Wireless Peer-to-Peer Scheduling in Mobile Networks

Michael J. Neely
University of Southern California
http://www-bcf.usc.edu/~mjneely

Abstract— **This paper considers peer-to-peer scheduling for a network with multiple wireless devices. A subset of the devices are mobile users that desire specific files. Each user may already have certain popular files in its cache. The remaining devices are access points that typically have access to a larger set of files. Users can download packets of their requested file from an access point or from a nearby user. Our prior work optimizes peer scheduling in a general setting, but the resulting delay can be large when applied to mobile networks. This paper focuses on the mobile case, and develops a new algorithm that reduces delay by opportunistically grabbing packets from current neighbors. However, it treats a simpler model where each user desires a single file with infinite length. An algorithm that provably optimizes throughput utility while incentivizing participation is developed for this case. The algorithm extends as a simple heuristic in more general cases with finite file sizes and random active and idle periods.**

I. Introduction

Consider a network with N wireless devices. Let K of these devices be identified as *users*, where $K \leq N$. The users can send packets to each other via direct peer-to-peer transmissions. Each user has a certain collection of popular files in its cache. The remaining $N - K$ devices are *access points*. The access points are connected to a larger network, such as the internet, and hence typically have access to a larger set of files. While a general network may have users that desire to upload packets to the access points, this paper focuses only on the user downloads. Thus, throughout this paper it is assumed that the access points only send packets to the users, but do not receive packets. In contrast, the users can both send and receive. The network is mobile, and so the transmission options between access points and users, and between user pairs, can change over time.

Each user only wishes to download, and does not naturally want to send any data. Users will only send data to each other if they agree to operate according to a control algorithm that schedules such transmissions. This paper assumes the users have already agreed to abide by the control algorithm, and thus focuses attention on altruistic network design that optimizes a global network utility function. Nevertheless, this paper includes a tit-for-tat constraint, similar to the work in [1], which is an effective mechanism for incentivizing participation. Further, while this paper develops a single algorithm for optimizing the entire network, this does not automatically require the algorithm to be centralized. Indeed, the resulting algorithm often has a distributed implementation.

This material is supported in part by one or more of: the NSF Career grant CCF-0747525, the Network Science Collaborative Technology Alliance sponsored by the U.S. Army Research Laboratory W911NF-09-2-0053.

The model of this paper applies to a variety of practical network situations. For example, the access points can be wireless base stations in a future cellular network that allows both base-station-to-user transmissions as well as direct user-to-user transmissions. Alternatively, some of the access points can be smaller femtocell nodes. To increase the capacity of wireless systems, it is essential for future networks to enable such femtocell access and/or direct user-to-user transmission. This paper considers only 1-hop communication, so that all downloads are received either directly from an access point or from another user. The possibility of a user acting as a multi-hop relay is not considered here.

Our prior work [1] treats a more complex model where each user can actively download multiple files at the same time, and where the arrival process of desired files at each user is random. A key challenge in this case is the complexity explosion associated with labeling each file according to the subset of other devices that already have it. This is solved in [1] by first observing the subset information of each newly arriving file, making an immediate decision about which device in this subset should transmit the desired packets, and then placing this request in a *request queue* at that selected device. The devices are not required to transmit these packets immediately. Rather, they can satisfy the requests over time. This procedure does not sacrifice optimality, and yields an algorithm with polynomial complexity. However, while this is effective for networks with static topology, it can result in significant delays in mobile networks. That is because a device that is pre-selected for transmission may not currently be in close proximity to the intended user, and/or may move out of transmission range before transmission occurs.

The current paper provides an alternative algorithm that reduces delay, particularly in the mobile case. To do so, we use a simpler model that assumes each user desires only one file that has infinite size. This enables us to focus on scheduling to achieve optimally fair download rates for each user. Rather than pre-selecting devices for eventual transmission, our algorithm makes opportunistic packet transmission decisions from the set of current neighbors that have the desired file. This also facilitates distributed implementation. The infinite file size assumption is an approximation that is reasonable when file sizes are large, such as for video files. A heuristic extension to the case of finite file sizes is treated in Section V. This heuristic is based on the optimality insights obtained from the infinite size case.

Prior work on fair scheduling in mobile ad-hoc networks has considered token-based and economics-based mechanisms to incentivize participation [2][3][4]. Incentives are also well studied in the peer-to-peer literature. For example, algorithms in [5][6][7][8] track the number of uploads and downloads for each user, and give preferential treatment to those who have helped others. Algorithms based on tokens, markets, and peer reputations are considered in [9][10][11]. Such algorithms are conceptually based on the simple "tit-for-tat" or "treat-for-treat" principle, where rewards are given in direct proportion to the amount of self-sacrificial behavior at each user. Our current paper also considers a tit-for-tat mechanism. However, a key difference is that it designs a *tit-for-tat constraint* directly into the optimization problem (similar to our prior work [1] that used a different network model). Remarkably, the solution of the optimization naturally results in an intuitive token-like procedure, where the number of tokens in a virtual queue at each user determines the *reputation* of the user. Peer-to-peer transmissions between user pairs are given preferential treatment according to the *differential reputation* between the users. This is similar to the *backpressure* principle for optimal network scheduling [12][13]. However, the "backpressure" in the present paper is determined by token differentials in virtual reputation queues, rather than congestion differentials.

