Utility Optimization for Dynamic Peer-to-Peer Networks with Tit-For-Tat Constraints

Michael J. Neely , Leana Golubchik

***Abstract*— We consider a peer-to-peer network where nodes can send and receive files amongst their peers. File requests are generated randomly, and each new file can correspond to a different subset of peers that already have the file and hence can assist in the download. Nodes that help others are rewarded by being able to download more. The goal is to design a control algorithm that allocates requests and schedules transmissions to maximize overall throughput-utility, subject to meeting "tit-for-tat" constraints that incentivize participation. Our algorithm is shown to operate efficiently on networks with arbitrary traffic and channel sample paths, including wireless networks whose capacity can be significantly extended by the peer-to-peer functionality.**

***Index Terms*— Queueing analysis, optimization, Lyapunov drift**

I. Introduction

This paper develops utility-efficient scheduling algorithms for peer-to-peer communication networks. We particularly consider networks with randomly arriving file requests and time-varying connectivity. This includes wireless networks with (possibly non-ergodic) traffic and channels. We assume there are N nodes in the network that participate in peer-to-peer communication, and denote this set of nodes by $\mathcal{N} = \{1, \ldots, N\}$. There is an additional special node, called "node 0," that is connected to all others at all times. The special node 0 contains information about files that are located in each of the other nodes, and can coordinate request allocation and transmission scheduling.

The nodes connect to each other according to possibly different peer-to-peer social groups. Specifically, each node $n \in \mathcal{N}$ has a *download social group* $\mathcal{G}_n \subseteq \mathcal{N} \cup \{0\}$ from which it can download files. We assume node 0 does not desire any files, but it can possibly transmit files to others. Each new file request from a node $n \in \mathcal{N}$ is assumed to be contained in at least one (but possibly not all) of the nodes in the download social group $\mathcal{G}_n$. Each file consists of an integer number of fixed size packets, and these packets can be transmitted to node n by one or more of the nodes that have the file and are in the set $\mathcal{G}_n$.

The system operates in slotted time with unit timeslots $t \in \{0, 1, 2, \ldots\}$. Every slot the special node 0 coordinates the packet transfers. For simplicity, we consider only 1-hop communication. Let $S(t)$ denote the current *topology state* on slot t, which represents the collection of node locations and channel states between links, as in [1]. Every slot t, node 0 observes the current topology state $S(t)$ and chooses a matrix $\boldsymbol{\mu}(t) = (\mu_{ab}(t))$ of *transmission rates* within a set of rate matrix options $\Gamma_{S(t)}$. The rate $\mu_{ab}(t)$ represents the (integer) number of packets that can be transmitted over the (a, b) link on slot t. We consider two basic models that fit this description: (i) The *Internet cloud model*, and (ii) The *wireless basestation model*.

M. J. Neely is with the Electrical Engineering department and L. Golubchik is with the Computer Science department at the University of Southern California, Los Angeles, CA.

This material is supported in part by one or more of the following: the NSF Career grant CCF-0747525, NSF grant 0540420, the Network Science Collaborative Technology Alliance sponsored by the U.S. Army Research Laboratory W911NF-09-2-0053.

Fig. 1. An illustration of the Internet cloud with N access points that form the peer-to-peer network.

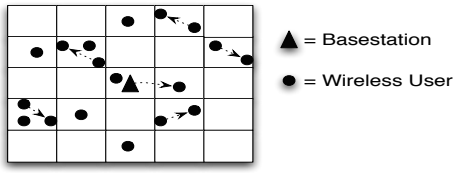

Fig. 2. The wireless basestation model with multiple sub-cells.

A. The Internet Cloud Model

In this model, node 0 represents an Internet cloud with high speed inter-connections, and all other nodes $\{1, \ldots, N\}$ are access points that form a peer-to-peer network (see Fig. 1). Because node 0 here represents an abstract network that has many components, the peer-to-peer scheduling does not need to be centralized and can often be distributed amongst individual social groups. In this model, node 0 can actually represent several different "seed nodes" that offer data without requiring any in return.

A "1-hop" packet transfer from node $a \in \mathcal{N}$ to node $b \in \mathcal{N}$ on a particular slot t is actually implemented via a possibly multi-hop path through the Internet cloud. However, we ignore the details of routing protocols and delays within the cloud, and simply treat the cloud as an abstraction that can support timely data transfer. Thus, the transmission rate between node a and node b is limited only by the minimum of the uplink and downlink capacities from node a to the cloud and from the cloud to node b. This model is a reasonable first-order approximation if the network that forms the interior of the cloud operates with a much higher speed than the access

point uplinks and downlinks. The model highlights the peer-to-peer scheduling issues between the access points, and leads to robust peering algorithms that we believe will work well when implemented on top of any high-speed network.

In the Internet cloud model, we typically assume the *topology state* $S(t)$ is the same for all time. However, we can also consider states $S(t)$ that change with time. For example, a particular node in the set $\mathcal{N}$ might leave the peer-to-peer environment for some time and return later.

B. The Wireless Basestation Model

In this model, node 0 is a basestation and all other nodes $\{1, \ldots, N\}$ are potentially mobile devices that move within a cellular region. In addition to connecting to the basestation and possibly receiving files from it, these other devices can connect directly with each other in sub-cells or "femtocells" of the network area (see Fig. 2). The topology state $S(t)$ in this case represents the collection of current channel states between devices, and these states can change due to mobility. Current cellular capacity is limited by the fact that all transmissions go through the basestation. By using such additional low-power "device-to-device" transmissions, possibly coordinated by the basestation to ensure low interference [2][3], one can expand capacity by a factor proportional to the number of sub-cells. This is because popular files are likely to be in the cache of neighboring nodes, and hence can be delivered in a 1-hop sub-cell transmission. This greatly alleviates the load on the basestation downlink. However, this gain can only be achieved through wireless peer-to-peer communication.

C. Incentives for Peering

A node $n \in \mathcal{N}$ sees three main disadvantages in sending data to others. First, this may take away from its uplink capacity and thus reduce the rate at which it can send its own traffic (such as traffic it wants to send to a destination that is outside of the peer-to-peer network). Second, it may take away from its downlink capacity and thus reduce the rate at which it can receive data. This happens when the uplink and downlink are not orthogonal, such as in wireless networks where a node can either transmit or receive on one slot, but not both. Third, such transmissions can be costly in terms of energy expenditure. Therefore, a node will not send data to another node unless it has some incentive for doing so.

The same issues of incentives arise in ad-hoc mobile networks, where token-based and economics-based incentive mechanisms have been studied [4][5][6]. Incentives are also well studied in the peer-to-peer literature [7][8][9][10][11][12]. In this paper, we consider a simple "tit-for-tat" type incentive mechanism that forces the long-term rate of delivering data to peers to be greater than or equal to some factor α (where $0 \leq \alpha \leq 1$) of the long-term rate of receiving data. Here we use the term "tit-for-tat" broadly to represent the notion of giving as much as we receive (perhaps more appropriately called "treat-for-treat"), which is qualitatively similar to specific notions of tit-for-tat used in game theory.

D. Our Approach

We enforce the tit-for-tat constraint using the *drift-plus-penalty* and *virtual queue* techniques from Lyapunov optimization theory [1][13]. Lyapunov optimization was developed for stochastic networks without peer-to-peer capabilities in [1][14][15]. This paper is perhaps the first use of Lyapunov optimization techniques in a peer-to-peer setting. The extension is not obvious. Indeed, the prior work on Lyapunov optimization treats each data type as a separate *commodity*, and develops algorithms with complexity and convergence time that grow polynomially with the number of commodities. In peer-to-peer networks, there can be an infinite number of files. One can try grouping files into commodities defined by the *subset of nodes that have this file*, so that different files are treated as the same commodity if the subset of nodes that have them is exactly the same. However, this approach leads to a complexity explosion due to the number of commodities being exponential in the number of nodes. Instead, in this paper we overcome this difficulty by using the subset information associated with each new file only on the slot when the file enters, making scheduling plans on that slot and discarding this subset information at the end of the slot. This results in only one commodity per node, enabling a low complexity solution that, we will show, can still approach optimality.

