Client and Server Games in Peer-to-peer Networks

Iordanis Koutsopoulos Leandros Tassiulas Lazaros Gkatzikis
Department of Computer and Communications Engineering , CERTH
University of Thessaly, Volos, Greece

***Abstract*—We consider a content sharing network of non-cooperative peers. The strategy set of each peer comprises, (i) *client* strategies, namely feasible request load splits to servers, and (ii) *server* strategies, namely scheduling disciplines on requests. First, we consider the request load splitting game for given server strategies such as First-In-First-Out or given absolute priority policies. A peer splits its request load to servers to optimize its performance objective. We consider the class of best response load splitting policies residing between the following extremes: a truly selfish, or *egotistic* one, where a peer optimizes its own delay, and a pseudo-selfish or *altruistic* one, where a peer also considers incurred delays to others. We derive conditions for *Nash equilibrium points (NEPs)* and discuss convergence to NEP and properties of the NEP. For both the egotistic cases, the NEP is unique. For the altruistic case, each of the multiple NEPs is an optimum, a global one for the FIFO case and a local one otherwise.**

Next, we include scheduling in peer strategies. With its scheduling discipline, a peer cannot directly affect its delay, but it can affect the NEP *after* peers play the load splitting game. The idea is that peer i should offer high priority to (and thus attract traffic from) higher-priority peers that cause large delay to i at other servers. We devise two-stage game models, where, at a first stage, a peer selects a scheduling rule in terms of a convex combination of absolute priorities, and subsequently peers play the load splitting game. In the most sophisticated rule, a peer selects a scheduling discipline that minimizes its delay at equilibrium, after peers play the load splitting game. We also suggest various heuristics for picking the scheduling discipline. Our models and results capture the dual client-server peer role and aim at quantifying the impact of selfish peer interaction on equilibria.

I. Introduction

Peer-to-peer networks are popular for file sharing (e.g, eDonkey, BitTorrent), video and media distribution (e.g Joost), VoIP (Skype), distributed directory services, web caching, streaming media, grid-based computing, and recently, distributed storage and social networks. We consider a flat architecture of peers that spontaneously interact, and we adhere to a content sharing scenario to demonstrate the intuition of peer interactions. Each peer operates simultaneously as server and client. That is, it serves others' content requests, while having its own requests served by others. As client, it generates a load of content requests. Each request is followed by a advertisement about peers with that content. In BitTorrent, this is accomplished by the tracker that returns a random set of peers that have (part of) the object. For each request, the client peer decides on the peers from which it will request content. In advanced systems, a client may download different parts of an object from different peers. Thus, viewed in long-term fashion, a client determines the portion of its request load to route to each server. On the other hand, a server peer serves received requests based on a scheduling discipline. A meaningful performance metric is average retrieval delay of requests, namely the average time elapsed between a request release and its satisfaction.

In structured systems, one defines a *social objective* reflecting system operation, such as minimization of total average delay. However, peer-to-peer networks may have no operator and usually lack global coordination towards the global social objective. Peers do not know utilities of others, and they autonomously and spontaneously interact with each other. Thus, they selfishly decide on their controls (strategies) to pursue their individual performance objective–in our case, minimize average retrieval delay. The strategy set of each peer consists of (i) *client* strategies, namely the set of request load splits to servers and (ii) *server* strategies, namely the set of scheduling disciplines. By adopting a certain client - server strategy, a peer affects delays of others. These in turn need to respond best and readjust their strategies to improve their own delay, and so on. In such sequences of non-cooperative peer interactions, it makes sense to study resulting stable network operating points, namely peer strategies from which unilateral peer deviation does not improve performance. These operating points are the *Nash equilibrium points (NEPs)* of the non-cooperative game. The social objective can serve as a benchmark, as it quantifies the performance loss of selfish interaction compared to the social optimum. In this work, we take a first step towards understanding best response request load splitting *and* scheduling strategies, and their impact on average retrieval delay and on properties of NEPs.