II. Basic Model

Let $\mathcal{N}$ represent the set of devices, and $\mathcal{K}$ represent the set of users, where $\mathcal{K} \subseteq \mathcal{N}$. Let N and K be the sizes of these sets. For simplicity in this basic model, assume that each user $k \in \mathcal{K}$ wants a single file that consists of an infinite number of fixed-length packets (this assumption is modified to treat finite file sizes in Section V). For each $k \in \mathcal{K}$, define $\mathcal{F}_k$ as the subset of devices in $\mathcal{N}$ that have the file desired by user k. The set $\mathcal{F}_k$ can include both users and access points, and represents the set of devices that user k can potentially receive packets from as it moves throughout the network. If each user k only accepts downloads from a certain subset of devices that it identifies as its *social group*, then $\mathcal{F}_k$ can be viewed as the intersection of its social group and the set of devices that have its file. We assume that $k \notin \mathcal{F}_k$, so that no user wants a file that it already has. Further, we assume $\mathcal{F}_k$ is non-empty, so that at least one device has the desired file. This latter assumption is reasonable if the network has one or more access points with high speed internet connections. Such access points can be modeled as having all desired files.

The network operates in slotted time with time slots $t \in \{0, 1, 2, \ldots\}$. Every slot t, the network makes *transmission actions*. Let $(\mu_{nk}(t))$ be the matrix of transmission actions chosen on slot t, where each entry $\mu_{nk}(t)$ is the number of packets that device $n \in \mathcal{N}$ transmits to user $k \in \mathcal{K}$. The set of all possible matrix options to choose from is determined by the current *topology state* of the network, as in [13]. Specifically, let $\omega(t)$ represent the topology state on slot t, being a vector of parameters that affect transmission, such as current device locations and/or channel conditions. Assume $\omega(t)$ takes values in an abstract set Ω, possibly being an infinite set. The $\omega(t)$ process is assumed to be ergodic. In the case when Ω is finite or countably infinite, the steady state probabilities are represented by $\pi(\omega) = Pr[\omega(t) = \omega]$ for all $\omega \in \Omega$. Else, the steady state probabilities are represented by an appropriate probability density function. These probabilities are not necessarily known by the network controller. Extensions to the case when $\omega(t)$ is non-ergodic are explored in Sections V and VI.

For each $\omega \in \Omega$, define $\mathcal{R}(\omega)$ as the set of all transmission matrices (μ_{nk}) that are possible when $\omega(t) = \omega$. The exact structure of $\mathcal{R}(\omega)$ depends on the particular physical characteristics and interference properties of the network. One may choose the $\mathcal{R}(\omega)$ sets to constrain the transmission variables μ_{nk} to take integer values, although this is not required in our analysis. We assume only that each set $\mathcal{R}(\omega)$ has the following basic properties:

- Every matrix $(\mu_{nk}) \in \mathcal{R}(\omega)$ has non-negative entries.
- If $(\mu_{nk}) \in \mathcal{R}(\omega)$, then $(\tilde{\mu}_{nk}) \in \mathcal{R}(\omega)$, where $(\tilde{\mu}_{nk})$ is any matrix formed by setting one or more entries of (μ_{nk}) to zero.
- Every matrix $(\mu_{nk}) \in \mathcal{R}(\omega)$ must satisfy the constraint $0 \leq \sum_{n\in\mathcal{N}} \mu_{nk} \leq x_k^{max}$ for all $k \in \mathcal{K}$, where x_k^{max} is a given bound on the number of packets that can be delivered to user k on one slot, regardless of ω.

A. Example Network Structure

This section presents an example network model that fits into the above framework. The network region is divided into C non-overlapping subcells. Let $c_n(t)$ be the current subcell of device n, so that $c_n(t) \in \{1, \ldots, C\}$. Let $\boldsymbol{c}(t) = (c_1(t), \ldots, c_N(t))$ be the vector of device locations on slot t. The access points have fixed locations, while the mobile users can change subcells from slot to slot. Let $\boldsymbol{S}(t) = (S_{nk}(t))$ be a channel state matrix, where $S_{nk}(t)$ is the number of packets that device n can transmit to device k on slot t, provided that there are no competing transmissions (as defined below). The value $S_{nk}(t)$ can depend on the $\boldsymbol{c}(t)$ location vector. Let the topology state $\omega(t)$ be given by $\omega(t) = (\boldsymbol{c}(t), \boldsymbol{S}(t))$. Assume access point transmissions are orthogonal from all other access point transmissions and from all peer-to-peer user transmissions. For each $\omega(t)$, define $\mathcal{R}(\omega(t))$ as the set of all $(\mu_{nk}(t))$ matrices with entries that satisfy:

- $\mu_{nk}(t) \in \{0, S_{nk}(t)\}$ for all $n \in \mathcal{N}$ and $k \in \mathcal{K}$.
- Users can only transmit to other users currently in their same subcell, so that $\mu_{nk}(t) = 0$ whenever $n \in \mathcal{K}$, $k \in \mathcal{K}$, and $c_n(t) \neq c_k(t)$.
- At most one user-to-user transmission can take place per subcell on a given slot.
- Each access point can send to at most one user per slot.