This paper also exploits a sample-path version of Lyapunov optimization that we recently developed in [16] in the context of non-peer-to-peer networks. We apply this framework here because file requests are typically non-ergodic and may have long periods of inactivity followed by bursts of many requests. Channel states for wireless networks can also be non-ergodic. We show that our algorithms are efficient for arbitrary sample paths for traffic and channels, with no assumed probability model. Traditional notions of equilibrium and steady state throughput-utility are not appropriate for this context [17]. We thus evaluate performance in this setting using the T*-slot lookahead metric* from [16].

For simplicity, we only explicitly consider the peer-to-peer traffic in the network. We do not model the non-peer-to-peer traffic that a node generates and wants to send to a node outside of the set $\mathcal{N}$. While we could explicitly put this traffic into our framework, we have chosen not to do so mainly to reduce notation. Instead, we indirectly account for such external traffic by having each node incur an uplink transmission cost whenever it sends a packet to another peer. This cost can represent power expenditure, or it can model the "cost" associated with not using the link for its own traffic.

E. Related Work

A number of works (e.g., [7][8][9][10][11][12]) studied various incentive-related issues in the context of peer-to-peer systems. For instance, [10] argues, using a game-theoretic framework, that the incentive mechanisms in the original BitTorrent protocol [18] are not sufficiently effective and considers a mechanism that maintains a certain "deficit" between uploaded and downloaded data. Other efforts consider similar approaches, such as bounding the difference between the amount of uploaded and downloaded data (as in [7],

[12]) or favoring links with a greater download to upload ratio (as in [9]). Token-based techniques that account for upload contributions versus downloaded data have also been investigated, e.g., as in [11]; such techniques allow a system to tradeoff overall performance (or social good) versus better performance for higher contributing peers. Market-based and reputation-based techniques (e.g., as in [19]) have also been considered in the context of peer-to-peer systems. For instance, [20] uses a market based approach to incentivize sharing, where efficient network resource allocation is achieved by relating file values (in the market) to their relative demand. Our proposed approach differs in that it is a general framework (useful in both Internet and wireless environments) with provable performance characteristics (optimizing throughput utility while maintaining tit-for-tat type constraints) that accounts for multi-swarm environments (e.g., as in [21]).

Utility optimization for static Internet and wireless network models is considered in [22][23][24], and stochastic networks are considered in [25][26][14][27][15][28][29][1], including the Lyapunov optimization work that we discussed in the previous subsection. These prior works treat traditional network scenarios without peer-to-peer communication, and assume that any time-variation in the network is ergodic.

II. Network Model

Each node $n \in \mathcal{N}$ can generate requests for files, and each file f has a particular subset $\mathcal{N}_f \subseteq \mathcal{N} \cup \{0\}$ consisting of nodes that contain the file. It is assumed that if node n requests a file f, then $n \notin \mathcal{N}_f$ (else, it would not need to request the file). While each file may be very large, perhaps containing 1000 packets, it is important that these packet requests are apportioned over slots so that at most A_n^{max} packet requests are generated by node n on one slot. The A_n^{max} value can be much smaller than the file size, and can be chosen as the maximum number of packets that node n can receive from other nodes on a slot. For example, in a wireless network where a node can receive at most one packet per slot, we use $A_n^{max} = 1$. Let $A_n(t)$ represent the number of packet requests generated by node n on slot t (where $0 \leq A_n(t) \leq A_n^{max}$ for all t), which can be viewed as an "arrival process."

We assume that packet requests $A_n(t)$ generated by node n on slot t are all for the same file, but that packets of different files can be requested on different slots. Define $\mathcal{N}_n(t)$ as the subset of nodes that contain the file associated with the packets of $A_n(t)$. We do not impose any probabilistic model for $A_n(t)$ and $\mathcal{N}_n(t)$. However, a "typical" $A_n(t)$ process is zero for some duration of time until a file request is generated by node n. It then has a long string of slots for which $A_n(t) = A_n^{max}$, a final slot where the residual packets of the file are requested, followed by another idle period until the next request (for a different file) is generated.

On each slot t, each node $n \in \mathcal{N}$ informs the special node 0 of its current requests (including the $A_n(t)$ packets and the file these are from). The special node observes the file label to find the set $\mathcal{N}_n(t)$ of nodes that have the file, and then makes *request allocation decisions* $R_{mn}(t)$, representing the number of packets it asks node $m \in \mathcal{N} \cup \{0\}$ to deliver to node n. These decision variables are constrained as follows for all $n \in \mathcal{N}$ and $m \in \mathcal{N} \cup \{0\}$:

$$R_{mn}(t) \in \{0, 1, \ldots, A_n^{max}\} \ , \ \sum_{m=0}^{N} R_{mn}(t) \leq A_n(t) \quad (1)$$

$$R_{mn}(t) = 0 \text{ if } m \notin \{\mathcal{N}_n(t) \cap \mathcal{G}_n\} \quad (2)$$

The constraints in (1) ensure that requests for at most $A_n(t)$ packets are allocated (as integer units) to the different nodes. If $\sum_{m=0}^{N} R_{mn}(t)$ is strictly less than $A_n(t)$, then the un-allocated packet requests are *dropped* (described in more detail in the next subsection). The constraint (2) ensures that a node can be allocated a request from node n on slot t only if it is in the download social group $\mathcal{G}_n$ for node n *and* it has the corresponding file. We assume the set $\{\mathcal{N}_n(t) \cap \mathcal{G}_n\}$ is non-empty. In the case when node 0 represents a wireless basestation that coordinates the transfers, it is often useful to assume node 0 has *all* the files and is in the download social group of every node $n \in \mathcal{N}$. On the other hand, as in the Internet cloud model, centralized control is not crucial to our model, and node 0 can actually represent a collection of N different "information nodes" $\{\mathcal{O}_n\}_{n=1}^{N}$, each containing file information associated with download social group $\mathcal{G}_n$.

The requests are put in a *request queue* $Q_{mn}(t)$, representing the total number of not-yet-delivered packets that node m has been requested to deliver to node n. Note that $Q_{mn}(t)$ may contain requests for packets of different files. While this information is important at the time of transmission (so that the correct packet can be transmitted), it is no longer needed for our higher layer scheduling decisions. That is because all data in this queue is now treated as a single commodity that must be delivered to node n. The queue update is given as follows for all $n \in \mathcal{N}$ and $m \in \mathcal{N} \cup \{0\}$:

$$Q_{mn}(t+1) = \max[Q_{mn}(t) + R_{mn}(t) - \mu_{mn}(t), 0] \quad (3)$$

where $\mu_{mn}(t)$ is a control decision variable representing the amount of packets node m can transmit to node n on slot t.

A. Recovering from Packet Drops

It is important to limit the amount of data requested so that the network is not overloaded. Hence, as suggested by the second constraint in (1), some of the packet requests $A_n(t)$ may not be allocated on slot t. When analyzing the queueing equations and computing throughput, these un-allocated requests are treated as if they are dropped—so those packets are not delivered. However, such dropping may result in missing pieces for the files. We thus consider two models for recovery:

Model 1: Dropped packets become the next arrivals. In this model, we keep a transport layer queue that buffers dropped packets and puts them into the next "generated requests" $A_n(t+d)$ for some $d > 0$. This ensures all files are persistently served. The T-slot analysis we soon use does apply to this model. However, comparison to ideal algorithms that know the future is less meaningful here because "future arrivals" now must be defined as those that arise from our implemented control decisions.

