree-scheduling algorithm with network signaling. It achieves the throughput region and stabilizes the virtual queues. Moreover, the average traffic intensity at each link is strictly less than one. However, whether or not the real queues are stable remains an open question.

References

[1] J. Lee and G. de Veciana, "On application-level load balancing in FastReplica," *Computer Communications*, vol. 30, no. 17, pp. 3218–3231, November 2007.

[2] D. Kostić, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: high bandwidth data dissemination using an overlay mesh," in *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP '03)*, October 2003, pp. 282–297.

[3] 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*, Anaheim, CA, USA, April 2005, pp. 14–14.

[4] B.-G. Chun, P. Wu, H. Weatherspoon, and J. Kubiatowicz, "ChunkCast: An anycast service for large content distribution," in *Proceedings of the Internaltional Workshop on Peer-to-Peer Systems (IPTPS)*, Santa Barbara, CA, USA, February 2006.

[5] BitTorrent Website, http://www.bittorrent.com/.

[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, pp. 3–3.

[7] X. Zheng, C. Cho, and Y. Xia, "Optimal peer-to-peer technique for massive content distribution," in *Proceedings of the IEEE INFOCOM*, Phoenix, AZ, USA, April 2008, pp. 151–155.

[8] ——, "Content distribution by multiple multicast trees and intersession cooperation: Optimal algorithms and approximations," to appear in *Proceedings of the 48th IEEE Conference on Decision and Control*, Shanghai, China, December 2009.

[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, pp. 1936–1948, December 1992.

[11] L. Tassiulas, "Adaptive back-pressure congestion control based on local information," *IEEE Transactions on Automatic Control*, vol. 40, pp. 236–250, 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*, Miami, USA, March 2005, pp. 1794–1803.

[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 varying wireless networks," *IEEE Transactions on Information Theory*, vol. 52, no. 7, pp. 2915–2934, 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 and N. B. Shroff, "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 the IEEE INFOCOM*, Phoenix, AZ, USA, April 2008, pp. 1103–1111.

[19] L. Tassiulas, "Linear complexity algorithms for maximum throughput in radio networks and input queued switches," in *Proceedings of the IEEE INFOCOM*, San Francisco, USA, March 1998, pp. 533–539.

[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 the IEEE INFOCOM*, San Francisco, USA, March 2003, pp. 818–824.

[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 the IEEE INFOCOM*, Anchorage, Alaska, USA, May 2007, pp. 2099–2107.

[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*, Miami, USA, March 2005, pp. 2212–2222.

[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*, Barcelona, Spain, April 2006, pp. 1–13.

[26] J. Yuan, Z. Li, W. Yu, and B. Li, "A cross-layer optimization framework for multicast in multi-hop wireless networks," in *Proceedings of the 1st International Conference on Wireless Internet*, Budapest, Hungary, 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 Probabilitys*, vol. 38, no. 2, pp. 505–521, 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 the IEEE INFOCOM*, Anchorage, Alaska, USA, May 2007, pp. 2072–2080.

[29] A. Gupta, X. Lin, and R. Srikant, "Low-complexity distributed scheduling algorithms for wireless networks," in *Proceedings of the IEEE INFOCOM*, Anchorage, Alaska, USA, May 2007, pp. 1631–1639.

[30] C. Joo, X. Lin, and N. B. Shroff, "Performance limits of greedy maximal matching in multi-hop wireless networks," in *Proceedings of the 46th IEEE Conference on Decision and Control*, New Orleans, Louisiana, USA, December 2007, pp. 1128–1133.

[31] X. Zheng, C. Cho, and Y. Xia, "Algorithms and stability analysis for content distribution over multiple multicast trees," in *Manuscript*, 2009, http://www.cise.ufl.edu/~yx1/publication.html.

[32] 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.

[33] M. Charikar and C. Chekuri, "Approximation algorithms for directed Steiner problems," *J. Algorithms*, vol. 33, no. 1, pp. 73–91, 1999.