This particular structure is useful because it allows user transmissions to be separately scheduled in each subcell, and access point transmissions to be scheduled separately from all other decisions. The sets $\mathcal{R}(\omega(t))$ can be defined differently for more sophisticated interference models. For example, the transmission rate of an access point can depend on whether or not neighboring access points are scheduled for transmission.

B. Optimization Objective

For each $a \in \mathcal{N}$ and $b \in \mathcal{K}$, define $f_{ab}(t)$ to be 1 if, on slot t, device a has the file requested by user b, and 0 otherwise:

$$f_{ab}(t) \overset{\triangle}{=} \begin{cases} 1 & \text{if } a \in \mathcal{F}_b \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Because each user desires a single infinite size file, the $f_{ab}(t)$ values do not change with time. However, we use the "(t)" notation because it facilitates the extension to more general cases in Section V, where the sets $\mathcal{F}_b$ in the right-hand-side of (1) are extended to $\mathcal{F}_b(t)$. Every slot t, the network controller observes $\omega(t)$ and chooses $(\mu_{nk}(t)) \in \mathcal{R}(\omega(t))$. For each $k \in \mathcal{K}$, define $x_k(t)$ as the total number of packets that user k receives from others on slot t, and define $y_k(t)$ as the total number of packets that user k delivers to others on slot t:

$$x_k(t) \triangleq \textstyle\sum_{a\in\mathcal{N}} \mu_{ak}(t) f_{ak}(t) \quad (2)$$

$$y_k(t) \triangleq \textstyle\sum_{b\in\mathcal{K}} \mu_{kb}(t) f_{kb}(t) \quad (3)$$

The multiplication $\mu_{ab}(t)f_{ab}(t)$ in (2) and (3) formally ensures that user b can only receive a packet from another device that has the file it is requesting.

For a given control algorithm, let $\overline{x}_k$ and $\overline{y}_k$ represent the time averages of the $x_k(t)$ and $y_k(t)$ processes for all $k \in \mathcal{K}$:

$$\overline{x}_k = \lim_{t\rightarrow\infty} \tfrac{1}{t}\textstyle\sum_{\tau=0}^{t-1} x_k(\tau) \ , \ \overline{y}_k = \lim_{t\rightarrow\infty} \tfrac{1}{t}\textstyle\sum_{\tau=0}^{t-1} y_k(\tau)$$

These limits are temporarily assumed to exist.[1] The value $\overline{x}_k$ is the time average download rate of user k, and $\overline{y}_k$ is the time average upload rate. The goal is to develop a control algorithm that solves the following:[2]

$$\text{Maximize:} \quad \textstyle\sum_{k\in\mathcal{K}} \phi_k(\overline{x}_k) \quad (4)$$

$$\text{Subject to:} \quad \alpha_k\overline{x}_k \leq \beta_k + \overline{y}_k \ \ \forall k \in \mathcal{K} \quad (5)$$

$$(\mu_{nk}(t)) \in \mathcal{R}(\omega(t)) \ \ \forall t \in \{0,1,2,\ldots\} \quad (6)$$

where for each $k \in \mathcal{K}$, $\phi_k(x)$ are given concave functions and α_k, β_k are given non-negative weights. The value $\phi_k(\overline{x}_k)$ represents the *utility* associated with user k downloading at rate $\overline{x}_k$. The constraints (5) are the *tit-for-tat* constraints from [1]. These constraints incentivize participation. They allow a "free" download rate of β_k/α_k. Users can only receive rates beyond this value in proportion to the rate at which they help others. Choosing larger values of α_k (typically in the range $0 \leq \alpha_k \leq 1$) leads to more stringent requirements about helping others. These tit-for-tat constraints restrict the system operation and thus can affect overall network utility. Removing these constraints by setting $\alpha_k = 0$ for all k leads to the largest network utility, but does not embed any participation incentives into the optimization problem.

The functions $\phi_k(x)$ are assumed to be concave, continuous, and non-decreasing over the interval $x \geq 0$. They are not required to be differentiable. For example, they can be piecewise linear, such as $\phi_k(x) = \min[x, \theta_k]$, where θ_k is a given constant rate desired by user k. Alternatively, one can choose $\phi_k(x) = \ln(x)$ for each $k \in \mathcal{K}$, which leads to the well known *proportional fairness utility* [14].

One may want to modify the problem (4)-(6) by specifying separate utility functions for the user-to-user download rates and the access-point-to-user download rates. This is possible by creating two "virtual users" $m_1(k)$ and $m_2(k)$ for each actual user $k \in \mathcal{K}$. Channel conditions for the virtual users $m_1(k), m_2(k)$ are defined to be the same as for the actual user k, with the exception that virtual user $m_1(k)$ is restricted to receive only from other users, while virtual user $m_2(k)$ is restricted to receive only from the access points.

[1]This is only to simplify exposition of the optimization goal. The analysis in later sections does not a-priori assume the limits exist.

[2]Note that the trivial all-zero solution is always feasible.