Model 2: Dropped packets are exchanged for the next arrivals whenever possible. In this model, our throughput analysis is compared against algorithms that know the future,

where the future arrivals are the raw arrival processes that do not depend on our past decisions. However, the dropped data is not forgotten. While dropped requests for destination n are removed from the request queue, they are placed in a *dropped-packet-queue*. Any time a new request for packet delivery to node n is made and assigned to a particular node k (so that it would be placed in the queue $Q_{kn}(t)$), we also check the head-of-line packet in the type n dropped-packet-request queue. If this head-of-line packet is for a file that is *also* contained in node k, then this old (dropped) packet request is switched and placed in the node k (i.e., in the queue $Q_{kn}(t)$), while the new request is placed in the dropped packet queue.

B. Transmission Decisions

Define $\boldsymbol{\mu}(t) = (\mu_{ab}(t))$ as a matrix of transmission decisions for $a \in \mathcal{N} \cup \{0\}$ and $b \in \mathcal{N}$. Every slot t, node 0 observes the current topology state $S(t)$ and chooses $\boldsymbol{\mu}(t) \in \Gamma_{S(t)}$. The set $\Gamma_{S(t)}$ represents transmission options under a given $S(t)$. We assume it has the natural property that for any $\boldsymbol{\mu} \in \Gamma_{S(t)}$, the matrix $\boldsymbol{\mu}'$ that sets one or more entries of $\boldsymbol{\mu}$ to zero is also in $\Gamma_{S(t)}$. We further assume that all matrices $\boldsymbol{\mu}(t) \in \Gamma_{S(t)}$ have components $\mu_{mn}(t)$ that are integers within the set $\{0, 1, \ldots, \mu_{mn}^{max}\}$ for some finite values μ_{mn}^{max}. Finally, we assume that $\mu_{mn}(t) = 0$ whenever $m \notin \mathcal{G}_n$, i.e., whenever m is not in the download social group of node n.

For example, in the Internet cloud model, we can assume that $S(t)$ is fixed and represents the uplink and downlink capacities between all nodes $n \in \mathcal{N}$ and the cloud, and $\boldsymbol{\mu}(t)$ is constrained by:

$$\textstyle\sum_{a\in\mathcal{N}} \mu_{an}(t) \leq C_n^{downlink} \ \ \forall n \in \mathcal{N} \quad (4)$$

$$\textstyle\sum_{b=1}^{N} \mu_{nb}(t) \leq C_n^{uplink} \ \ \forall n \in \mathcal{N} \quad (5)$$

A simpler special case of this is when the downlink capacity (for receiving data) is much larger than the uplink (for sending data), so that the constraints (4) are inactive and transmissions are limited only by (5). For the wireless basestation model, $S(t)$ can represent the current node pairs that are close enough for communication, and the set $\Gamma_{S(t)}$ can be the set of all $0/1$ matrices that restrict nodes from either transmitting one packet per slot, or receiving one packet per slot, but not both. Alternatively, it might contain the collection of *independent sets* of links that can be simultaneously activated subject to a general interference model.

Another important example (that holds for either the Internet cloud model or the wireless basestation model) is the *no transmission scheduling* model: In this model, the set $\Gamma_{S(t)}$ consists only of a single "dominant" rate matrix $(\mu_{an}(t))$, being the current rates that can be supported under the given topology state, together with all sub-matrices obtained by setting one or more entries of $(\mu_{an}(t))$ to zero. The rates $(\mu_{an}(t))$ might be determined by a transport and routing protocol that is uncontrollable by the peer-to-peer application. Our sample path analysis allows for arbitrary topology state variations and hence is compatible with any such lower level protocols. However, in our mathematical analysis that compares our peer-to-peer decisions against those that could be implemented by an "ideal" algorithm with perfect knowledge of the future $S(t)$ values, we use the $S(t)$ sample path that actually arises (under our peer-to-peer decisions). This comparison is the most meaningful when future $S(t)$ values are not influenced by past peer-to-peer control decisions. This is an approximation in some cases, such as when the Internet transport and routing protocols that affect the current delivery rates $(\mu_{an}(t))$ are influenced by the amount of peer-to-peer data that has been delivered in the past. However, the approximation is reasonable when the traffic generated by our N-node peer-to-peer community is small in comparison to all other Internet traffic.

Let $p_n(t)$ be a penalty incurred by node n for transmitting data on its uplink channel. We assume $p_n(t)$ is a general function of the transmission rate vector:

$$p_n(t) \triangleq \textstyle\sum_{b=1}^{N} \eta_{nb}(S(t))\mu_{nb}(t)$$

where the function $\eta_{nb}(S(t))$ is a cost-per-unit transmission rate incurred over link (n, b) when the topology state is $S(t)$.

C. Utility Functions and Auxiliary Variables

To define our control objective, it is useful to (temporarily) assume we have an ergodic system with well defined time averages. For each $m \in \mathcal{N} \cup \{0\}$, $n \in \mathcal{N}$, define $\overline{r}_{mn}$ as the time average of the request allocations $R_{mn}(t)$:

$$\overline{r}_{mn} \triangleq \lim_{t\to\infty} \tfrac{1}{t} \textstyle\sum_{\tau=0}^{t-1} R_{mn}(\tau)$$

Define $\overline{y}_n \triangleq \sum_{a=0}^{N} \overline{r}_{an}$. Assuming we use an algorithm that ensures all allocated requests are eventually fulfilled, then $\overline{y}_n$ represents the total throughput delivered to node n. Now define $g_n(y)$ as a continuous, concave, and non-decreasing *utility function* of the node n throughput. We assume the right-derivatives of the $g_n(y)$ functions are finite at $y = 0$, and define $\nu_n \triangleq g_n'(0)$. Let $\overline{p}_n$ be the time average penalty incurred by node n, and let $\overline{Q}_{mn}$ be the time average queue backlog for queue (m, n). Our objective is thus to solve:

$$\text{Maximize:} \quad \textstyle\sum_{n\in\mathcal{N}} [g_n(\sum_{a=0}^{N} \overline{r}_{an}) - \overline{p}_n] \quad (6)$$

$$\text{Subject to:} \quad \alpha \textstyle\sum_{a=0}^{N} \overline{r}_{an} \leq \beta + \sum_{b=1}^{N} \overline{r}_{nb} \ \ \forall n \in \mathcal{N} \quad (7)$$

$$\overline{Q}_{mn} < \infty \ \ \forall m \in \mathcal{N} \cup \{0\}, \forall n \in \mathcal{N} \quad (8)$$

$$\boldsymbol{\mu}(t) \in \Gamma_{S(t)} \ \ \forall t \quad (9)$$

$$R_{mn}(t) \text{ satisfies (1)-(2) } \forall m, n, \forall t \quad (10)$$

where α, β are constants such that $0 \leq \alpha \leq 1$ and $\beta \geq 0$. The constraints (7) are the *tit-for-tat* constraints showing that what each node $n \in \mathcal{N}$ delivers to others must be at least a factor α of what it receives, minus a constant β. If $\alpha \neq 0$, the value β/α represents the "free" rate at which nodes can download without requiring any uploads. Using $\alpha = \beta = 0$ effectively removes the tit-for-tat constraints (7) and still results in a meaningful optimization. The constraints (7) consider only nodes in $n \in \mathcal{N}$ (excluding node 0) because node 0 can only deliver data and does not request any. Constraints (8) are mild queue stability constraints that ensure the time average rate of requests (into the queue) is the same as the time average rate of delivered data (out of the queue). Our resulting algorithm will actually ensure that, under some additional assumptions, all queues are deterministically bounded by finite values Q_{mn}^{max}. Enhancements to worst-case delay bounds are discussed in

Section IV. Constraints (9)-(10) ensure the transmission rates $\mu_{ab}(t)$ and request allocations $R_{mn}(t)$ are chosen subject to the constraints specified in previous subsections.