III. The Dynamic Algorithm

The problem (4)-(6) is solved via the stochastic network optimization theory of [13][15]. First note that problem (4)-(6) is equivalent to the following problem that uses *auxiliary variables* $\gamma_k(t)$:

$$\text{Maximize:} \quad \textstyle\sum_{k\in\mathcal{K}} \overline{\phi_k(\gamma_k)} \quad (7)$$

$$\text{Subject to:} \quad \alpha_k\overline{x}_k \leq \beta_k + \overline{y}_k \ \ \forall k \in \mathcal{K} \quad (8)$$

$$\overline{\gamma}_k \leq \overline{x}_k \ \ \forall k \in \mathcal{K} \quad (9)$$

$$(\mu_{nk}(t)) \in \mathcal{R}(\omega(t)) \ \ \forall t \in \{0,1,2,\ldots\} \quad (10)$$

$$0 \leq \gamma_k(t) \leq x_k^{max} \ \ \forall t \in \{0,1,2,\ldots\} \quad (11)$$

where $\overline{\phi_k(\gamma_k)}$ is defined as the time average of the process $\phi_k(\gamma_k(t))$. The auxiliary variables $\gamma_k(t)$ act as proxies for the actual download variables $x_k(t)$. This is useful for separating the nonlinear utility optimization from the network transmission decisions.

It can be shown that the optimal utility value is the same for both problems (4)-(6) and (7)-(11) [15]. Let ϕ^* denote this optimal utility. Now consider an algorithm that solves the problem (7)-(11). Let $(\mu_{nk}(t)), (\gamma_k(t))$ be the decisions made over time, and let $\overline{x}_k, \overline{y}_k, \overline{\gamma}_k, \overline{\phi_k(\gamma_k)}$ be the corresponding time averages, all of which satisfy the constraints of the problem (7)-(11). Note that these constraints include all of the desired constraints of the original problem (4)-(6). Then:

$$\phi^* = \textstyle\sum_{k\in\mathcal{K}} \overline{\phi_k(\gamma_k)} \ \leq \ \displaystyle\sum_{k\in\mathcal{K}} \phi_k(\overline{\gamma}_k) \quad (12)$$

$$\leq \ \textstyle\sum_{k\in\mathcal{K}} \phi_k(\overline{x}_k) \quad (13)$$

$$\leq \ \phi^* \quad (14)$$

where (12) holds because this algorithm achieves the optimal utility ϕ^* (together with Jensen's inequality for the concave function $\phi_k(\gamma)$), (13) holds because this algorithm must yield time averages that satisfy $\overline{\gamma}_k \leq \overline{x}_k$ for all $k \in \mathcal{K}$, and (14) holds because the transmission decisions of the algorithm satisfy all desired constraints of the original problem, and thus produce $\overline{x}_k$ values that give a utility that is less than or equal to the optimal utility of the original problem (which is also ϕ^*). It follows that any algorithm that is optimal for (7)-(11) makes decisions that are also optimal for the original problem.

A. Virtual Queues

To facilitate satisfaction of the tit-for-tat constraints (8), for each $k \in \mathcal{K}$ define a *virtual queue* $H_k(t)$, with dynamics:

$$H_k(t+1) = \max\left[H_k(t) + \alpha_k x_k(t) - \beta_k - y_k(t), 0\right] \quad (15)$$

where $x_k(t)$, $y_k(t)$ are defined in (2)-(3). The intuition is that $\alpha_k x_k(t)$ can be viewed as the "arrivals" on slot t, and $\beta_k + y_k(t)$ can be viewed as the "offered service" on slot t. Stabilizing queue $H_k(t)$ ensures the time average of the "arrivals" is less than or equal to the time average of the "service," which ensures constraints (8).

Similarly, to satisfy the constraints (9), for each $k \in \mathcal{K}$ define another virtual queue $Q_k(t)$ with dynamics:

$$Q_k(t+1) = \max[Q_k(t) + \gamma_k(t) - x_k(t), 0] \quad (16)$$

The update (16) can be interpreted as a queueing equation where $\gamma_k(t)$ is the amount of data requested by user k on slot t, and $x_k(t)$ is the amount of service. Stabilizing $Q_k(t)$ ensures $\overline{\gamma}_k \leq \overline{x}_k$.

B. The Drift-Plus-Penalty Algorithm

Define the following quadratic function $L(t)$:

$$L(t) \triangleq \frac{1}{2}\sum_{k\in\mathcal{K}}[Q_k(t)^2 + H_k(t)^2]$$

Intuitively, taking actions to push $L(t)$ down tends to maintain stability of all queues. Define $\Delta(t)$ as the drift on slot t:

$$\Delta(t) \triangleq L(t+1) - L(t)$$

Let $\boldsymbol{\Theta}(t) = (Q_k(t), H_k(t))|_{k\in\mathcal{K}}$ be the vector of all virtual queue values on slot t. The algorithm is designed to observe the queues and the current $\omega(t)$ on each slot t, and to then choose $(\mu_{nk}(t)) \in \mathcal{R}(\omega(t))$ and $\gamma_k(t)$ subject to $0 \leq \gamma_k(t) \leq x_k^{max}$ to minimize a bound on the following *drift-plus-penalty expression* [15]:

$$\Delta(t) - V\sum_{k\in\mathcal{K}}\phi_k(\gamma_k(t))$$

where V is a non-negative weight that affects a performance bound. Intuitively, the value of V affects the extent to which our control action on slot t emphasizes utility optimization in comparison to drift minimization.

Lemma 1: Under any control algorithm, we have:

$$\begin{aligned}\Delta(t) - V\sum_{k\in\mathcal{K}}\phi_k(\gamma_k(t)) \leq B(t) - V\sum_{k\in\mathcal{K}}\phi_k(\gamma_k(t)) \\ +\sum_{k\in\mathcal{K}}H_k(t)[\alpha_k x_k(t) - \beta_k - y_k(t)] \\ +\sum_{k\in\mathcal{K}}Q_k(t)[\gamma_k(t) - x_k(t)] \quad (17)\end{aligned}$$

where $B(t)$ is defined:

$$\begin{aligned}B(t) \triangleq\; & \frac{1}{2}\sum_{k\in\mathcal{K}}(\alpha_k x_k(t) - \beta_k - y_k(t))^2 \\ & +\frac{1}{2}\sum_{k\in\mathcal{K}}(\gamma_k(t) - x_k(t))^2\end{aligned}$$

Proof: Squaring (15) and using $\max[y,0]^2 \leq y^2$ for any real number y yields:

$$\begin{aligned}H_k(t+1)^2 \leq\; & H_k(t)^2 + (\alpha_k x_k(t) - \beta_k - y_k(t))^2 \\ & 2H_k(t)(\alpha_k x_k(t) - \beta_k - y_k(t))\end{aligned}$$

Similarly, squaring (16) gives:

$$\begin{aligned}Q_k(t+1)^2 \leq\; & Q_k(t)^2 + (\gamma_k(t) - x_k(t))^2 \\ & +2Q_k(t)(\gamma_k(t) - x_k(t))\end{aligned}$$

Summing over $k \in \mathcal{K}$ and dividing by 2 yields the result. $\square$

The value of $B(t)$ can be upper bounded by a finite constant B every slot, where B depends on the maximum possible values that $\mu_{nk}(t)$ and $\gamma_k(t)$ can take. The algorithm below is defined by observing the queue states and $\omega(t)$ every slot t, and choosing actions to minimize the last three terms on the right-hand-side of (17) (not including the first term $B(t)$), given these observed quantities. Using definitions of $x_k(t)$ and $y_k(t)$ in (2)-(3) leads to the following on each slot t:

- ($\gamma_k(t)$ decisions) Each user $k \in \mathcal{K}$ observes $Q_k(t)$ and chooses $\gamma_k(t)$ to solve:

$$\begin{aligned}\text{Maximize:} \quad & V\phi_k(\gamma_k(t)) - Q_k(t)\gamma_k(t) \\ \text{Subject to:} \quad & 0 \leq \gamma_k(t) \leq x_k^{max}\end{aligned}$$

- ($\mu_{nk}(t)$ decisions) The network controller observes all queues $(\boldsymbol{Q}(t), \boldsymbol{H}(t))$ and the topology state $\omega(t)$ on slot t, and chooses matrix $(\mu_{nk}(t)) \in \mathcal{R}(\omega(t))$ to maximize the following expression:

$$\sum_{n\in\mathcal{N}}\sum_{k\in\mathcal{K}}\mu_{nk}(t)f_{nk}(t)W_{nk}(t) \quad (18)$$

  where weights $W_{nk}(t)$ are defined:

$$W_{nk}(t) \triangleq \left[Q_k(t) + 1_{\{n\in\mathcal{K}\}}H_n(t) - \alpha_k H_k(t)\right]$$

  where $1_{\{n\in\mathcal{K}\}}$ is an indicator function that is 1 if device n is a user, and 0 if device n is an access point.
- (Queue updates) Update virtual queues $H_k(t)$ and $Q_k(t)$ for all $k \in \mathcal{K}$ via (15) and (16).

The $\gamma_k(t)$ decisions can be viewed as flow control actions that restrict the amount of data requested from user k on each slot. They are made separately at each user k. The $(\mu_{nk}(t))$ decisions are transmission actions made at the network layer. Examples are given below.

C. Example Flow Control Decisions

Suppose we have the following strictly concave utility functions for all $k \in \mathcal{K}$:

$$\phi_k(x_k) = \ln(1 + \nu_k x_k) \quad (19)$$

Then the $\gamma_k(t)$ decisions are:

$$\gamma_k(t) = \left[\frac{V}{Q_k(t)} - \frac{1}{\nu_k}\right]_0^{x_k^{max}} \quad (20)$$

where the operation $[y]_0^b$ is equal to y if $0 \leq y \leq b$, 0 if $y < 0$, and b if $y > b$. These utility functions can be viewed as an accurate approximation of the proportionally fair utility function if we use $\nu_k = \nu$ for all k, for a large value of ν. Alternatively, using the proportionally fair utilities $\phi_x(x_k) = \ln(x_k)$ for all $k \in \mathcal{K}$ leads to $\gamma_k(t) = [V/Q_k(t)]_0^{x_k^{max}}$, which is indeed the same as (20) in the limit as $\nu_k \rightarrow \infty$.