To solve the problem of optimizing the time average of a concave function, it is important to introduce *auxiliary variables* $\gamma_n(t)$ [1][13], chosen every slot subject to:

$$0 \leq \gamma_n(t) \leq A_n^{max} \ \forall n \in \mathcal{N}, \forall t \quad (11)$$

Define $\overline{\gamma}_n$ as the time average of $\gamma_n(t)$. The problem (6)-(10) can then be modified (without changing the solution) by adding a new constraint:

$$\sum_{a=0}^{N} \overline{r}_{an} \geq \overline{\gamma}_n \ \forall n \in \mathcal{N} \quad (12)$$

and replacing the objective (6) with the following:

$$\text{Maximize: } \sum_{n\in\mathcal{N}}[g_n(\overline{\gamma}_n) - \overline{p}_n]$$

D. The T-Slot Lookahead Utility Metric

Systems with general sample paths may not have an "ergodic optimal utility." Here we develop a modified utility metric that is based on an "ideal" algorithm with limited knowledge of the future. The utility defined here will be used as a target to compare against the actual algorithm we develop in the next section. Fix integers $T > 0$ and $K > 0$, and consider the first KT slots decomposed into K frames of size T. For each frame $k \in \{0, \ldots, K-1\}$, we define the following *T-slot lookahead problem*. The problem optimizes over decision variables $R_{mn}(\tau)$, $\boldsymbol{\mu}(\tau)$ for $\tau \in \{kT, \ldots, kT + T - 1\}$, assuming that the random events $A_n(\tau)$, $\mathcal{N}_n(\tau)$, $S(\tau)$ are *completely known for all τ in this frame*. We further allow the T-slot lookahead problem to relax the decision variable constraints: the $R_{mn}(\tau)$ decisions can be real numbers in the interval $[0, A_n^{max}]$ (not necessarily integers). The $\boldsymbol{\mu}(\tau)$ variables are elements of the *convex hull* of $\Gamma_{S(\tau)}$.

$$\text{Max: } \sum_{n\in\mathcal{N}}[g_n(\gamma_n) - \frac{1}{T}\sum_{\tau=kT}^{kT+T-1}\sum_{b=1}^{N}\mu_{nb}(\tau)\eta_{nb}(S(\tau))] \quad (13)$$

$$\text{Subject to: } \gamma_n = \frac{1}{T}\sum_{\tau=kT}^{kT+T-1}\sum_{a=0}^{N} R_{an}(\tau) \ \forall n \in \mathcal{N} \quad (14)$$

$$A_n(\tau) \geq \sum_{m=0}^{N} R_{mn}(\tau) \ \forall n \in \mathcal{N}, \forall \tau \quad (15)$$

$$0 \leq R_{mn}(\tau) \leq A_n^{max} \ \forall (m,n), \forall \tau \quad (16)$$

$$\boldsymbol{\mu}(\tau) \in Conv(\Gamma_{S(\tau)}) \ \forall \tau \quad (17)$$

$$R_{mn}(\tau) = 0 \ \text{ if } m \notin \{\mathcal{N}_n(\tau), \mathcal{G}_n\} \ \forall \tau \quad (18)$$

$$\frac{1}{T}\sum_{\tau=kT}^{kT+T-1}[\alpha\sum_{a=0}^{N} R_{an}(\tau) - \sum_{b=1}^{N} R_{nb}(\tau)] \leq \beta \ \forall n \in \mathcal{N} \quad (19)$$

$$\sum_{\tau=kT}^{kT+T-1}[R_{mn}(\tau) - \mu_{mn}(\tau)] \leq 0 \ \forall m, n \quad (20)$$

The above problem is analogous to the ergodic problem (6)-(10), but has more stringent "tit-for-tat" constraints (19) that must be enforced exactly over the T-slot frame, rather than over an infinite horizon. Likewise, (20) requires the total requests allocated to node m to be less than or equal to the departures for these requests over the frame. The constraints (14)-(20) are always feasible by the trivial strategy that uses $R_{mn}(\tau) = \mu_{ab}(\tau) = \gamma_n(\tau) = 0$ for all τ.

Define $util_k^*(T)$ as the optimal utility value in (13) for the T-slot lookahead problem for frame k. Let $R_{mn}^*(\tau)$, $\boldsymbol{\mu}^*(\tau)$ be the variables that achieve this optimum in the above problem. Note that it is impossible to causally compute these quantities, as they are based on full knowledge of the future events in the frame. However, these solutions *exist*, and we will find it is possible to design an algorithm that, over time, yields total utility over the first KT frames (for any $K > 0, T > 0$) that keeps all queues bounded by a constant that is $O(V)$ (where $V > 0$ is a control parameter that affects a utility-congestion tradeoff), with total utility that satisfies:

$$\liminf_{K\to\infty}\sum_{n=1}^{N}[g_n(\overline{y}_n(KT)) - \overline{p}_n(KT)] \geq \liminf_{K\to\infty}\frac{1}{K}\sum_{k=0}^{K-1} util_k^*(T) - BT/V \quad (21)$$

where $\overline{y}_n(KT)$ and $\overline{p}_n(KT)$ are the average throughput delivered to node n and the average penalty incurred by node n over the first KT slots, and B is a constant that is independent of T and V. Thus, the V parameter can be chosen as large as desired to make the deviation BT/V from the target utility arbitrarily small, at the cost of an $O(V)$ increase in worst-case-queue backlog.

The value of $\frac{1}{K}\sum_{k=0}^{K-1} util_k^*(T)$ does *not* represent the maximum time average utility that can be achieved over the first KT slots, because it forces the constraints (19)-(20) to be satisfied on every frame. However, it still represents a meaningful target, as the $util_k^*(T)$ values are obtained from an ideal algorithm that has knowledge of future events in frame k, whereas an actual policy does not have any future knowledge. Further, in ergodic systems it can be shown that:

$$\lim_{T\to\infty}\left[\liminf_{K\to\infty}\frac{1}{K}\sum_{k=0}^{K-1} util_k^*(T)\right] = util^{opt}$$

where $util^{opt}$ is the optimal infinite horizon utility over all algorithms that stabilize the system and meet the tit-for-tat constraints. The intuition for this is that when the frame size T is large, the empirical fraction of time that certain network events occur approaches the ergodic distribution, and so $util_k^*(T)$ is close to $util^{opt}$ for every frame k.

III. The Dynamic Algorithm

To develop an algorithm, we use the technique of Lyapunov optimization [1][13]. To enforce the constraints (12) and (7), for each $n \in \mathcal{N}$ we use virtual queues $H_n(t)$ and $F_n(t)$, respectively, with update equations:

$$H_n(t+1) = \max\left[H_n(t) - \sum_{a=0}^{N} R_{an}(t) + \gamma_n(t), 0\right] \quad (22)$$

$$F_n(t+1) = \max\left[F_n(t) - \beta - \sum_{b=1}^{N} R_{nb}(t) + \alpha\sum_{a=0}^{N} R_{an}(t), 0\right] \quad (23)$$

The intuition behind these virtual queues can be understood in the special case of an ergodic system: If we stabilize these virtual queues, then the time average of their "service variables" must be greater than or equal to the time average of their "arrival variables" so that the desired constraints (12) and (7) are satisfied. For example, by (22) we see that stabilizing $H_n(t)$ ensures $\sum_{a=0}^{N} \overline{r}_{an} \geq \overline{\gamma}_n$. The virtual queues $F_n(t)$ in (23) enforce the tit-for-tat constraints (7). If $\alpha = 0$ and $F_n(0) = 0$, then $F_n(t) = 0$ for all t, which is consistent with the fact that constraints (7) are removed in this case.