D. Example Transmission Decisions

Suppose the network has the special structure specified in Section II-A. Let $\mathcal{A}$ be the set of access points. Let $\mathcal{K}_a(t)$ be the set of users within reach of access point a on slot t. Then each access point $a \in \mathcal{A}$ observes channels $S_{ak}(t)$ and queues $Q_k(t)$, $H_k(t)$ for all users $k \in \mathcal{K}_a(t)$ and chooses to serve the single user in $\mathcal{K}_a(t)$ with the largest non-negative value of $f_{ak}(t)S_{ak}(t)[Q_k(t) - \alpha_k H_k(t)]$ (breaking ties arbitrarily), and chooses no users if this value is negative for all $k \in \mathcal{K}_a(t)$.

Further, the user pairs in each subcell $c \in \{1, \ldots, C\}$ are observed. Amongst all users in a given subcell, the ordered user pair (a, k) with the largest non-negative value of $f_{ak}(t)S_{ak}(t)[Q_k(t) + H_a(t) - \alpha_k H_k(t)]$ is selected for peer-to-peer transmission in that subcell (breaking ties arbitrarily). No peer-to-peer transmission occurs in the subcell if this value is negative for all user pairs.

IV. Algorithm Performance

The algorithm fits into the stochastic network optimization framework of [15], and hence it satisfies all desired constraints of the original problem (4)-(6) (as well as the transformed problem (7)-(11)), with an overall utility value that differs from the optimal ϕ^* by at most $O(1/V)$, which can be made arbitrarily small as the parameter V is increased. However, the V parameter directly affects the size of the virtual queues, which affects convergence time of the algorithm to this utility. Here we show that if the utility functions and network transmission sets have additional structure, the virtual queues $Q_k(t)$ and $H_k(t)$ can be deterministically bounded for all time by a constant that is proportional to V. Proofs are omitted due to page length restrictions (see [16] for full proofs).

A. Bound on Data Queues $Q_k(t)$

Suppose each utility function $\phi_k(t)$ has right-derivatives that are bounded by a finite constant $\nu_k > 0$ over the interval $0 \leq x_k \leq x_k^{max}$. For example, this holds for the utility functions (19), with the ν_k parameters specified there indeed being the maximum right derivatives. Alternatively, if $\phi_k(x) = \min[x, \theta_k]$, then $\nu_k = 1$.

Lemma 2: If utility function $\phi_k(x)$ has maximum right derivative $\nu_k > 0$, then:

$$0 \leq Q_k(t) \leq V\nu_k + x_k^{max} \;\; \forall t \in \{0, 1, 2, \ldots\}$$

provided that this inequality holds for $Q_k(0)$.

B. Bound on Reputation Queues $H_k(t)$

The $H_k(t)$ processes act as *reputation queues* for each user $k \in \mathcal{K}$, being low if user k has a good reputation for helping others, and high otherwise (see queue dynamics in (15)). These reputations directly affect the transmission decisions via the weights $W_{nk}(t)$ in (18). To see this, define $\mathcal{A}$ as the set of access points. First consider the weight seen by an access point $a \in \mathcal{A}$ for user k on slot t:

$$Q_k(t) - \alpha_k H_k(t)$$

This weight is large if $H_k(t)$ is small.

Lemma 3: If $\phi_k(t)$ has maximum right-derivative $\nu_k > 0$, and if initial queue backlog satisfies $0 \leq Q_k(0) \leq V\nu_k + x_k^{max}$, then no access point will send to user k on a given slot t if $H_k(t) > \frac{1}{\alpha_k}[V\nu_k + x_k^{max}]$.

Now suppose user $u \in \mathcal{K}$ considers transmitting to another user $k \in \mathcal{K}$. User u sees the weight:

$$Q_k(t) + H_u(t) - \alpha_k H_k(t)$$

The value $H_u(t) - \alpha_k H_k(t)$ can be viewed as a *differential reputation*. We again see that a relatively low value of $H_k(t)$ improves the weights for user k. The next lemma shows that all queues $H_k(t)$ are deterministically bounded. For simplicity, we state the lemma under the assumption that all initial queue backlogs are zero.

Lemma 4: If all utility functions have right-derivatives bounded by finite constants $\nu_k > 0$, if $\beta_k > 0$ for all $k \in \mathcal{K}$, and if initial backlog satisfies $Q_k(0) = H_k(0) = 0$ for all $k \in \mathcal{K}$, then there are finite constants C_1 and C_2, both independent of V, such that:

$$||\mathbf{\Theta}(t)|| \leq C_1 + C_2 V \;\; \forall t \in \{0, 1, 2, \ldots\}$$

where $||\mathbf{\Theta}(t)||$ is defined:

$$||\mathbf{\Theta}(t)|| \leq \sqrt{\sum_{k\in\mathcal{K}} H_k(t)^2 + \sum_{k\in\mathcal{K}} Q_k(t)^2}$$

These deterministic queue bounds ensure that the time average tit-for-tat constraints (8), as well as the auxiliary variable constraints (9), are satisfied on every sample path, regardless of whether or not the topology state process $\omega(t)$ is ergodic (see [16]).