Define $\boldsymbol{\Theta}(t) \triangleq [\boldsymbol{Q}(t), \boldsymbol{F}(t), \boldsymbol{H}(t)]$ as the vector of all queues in the system, and define the *Lyapunov function* $L(\boldsymbol{\Theta}(t))$ by:

$$L(\boldsymbol{\Theta}(t)) \triangleq \frac{1}{2}\sum_{n=1}^{N}\sum_{m=0}^{N} Q_{mn}(t)^2 + \frac{1}{2}\sum_{n=1}^{N}[H_n(t)^2 + F_n(t)^2] \quad (24)$$

For each integer $T > 0$, define $\Delta_T(\boldsymbol{\Theta}(t))$ as the T-slot sample-path drift:

$$\Delta_T(\boldsymbol{\Theta}(t)) \triangleq L(\boldsymbol{\Theta}(t+T)) - L(\boldsymbol{\Theta}(t))$$

Using the drift-plus-penalty framework of [1][13], every slot t we observe the current queue states, the current $A_n(t)$ and $\mathcal{N}_n(t)$ information, and the current topology state $S(t)$, and make decisions that "greedily" minimize a bound on the following expression that involves only *1-slot drift*:

$$\Delta_1(\boldsymbol{\Theta}(t)) - V\sum_{n=1}^{N}[g_n(\gamma_n(t)) - p_n(t)]$$

A. The Dynamic Drift-Plus-Penalty Peering Algorithm

By squaring the queue update equations (3), (22), (23) we can compute the following bound on the drift-plus-penalty:

$$\begin{gathered}\Delta_1(\boldsymbol{\Theta}(t)) - V\sum_{n=1}^{N}[g_n(\gamma_n(t)) - p_n(t)] \leq \\ B - V[\sum_{n=1}^{N} g_n(\gamma_n(t)) - \sum_{b=1}^{N}\mu_{nb}(t)\eta_{nb}(S(t))] \\ + \sum_{n=1}^{N}\sum_{m=0}^{N} Q_{mn}(t)[R_{mn}(t) - \mu_{mn}(t)] \\ + \sum_{n=1}^{N} H_n(t)[\gamma_n(t) - \sum_{a=0}^{N} R_{an}(t)] \\ + \sum_{n=1}^{N} F_n(t)[\alpha\sum_{a=0}^{N} R_{an}(t) - \beta - \sum_{b=1}^{N} R_{nb}(t)] \quad (25)\end{gathered}$$

where $B > 0$ is a constant that depends on the values A_n^{max} and μ_{mn}^{max} (we omit the exact computation of B for brevity).

It is easy to see that the following dynamic algorithm observes the queues and network events every slot t and makes control actions that minimize the right-hand-side of the above drift-plus-penalty bound:

- (Auxiliary Variable Update): For each $n \in \mathcal{N}$, choose $\gamma_n(t)$ as follows:

$$\text{Maximize:} \quad Vg_n(\gamma_n(t)) - H_n(t)\gamma_n(t) \quad (26)$$
$$\text{Subject to:} \quad 0 \leq \gamma_n(t) \leq A_n^{max} \quad (27)$$

- (Request Allocation) For each $n \in \mathcal{N}$, observe the value of the following for all $m \in \{\mathcal{G}_n \cap \mathcal{N}_n(t)\}$:

$$-Q_{mn}(t) + H_n(t) + (F_m(t) - \alpha F_n(t)) \quad (28)$$

  If these values are negative for all $m \in \{\mathcal{G}_n \cap \mathcal{N}_n(t)\}$, choose $R_{mn}(t) = 0$ for all m. Else, choose $R_{mn}(t) = A_n(t)$ for the queue $m \in \{\mathcal{G}_n \cap \mathcal{N}_n(t)\}$ with the largest (non-negative) value (breaking ties arbitrarily), and $R_{an}(t) = 0$ for all $a \neq m$.
- (Scheduling) Choose $\boldsymbol{\mu}(t) \in \Gamma_{S(t)}$ to maximize:

$$\sum_{nb}\mu_{nb}(t)[Q_{nb}(t) - V\eta_{nb}(S(t))]$$

  Clearly we can choose $\mu_{nb}(t) = 0$ for any (n, b) pair that multiplies a non-positive weight in the above sum, and so the above can be simplified to sum only over links with positive weights.[1]
- (Queue Updates) Update queues $Q_{mn}(t)$, $H_n(t)$, $F_n(t)$ via (3), (22), (23).

The above algorithm is simple to implement separately at each node n, except possibly for the scheduling decision for $\boldsymbol{\mu}(t)$. This is a classic "max-weight" type decision [30][31][1]. It is easy in the Internet cloud model, or when we use the "no transmission scheduling" model where rates $\mu_{ab}(t)$ are given to us every slot t. However, it can be difficult for wireless networks with interference, although constant-factor approximations can be used [1][13].

B. Incentives

Our tit-for-tat constraints were designed to incentivize participation, and hence we expect these incentives to appear in the algorithm itself. This is indeed the case if we view the queue backlog $F_n(t)$ as determining the *peer reputation* of node n, in that $F_n(t)$ is small when we consistently accept requests from others (see dynamics in (23)). For example, suppose $\beta = 0$, $\alpha > 0$, and node n *only* receives downloads from others and does not transmit anything (such as when it does not report to node 0 that it has a file that can be of use to others). Then $R_{nb}(t) = 0$ for all t and the $F_n(t)$ queue will grow large. The next time another node m considers accepting a request to send data to node n, it sees the weight:

$$-Q_{mn}(t) + H_n(t) + (F_m(t) - \alpha F_n(t))$$

We show in the next section that the queues $H_n(t)$ are bounded, and so the above weight will be negative if $\alpha F_n(t)$ is very large in comparison to $F_m(t)$. Thus, the request allocation will choose $R_{mn}(t) = 0$. If this happens for all other nodes m, then no more requests for delivery to node n will be allocated until node n decreases its $F_n(t)$ value by accepting requests from others. This is made precise in Section III-F.

C. Bounded Queues

Theorem 1: Assume $V > 0$. Under the above dynamic decisions for $\gamma_n(t)$, $R_{mn}(t)$, and under *any* transmission decisions $\boldsymbol{\mu}(t) \in \Gamma_{S(t)}$, we have for all $n \in \mathcal{N}$:

$$H_n(t) \leq V\nu_n + A_n^{max} \triangleq H_n^{max} \quad \forall t$$

provided that these bounds hold at time 0. Further, if the tit-for-tat constraints are removed (so that $\alpha = F_n(t) = 0$ for all t), then for all queues $Q_{mn}(t)$:

$$Q_{mn}(t) \leq V\nu_n + 2A_n^{max}$$

[1]Queue backlog in systems with $\eta_{nb}(S(t)) > 0$ can often be improved via *place-holder packets* [13], although we omit this extension for brevity.

provided this bound holds at time 0.

Proof: We first prove the H_n^{max} bound. Suppose it holds for slot t. We show it also holds for slot $t+1$. Suppose that $H_n(t) \leq V\nu_n$. Then $H_n(t+1) \leq V\nu_n + A_n^{max} \triangleq H_n^{max}$, because it can increase by at most A_n^{max} on any slot (see dynamic update equation (22) and recall that $\gamma_n(t) \leq A_n^{max}$ for all t by (27)). Alternatively, suppose that $H_n(t) > V\nu_n$. Because ν_n is the largest right-derivative of $g_n(y)$, we have for all $0 \leq y \leq A_n^{max}$:

$$g_n(y) \leq g_n(0) + \nu_n y \tag{29}$$

By (29), for any $\gamma_n(t)$ satisfying $0 \leq \gamma_n(t) \leq A_n^{max}$ we have:

$$\begin{aligned} &Vg_n(\gamma_n(t)) - H_n(t)\gamma_n(t) \\ &\quad\leq Vg_n(0) + V\nu_n\gamma_n(t) - H_n(t)\gamma_n(t) \\ &\quad= Vg_n(0) - \gamma_n(t)(H_n(t) - V\nu_n) \\ &\quad\leq Vg_n(0) \end{aligned}$$

with equality only if $\gamma_n(t) = 0$ (because $H_n(t) - V\nu_n > 0$). It follows that on any slot for which $H_n(t) > V\nu_n$, the auxiliary variable decisions (26)-(27) force $\gamma_n(t) = 0$. It follows by (22) that $H_n(t)$ cannot increase on the next slot, and so we have $H_n(t+1) \leq H_n(t) \leq H_n^{max}$. This proves the H_n^{max} bound.

To prove that $Q_{mn}(t) \leq H_n^{max} + A_n^{max}$ when the tit-for-tat constraints are removed, fix slot t and suppose it holds on this slot. If $Q_{mn}(t) \leq H_n^{max}$, then $Q_{mn}(t+1) \leq H_n^{max} + A_n^{max}$ because it can increase by at most A_n^{max} on any slot t. Alternatively, if $Q_{mn}(t) > H_n^{max}$, then the weight (28) for the request allocation algorithm is negative (recall that $F_m(t) = 0$), and so $R_{mn}(t) = 0$. Thus, $Q_{mn}(t)$ cannot increase on the next slot, so $Q_{mn}(t+1) \leq Q_{mn}(t) \leq H_n^{max} + A_n^{max}$. □

Theorem 2: Suppose we use the algorithm of Section III-A on a system with the general tit-for-tat constraints. Then all queues $H_n(t)$, $Q_{mn}(t)$, $F_n(t)$ are rate stable, in that:

$$\lim_{t\to\infty} \frac{H_n(t)}{t} = \lim_{t\to\infty} \frac{Q_{mn}(t)}{t} = \lim_{t\to\infty} \frac{F_n(t)}{t} = 0 \tag{30}$$

Further, if $\beta > 0$ and if there is an $\epsilon > 0$ such that every slot t there is a matrix $(\mu_{ab}^*(t)) \in Conv(\Gamma_{S(t)})$ such that $\mu_{ab}^*(t) \geq \epsilon$ for all (a,b) for which there are queues $Q_{ab}(t)$, then all queues $H_n(t)$, $Q_{mn}(t)$, $F_n(t)$ are deterministically bounded by a finite constant.

The proof of Theorem 2 is given in Section III-E.

D. Utility Performance

Here we prove the utility bound (21). We assume that, in addition to choosing $R_{mn}(\tau)$, $\gamma_n(\tau)$ decisions exactly according to the dynamic algorithm of Section III-A, we also choose $\boldsymbol{\mu}(\tau) \in \Gamma_{S(\tau)}$ according to the full max-weight solution. While we can also obtain results that scale gracefully with the factor θ we are away from the max-weight decision (as in [1][13]), we omit that analysis for brevity. To simplify the proof, we also assume that all initial queue backlogs are 0, so that $\boldsymbol{\Theta}(0) = \mathbf{0}$.

First note that the algorithm chooses decision variables to maximize the right-hand-side of the drift-plus-penalty bound (25) over all alternative decisions, including *convexified decisions* $R_{mn}^*(\tau)$, $\boldsymbol{\mu}^*(\tau)$ that extend their option space to the convexified regions $[0, A_n^{max}]$ and $Conv(\Gamma_{S(\tau)})$. This is because these decision variables appear *linearly* in the right-hand-side of (25). We thus have for any slot τ:

$$\begin{aligned} &\Delta_1(\boldsymbol{\Theta}(\tau)) - V\textstyle\sum_{n=1}^N [g_n(\gamma_n(\tau)) - p_n(\tau)] \leq \\ &B - V\textstyle\sum_{n=1}^N [g_n(\gamma_n^*) - \sum_{b=1}^N \mu_{nb}^*(\tau)\eta_{nb}(S(\tau))] \\ &+ \textstyle\sum_{n=1}^N \sum_{m=0}^N Q_{mn}(\tau)[R_{mn}^*(\tau) - \mu_{mn}^*(\tau)] \\ &+ \textstyle\sum_{n=1}^N H_n(\tau)[\gamma_n^* - \sum_{a=0}^N R_{an}^*(\tau)] \\ &+ \textstyle\sum_{n=1}^N F_n(\tau)[\alpha\sum_{a=0}^N R_{an}^*(\tau) - \beta - \sum_{b=1}^N R_{nb}^*(\tau)] \end{aligned} \tag{31}$$

where γ_n^*, $\boldsymbol{\mu}^*(\tau)$, $R_{mn}^*(\tau)$ are any alternative decisions that can be made on slot τ that take place within the convexified decision space. Now fix integers $K > 0$ and $T > 0$, and fix $k \in \{0, \ldots, K-1\}$. Summing (31) over $\tau \in \{kT, \ldots, kT + T - 1\}$, we can show that:

$$\begin{aligned} &\Delta_T(\boldsymbol{\Theta}(kT)) - V\sum_{\tau=kT}^{kT+T-1}\sum_{n=1}^N [g_n(\gamma_n(\tau)) - p_n(\tau)] \leq \\ &BT^2 - V\sum_{n=1}^N [Tg_n(\gamma_n^*) - \sum_{\tau=kT}^{kT+T-1}\sum_{b=1}^N \mu_{nb}^*(\tau)\eta_{nb}(S(\tau))] \\ &+ \sum_{n=1}^N\sum_{m=0}^N Q_{mn}(kT)\sum_{\tau=kT}^{kT+T-1}[R_{mn}^*(\tau) - \mu_{mn}^*(\tau)] \\ &+ \sum_{n=1}^N H_n(kT)\sum_{\tau=kT}^{kT+T-1}[\gamma_n^* - \sum_{a=0}^N R_{an}^*(\tau)] \\ &+ \sum_{n=1}^N F_n(kT)\sum_{\tau=kT}^{kT+T-1}[\alpha\sum_{a=0}^N R_{an}^*(\tau) - \beta - \sum_{b=1}^N R_{nb}^*(\tau)] \end{aligned}$$

While we omit the full derivation of the above inequality, we note that its left-hand-side follows from the fact that:

$$\begin{aligned} \sum_{\tau=kT}^{kT+T-1} \Delta_1(\boldsymbol{\Theta}(\tau)) &= \sum_{\tau=kT}^{kT+T-1} [L(\boldsymbol{\Theta}(\tau+1)) - L(\boldsymbol{\Theta}(\tau))] \\ &= L(\boldsymbol{\Theta}(kT+T)) - L(\boldsymbol{\Theta}(kT)) \\ &\triangleq \Delta_T(\boldsymbol{\Theta}(kT)) \end{aligned}$$

and its right-hand-side follows from the fact that we sum over T slots (which changes the B constant to BT) and the queue backlogs do not change by more than a bounded amount on any slot τ (which adds a term $BT(T-1)$), so that $BT + BT(T-1) = BT^2$. We now plug the alternative decisions γ_n^*, $\boldsymbol{\mu}^*(\tau)$, $R_{mn}^*(\tau)$ that solve the T-slot lookahead problem (13)-(20) to achieve utility $util_k^*(T)$. This gives significant cancellation and yields:

$$\begin{aligned} &\Delta_T(\boldsymbol{\Theta}(kT)) - V\sum_{\tau=kT}^{kT+T-1}\sum_{n=1}^N [g_n(\gamma_n(\tau)) - p_n(\tau)] \leq \\ &BT^2 - VT util_k^*(T) \end{aligned} \tag{32}$$