V. Extension to Finite File Sizes

Now suppose the files requested by users have finite sizes. Suppose each user requests at most one file at a time. We say a user is in the *active* state if it is requesting a file, and is in the *idle* state if it does not have any file requests. For each $k \in \mathcal{K}$, define $A_k(t)$ to be 1 if user k is active on slot t, and 0 else. If $A_k(t) = 1$, define $\mathcal{F}_k(t)$ as the set of devices in $\mathcal{N}$ that have the currently requested file of user k. Define $\mathcal{F}_k(t)$ to be the empty set $\{\}$ if the user is not active on slot t. Define $D_k(t)$ as the number of additional required packets for user k to complete its file request (where $D_k(t) > 0$ if and only if $A_k(t) = 1$). When an active user finishes downloading all packets of its requested file on some slot t, it goes to the idle state, so that $A_k(t+1) = D_k(t+1) = 0$, and $\mathcal{F}_k(t+1) = \{\}$.

We can naturally extend the algorithm developed in the previous section to this case (although in this more general case the algorithm is a heuristic). The only change is that $\mathcal{F}_b$ in (1) is changed to $\mathcal{F}_b(t)$. The algorithm proceeds exactly as before, still updating queues and making all $\gamma_k(t)$ and $\mu_{ab}(t)$ decisions the same way for all users on every slot, regardless of whether users are active or idle. Of course, the $f_{ab}(t)$ parameters in (18) and in the receive and send equations (2), (3) will remove any transmission link (a, b) from consideration if user b is idle. However, idle users can still participate in data delivery to other users. Intuitively, the algorithm will behave well, with performance close to that suggested by the infinite file size assumption, when file sizes are large.

VI. Simulation

We simulate the algorithm on the cell-partitioned network structure of Section II-A, with $K = 50$ users and a single base station as access point. The users move according to a Markov random walk on a 4×4 grid with 16 subcells. We use utility functions $\phi_k(x)$ given by (19) with $\nu_k = 1$. Each cell can support at most one user-to-user packet transmission per slot. The base station can transmit to at most one user k per slot, with transmission rate $S_k(t)$ that is independent over slots and across users with $Pr[S_k(t) = 0] = Pr[S_k(t) = 1] = Pr[S_k(t) = 2] = 1/3$. We simulate over 10^6 slots. On slot $t = 0$, we assign each user k a desired file that is independently in the other users with probability $p = 0.05$, which establishes the $\mathcal{F}_k$ sets. These sets are held fixed for the first third of the simulation, and then independently drawn again with a larger probability $p = 0.1$ and held fixed for the second third. New files are again drawn at the beginning of the final third of the simulation, with $p = 0.07$.

Fig. 1 plots the resulting throughput components from the base station traffic and peer-to-peer traffic separately, using $V = 10$, $\alpha_k = \alpha = 0.5$, $\beta_k = 0.05$, $x_k^{max} = 3$. Even though there are only an average of 50/16 = 3.125 users per cell, and in the first third of the simulation there is only a 5% chance that a given user has the file desired by another user, the peer-to-peer traffic is still more than twice that of the base station alone. This further increases in the middle of the simulation when the file availability probability jumps to 10%. Fig. 2 shows that the value of $Q_k(t)$ never exceeds 10 packets for any user k (recall that the worst-case guarantee from Lemma 2 is $Q_k(t) \leq V + 3 = 13$ packets). All tit-for-tat constraints were satisfied, with $H_k(t) \leq 24.6$ for all $k \in \mathcal{K}$ and all t.

Figs. 3 and 4 explore the throughput-backlog tradeoff with V (one can also plot the throughput-utility with V to see a similar convergence as in Fig. 3). Fig. 3 also treats the case when the tit-for-tat constraint is made more stringent ($\alpha = 0.75$), in which case throughput is reduced.

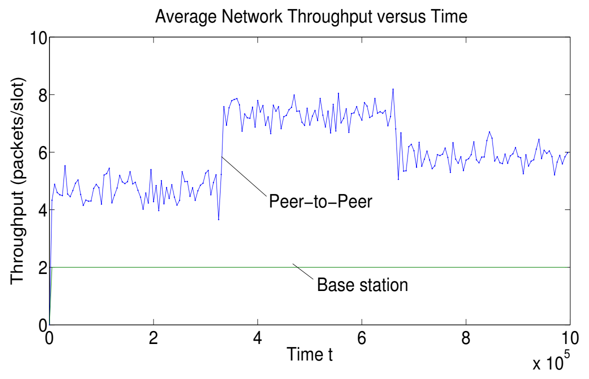

Fig. 1. Average throughput per-user versus time.

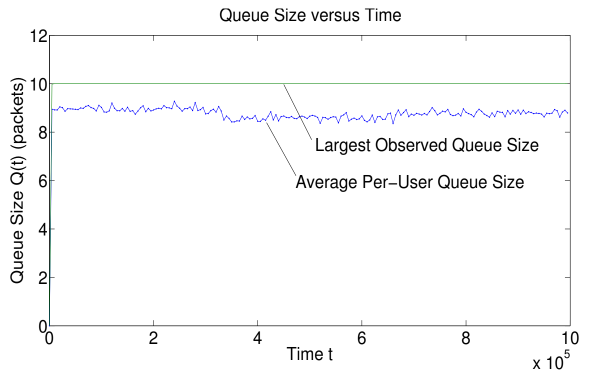

Fig. 2. Average and worst-case queue backlog per user versus time.