Summing the above over $k \in \{0, \ldots, K-1\}$ and dividing by VKT yields:

$$\begin{aligned} &\frac{L(\boldsymbol{\Theta}(KT)) - L(\boldsymbol{\Theta}(0))}{VKT} - \frac{1}{KT}\sum_{\tau=0}^{KT-1}\sum_{n=1}^N [g(\gamma_n(\tau)) - p_n(\tau)] \\ &\leq \frac{BT}{V} - \frac{1}{K}\sum_{k=0}^{K-1} util_k^*(T) \end{aligned}$$

Rearranging terms in the above, noting that $L(\mathbf{\Theta}(KT)) \geq 0$ and that $L(\mathbf{\Theta}(0)) = 0$, we have:

$$\sum_{n=1}^{N}[g_n(\overline{\gamma}_n(KT)) - \overline{p}_n(KT)] \geq \frac{1}{K}\sum_{k=0}^{K=1} util_k^*(T) - \frac{BT}{V} \quad (33)$$

where we define $\overline{\gamma}_n(KT)$ and $\overline{p}_n(KT)$ as time averages over the first KT slots, and we have used Jensen's inequality in the concave function $g_n(\cdot)$.

Now note by (22) that:

$$H_n(\tau+1) \geq H_n(\tau) - \sum_{a=0}^{N} R_{an}(\tau) + \gamma_n(\tau)$$

Hence, summing the above over $\tau \in \{0, \ldots, KT-1\}$ gives:

$$H_n(KT) - H_n(0) \geq \sum_{\tau=0}^{KT-1}[-\sum_{a=0}^{N} R_{an}(\tau) + \gamma_n(\tau)]$$

Dividing by KT, rearranging terms, and using $H_n(0) = 0$ yields:

$$\overline{y}_n(KT) \geq \overline{\gamma}_n(KT) - \frac{H_n(KT)}{KT} \geq \overline{\gamma}_n(KT) - \frac{H_n^{max}}{KT}$$

where we have defined $\overline{y}_n(KT)$ by:

$$\overline{y}_n(KT) \triangleq \frac{1}{KT}\sum_{\tau=0}^{KT-1}\sum_{a=0}^{N} R_{an}(\tau)$$

Further, because the maximum derivative of $g_n(\gamma_n)$ is ν_n, we have:

$$g_n(\overline{y}_n(KT)) \geq g_n(\overline{\gamma}_n(KT)) - \frac{\nu_n H_n^{max}}{KT}$$

Plugging the above into (33) yields:

$$\sum_{n=1}^{N}[g_n(\overline{y}_n(KT)) - \overline{p}_n(KT)] \geq \frac{1}{K}\sum_{k=0}^{K=1} util_k^*(T) - \frac{BT}{V} - \sum_{n=1}^{N}\frac{\nu_n H_n^{max}}{KT} \quad (34)$$

Taking a limit of the above as $K \to \infty$ proves (21).

We note that the dynamic algorithm itself does not use a T parameter, and hence the above bound (34) can be viewed as a class of bounds that hold for all integers $T > 0$ (and all integers $K > 0$). The value of $util_k^*(T)$ typically *improves* if we use a large value of T, as that allows a larger future lookahead. However, using a large value of T requires a larger value of V to make the BT/V term negligible.

E. Proof of Theorem 2

Note that the drift bound (32) can be rewritten for the case $T = 1$ to show:

$$\Delta_1(\mathbf{\Theta}(kT)) \leq D$$

for some finite constant D that depends on V and that bounds the utility values that can be achieved. This drift condition for the quadratic Lyapunov function implies that all queues are rate stable [13], which proves (30) from Theorem 2. If the assumptions of the second part of Theorem 2 hold, then plugging into the right-hand-side of (31) yields:

$$\Delta_1(\mathbf{\Theta}(\tau)) \leq C - \sum_{n=1}^{N}\sum_{m=0}^{N} Q_{mn}(\tau)\epsilon - \sum_{n=1}^{N} F_n(\tau)\beta$$

for some finite constant C. This, together with boundedness of the $H_n(t)$ queues (known from Theorem 1), can be used to show all queues are bounded (see [32] for a related proof).

F. Satisfaction of the Tit-For-Tat Constraints

From Theorem 2 we know that $F_n(t)/t \to 0$. On the other hand, by (23) we have for each $n \in \mathcal{N}$:

$$F_n(\tau+1) \geq F_n(\tau) - \beta - \sum_{b=1}^{N} R_{nb}(\tau) + \alpha\sum_{a=0}^{N} R_{an}(\tau)$$

Summing the above over $\tau \in \{0, \ldots, t-1\}$ (for any integer $t > 0$) and dividing by t yields:

$$\frac{F_n(t) - F_n(0)}{t} \geq -\beta - \sum_{b=1}^{N}\overline{r}_{nb}(t) + \alpha\sum_{a=1}^{N}\overline{r}_{an}(t)$$

where $\overline{r}_{ab}(t)$ is the time average of $r_{ab}(\tau)$ over the first t slots. Rearranging terms, taking a $\limsup$, and using rate stability of $F_n(t)$ yields:

$$\limsup_{t\to\infty}\left[\alpha\sum_{a=1}^{N}\overline{r}_{an}(t) - \sum_{b=1}^{N}\overline{r}_{nb}(t)\right] \leq \beta$$

The above holds for all $n \in \mathcal{N}$, and hence the tit-for-tat constraints are satisfied.

IV. Modifications for Worst Case Delay

Suppose we remove the tit-for-tat constraints. Theorem 1 ensures that $Q_{mn}(t) \leq Q_{mn}^{max} \triangleq V\nu_n + 2A_n^{max}$ for all m, n and all t. However, the max-weight scheduling may not ensure persistent service, and thus does not provide a worst-case delay bound. This can be remedied using an *ϵ-persistent service queue* as in [33], provided that we also introduce additional packet drops at the request queues. This section briefly outlines the necessary modifications.

The queue update is modified to (compare with (3)):

$$Q_{mn}(t+1) = \max[Q_{mn}(t) + R_{mn}(t) - \mu_{mn}(t) - d_{mn}(t), 0] \quad (35)$$

where $d_{mn}(t)$ are decision variables representing the amount of packets dropped from the $Q_{mn}(t)$ queue on slot t. We must ensure these additional drops appropriately reduce throughput, so we define:

$$y_n(t) \triangleq \sum_{a=0}^{N} R_{an}(t) - \sum_{a=0}^{N} d_{an}(t) \quad (36)$$

and note that $\overline{y}_n$, the time average of $y_n(t)$, is indeed the throughput delivered to node n. The auxiliary variable constraints are then $\overline{y}_n \geq \overline{\gamma}_n$ for all $n \in \mathcal{N}$, so that virtual queues $H_n(t)$ are modified to (compare with (22)):

$$H_n(t+1) = \max[H_n(t) - y_n(t) + \gamma_n(t), 0] \quad (37)$$

For given constants $\epsilon_{mn} > 0$, we introduce *ϵ-persistent service queues* $Z_{mn}(t)$, as in [33]:

$$Z_{mn}(t+1) = \max[Z_{mn}(t) + 1_{\{Q_{mn}(t)>0\}}(\epsilon_{mn} - \mu_{mn}(t)) - d_{mn}(t) - 1_{\{Q_{mn}(t)=0\}}\mu_{mn}^{max}, 0] \quad (38)$$

When $Q_{mn}(t) > 0$, this virtual queue has the same service process as $Q_{mn}(t)$, but has a constant arrival of size ϵ_{mn}. It is shown in [33] that if our algorithm yields finite upper bounds Q_{mn}^{max}, Z_{mn}^{max} on $Q_{mn}(t)$ and $Z_{mn}(t)$, then the worst-case delay of all non-dropped packets in queue $Q_{mn}(t)$ is $\lceil (Z_{mn}^{max} + Q_{mn}^{max})/\epsilon_{mn} \rceil$, where $\lceil x \rceil$ represents the smallest integer that is greater than or equal to x.