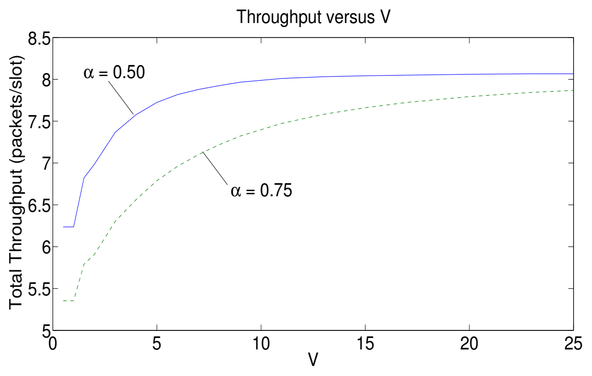

Fig. 3. Throughput versus V for $\alpha = 0.5$ and $\alpha = 0.75$.

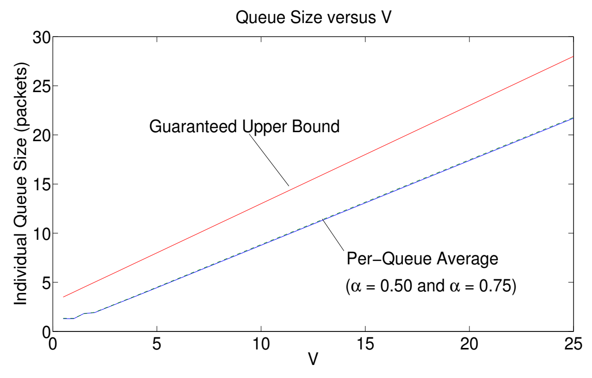

Fig. 4. Queue backlog versus V, demonstrating the $O(V)$ behavior. The results for $\alpha = 0.5$ and $\alpha = 0.75$ are right on top of each other. The upper bound $V + 3$ is also plotted.

References

[1] M. J. Neely and L. Golubchik. Utility optimization for dynamic peer-to-peer networks with tit-for-tat constraints. *Proc. IEEE INFOCOM*, 2011.

[2] L. Buttyan and J.-P. Hubaux. Stimulating cooperation in self-organizing mobile ad hoc networks. *ACM/Kluwer Mobile Networks and Applications (MONET)*, vol. 8, no. 5, pp. 579-592, Oct. 2003.

[3] J. Crowcroft, R. Gibbens, F. Kelly, and S. Ostring. Modeling incentives for collaboration in mobile ad-hoc networks. *presented at the 1st Int. Symp. Modeling and Optimization in Mobile, Ad-Hoc, and Wireless Networks (WiOpt '03), Sophia-Antipolis, France*, March 2003.

[4] M. J. Neely. Optimal pricing in a free market wireless network. *Wireless Networks*, vol. 15, no. 7, pp. 901-915, October 2009.

[5] S. Jun and M. Ahamad. Incentives in bittorrent induce free riding. In *The 3rd ACM SIGCOMM Workshop on Economics of Peer-to-Peer Systems (P2PECON)*, Philadelphia,PA, August 2005.

[6] A. R. Bharambe, C. Herley, and V. N. Padmanabhan. Analyzing and improving bittorrent performance. In *The 25th Conference on Computer Communications (INFOCOM)*, Barcelona, Catalunya, Spain, April 2006.

[7] Karthik Tamilmani, Vinay Pai, and Alexander E. Mohr. Swift: A system with incentives for trading. In *P2PECON*, 2004.

[8] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani. Do incentives build robustness in bittorrent? In *The 4th USENIX Symposium on Networked Systems Design & Implementation (NSDI)*, Cambridge, MA, April 2007.

[9] W.-C. Liao, F. Papadopoulos, and K. Psounis. Performance analysis of bittorrent-like systems with heterogeneous users. In *Performance*, 2007.

[10] Lian, Peng, Yang, Zhang, Dai, and Li. Robust incentives via multi-level tit-for-tat. In *IPTPS*, 2006.

[11] Freedman, Aperjis, and Johari. Prices are right: Managing resources and incentives in peer-assisted content distribution. In *IPTPS*, 2008.

[12] 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, Dec. 1992.

[13] L. Georgiadis, M. J. Neely, and L. Tassiulas. Resource allocation and cross-layer control in wireless networks. *Foundations and Trends in Networking*, vol. 1, no. 1, pp. 1-149, 2006.

[14] F. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, vol. 8, no. 1 pp. 33-37, Jan.-Feb. 1997.

[15] M. J. Neely. *Stochastic Network Optimization with Application to Communication and Queueing Systems*. Morgan & Claypool, 2010.

[16] M. J. Neely. Wireless peer-to-peer scheduling in mobile networks. *ArXiv Technical Report, arXiv:1202.4451*, Feb. 2012.