The following modified algorithm can indeed be shown to yield finite upper bounds for $Q_{mn}(t)$ and $Z_{mn}(t)$:

- (Auxiliary Variable Update): For each $n \in \mathcal{N}$, choose $\gamma_n(t)$ as follows:

$$\begin{aligned} \text{Maximize:} \quad & Vg_n(\gamma_n(t)) - H_n(t)\gamma_n(t) \\ \text{Subject to:} \quad & 0 \leq \gamma_n(t) \leq A_n^{max} \end{aligned}$$

- (Request Allocation) Same as before.
- (Scheduling) Choose $\boldsymbol{\mu}(t) \in \Gamma_{S(t)}$ to maximize:

$$\textstyle\sum_{nb} \mu_{nb}(t)[Q_{nb}(t) + Z_{nb}(t)1_{\{Q_{nb}(t)>0\}} - V\eta_{nb}(S(t))]$$

- (Dropping) Choose $d_{mn}(t)$ to solve:

$$d_{mn}(t) = \begin{cases} 0 & \text{if } Z_{mn}(t) + Q_{mn}(t) \leq H_n(t) \\ A_n^{max} & \text{otherwise} \end{cases}$$

- (Queue Updates) Update queues $Q_{mn}(t)$, $Z_{mn}(t)$, $H_n(t)$, via (35), (38), (37).

Using a similar drift-plus-penalty Lyapunov optimization analysis, it can be shown to yield utility that is close to that of a T-slot lookahead utility that is the same as (13)-(20), with the exception that the constraint (14) is replaced by $\gamma_n = \frac{1}{T}\sum_{\tau=kT}^{kT+T-1}\sum_{a=0}^{N}[R_{an}(\tau) - d_{an}]$ for drop variables d_{an} in the range $0 \leq d_{an} \leq A_n^{max}$, and an additional ϵ-*persistent constraint* is included for all (a, b) for which there are queues $Z_{ab}(t)$:

$$\textstyle\frac{1}{T}\sum_{\tau=kT}^{kT+T-1}[\mu_{ab}(\tau) + d_{ab}] \geq \epsilon_{ab}$$

V. Conclusions

This paper considers utility maximization for a peer-to-peer network model, and extends Lyapunov optimization theory to this context. We developed an algorithm that makes greedy decisions every slot and, for any $T > 0$, yields time average utility that can approach the utility of a "genie-aided" T-slot lookahead policy that has knowledge of the future over successive frames of T slots. The algorithm ensures that "tit-for-tat" constraints are satisfied, which incentivizes participation in the peer-to-peer network. Indeed, it was seen that the peering scheme requires nodes to help others in order to improve the "weights" they need to continue receiving downloads. This is perhaps the first use of max-weight and Lyapunov optimization principles in a peer-to-peer system.

References

[1] 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.

[2] K. Doppler, M. Rinne, C. Wijting, C. B. Ribeiro, and K. Hugl. Device-to-device communication as an underlay to lte-advanced networks. *IEEE Comm Mag*, pp. 42-49, Dec. 2009.

[3] A. F. Molisch, Z. Tao, P. V. Orlik, J. Zhang, and T. Kuze. Enhanced mbs with harq. *C802.16m-08/1000*, Sept. 2008.

[4] 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.

[5] 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.

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

[7] 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.

[8] B. Fan, D.-M. Chiu, and J. C.S. Lui. The delicate tradeoffs in bittorrent-like file sharing protocol design. In *The 14th IEEE International Conference on Network Protocols (ICNP)*, Santa Barbara, California, November 12-15 2006.

[9] 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.

[10] 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.

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

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

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

[14] M. J. Neely. *Dynamic Power Allocation and Routing for Satellite and Wireless Networks with Time Varying Channels*. PhD thesis, Massachusetts Institute of Technology, LIDS, 2003.

[15] M. J. Neely, E. Modiano, and C. Li. Fairness and optimal stochastic control for heterogeneous networks. *Proc. IEEE INFOCOM*, March 2005.

[16] M. J. Neely. Universal scheduling for networks with arbitrary traffic, channels, and mobility. *Proc. IEEE Conf. on Decision and Control (CDC)*, Atlanta, GA, Dec. 2010.

[17] J. Andrews, S. Shakkottai, R. Heath, N. Jindal, M. Haenggi, R. Berry, D. Guo, M. Neely, S. Weber, S. Jafar, and A. Yener. Rethinking information theory for mobile ad hoc networks. *IEEE Communications Magazine*, vol. 46, no. 12, pp. 94-101, Dec. 2008.

[18] Bram Cohen. Incentives build robustness in bittorrent. In *P2PECON*, Berkeley, CA, Jun 2003.

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

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

[21] Y. Yang, A. L.-H. Chow, and L. Golubchik. Multi-torrent: A performance study. In *International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (IEEE MASCOTS)*, Baltimore, MD, 2008.

[22] F.P. Kelly, A.Maulloo, and D. Tan. Rate control for communication networks: Shadow prices, proportional fairness, and stability. *Journ. of the Operational Res. Society*, vol. 49, no. 3, pp. 237-252, March 1998.

[23] S. H. Low and D. E. Lapsley. Optimization flow control, i: Basic algorithm and convergence. *IEEE/ACM Transactions on Networking*, vol. 7 no. 6, pp. 861-875, Dec. 1999.

[24] M. Chiang. Balancing transport and physical layer in wireless multihop networks: Jointly optimal congestion control and power control. *IEEE J. on Selected Areas in Comm.*, vol. 23, no. 1, pp. 104-116, Jan. 2005.

[25] R. Agrawal and V. Subramanian. Optimality of certain channel aware scheduling policies. *Proc. 40th Annual Allerton Conference on Communication , Control, and Computing, Monticello, IL*, Oct. 2002.

[26] H. Kushner and P. Whiting. Asymptotic properties of proportional-fair sharing algorithms. *Proc. of 40th Annual Allerton Conf. on Communication, Control, and Computing*, 2002.

[27] X. Lin and N. B. Shroff. Joint rate control and scheduling in multihop wireless networks. *Proc. of 43rd IEEE Conf. on Decision and Control, Paradise Island, Bahamas*, Dec. 2004.

[28] A. Eryilmaz and R. Srikant. Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control. *Proc. IEEE INFOCOM*, March 2005.

[29] A. Stolyar. Maximizing queueing network utility subject to stability: Greedy primal-dual algorithm. *Queueing Systems*, vol. 50, no. 4, pp. 401-457, 2005.

[30] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transacations on Automatic Control*, vol. 37, no. 12, pp. 1936-1948, Dec. 1992.

[31] L. Tassiulas and A. Ephremides. Dynamic server allocation to parallel queues with randomly varying connectivity. *IEEE Transactions on Information Theory*, vol. 39, no. 2, pp. 466-478, March 1993.

[32] M. J. Neely. Distributed and secure computation of convex programs over a network of connected processors. *DCDIS Conf., Guelph, Ontario*, July 2005.

[33] M. J. Neely. Opportunistic scheduling with worst case delay guarantees in single and multi-hop networks. *Proc. IEEE INFOCOM*, 2011.