A. Related work

Assuming that a server employs a processor sharing scheduling policy, modeling and performance evaluation of peer-to-peer networks can be done with queueing theory tools. In [1], the authors consider minimizing total weighted delay for a system of customers and queues with non-preemptive priority scheduling. The problem consists of finding an allocation of customer traffic portions to queues and a customer sequencing order at each queue. For the latter problem, a static priority discipline in line with the $c\mu$ rule is optimal. The authors proceed to convex relaxations to solve this non-convex problem, and the optimal solution cannot be guaranteed. A similar problem with FIFO scheduling is studied in [2], where the major finding is that at the optimal solution, the customer types allocated to each server are clustered in terms of first and second moment of service times. In [3], the authors study peer selection for downloading and streaming content to minimize required time for download and monetary cost for downloading, assuming that bandwidth prices are announced. In [4] the authors study

978-1-4244-3876-1/09/$25.00 ©2009 IEEE

unilateral or bilateral network formation games, where peers individually open up connections on multiple paths or mutually agree on setting up paths respectively.

The problem of request load splitting from clients to servers parallels that of *selfish routing* of flow by *atomic* users over shared links, each associated with a latency function. In the seminal work [5],the authors establish uniqueness of NEP for atomic users over a set of parallel links from a source to a destination for some classes of latency functions with certain convexity properties, among which is the average delay per unit of flow for an $M/M/1$ queue with FIFO service. However, convergence of best response updates to the NEP is formally proved only for the case of two users and two links. Results on uniqueness of the NEP and means for converging to it for cost functions that are linear in link flow appear in [6]. The existence of a NEP with certain efficiency properties is shown in [7] in a repeated game routing framework. An important line of works study the engineering of efficient NEPs by controls other than player strategies, which emerge in the network design or run-time phase. The works [8], [9] respectively consider allocating additional capacity to links and controlling some amount of flow through a central manager in the model of [5].

The worst case ratio between the value of the social objective at a NEP and that at social optimum is termed price of anarchy [10]. This is studied in [11] and [12] for routing games of non-atomic players. The price of anarchy is at most $4/3$ for linear latency functions regardless of network topology. More recently, [13] shows that the price of anarchy for non-atomic users in a network of N parallel links and the class of unbounded latency functions above is N.

B. Our contribution

We start from selfish routing over parallel links to model selfish best response client - server peer interactions. A link stands for a server. Request load splitting by clients to servers is mapped to flow splitting over parallel links. A scheduling discipline of server j assigns different priorities to clients and thus affects the latency functions of client peers on link j. First, we fix the scheduling disciplines to be First-In-First-Out (FIFO) or given absolute priority scheduling policies. Each peer splits its request load to servers to optimize its own performance objective. We introduce the class of load splitting policies that reside between the following extremes: a truly selfish or *egotistic* one, where each peer optimizes only its own delay, and (ii) a *pseudo-selfish* or *altruistic* flavored one, where each peer optimizes a function that includes incurred delays to others, in addition to its own delay. We give conditions for NEPs and expressions for best response updates. For the egotistic case, the NEP is always unique. For the altruistic case under FIFO scheduling, there exist multiple NEPs, and the sequence of best response updates converges to a NEP which is also a global optimum. For equal service capacities, the server load vector at NEP is most balanced and lexicographically minimal. Interestingly, for priority scheduling policies we observe convergence of altruistic best response to one of the multiple NEPs.

Next, we include scheduling in peer strategies. By its scheduling policy, a peer cannot directly affect its delay, yet it can affect the equilibrium operating point *after* peers play the load splitting game. The idea is that peer i should offer high priority to (and thus attract traffic from) higher-priority peers that cause large delay to i at other servers. If i knows that j is served with higher priority at a server k, it may decide to serve j with high priority and thus lower delay. By advertising lower delays to j, peer i may attract some traffic of j from server k and thus improve its own delay at k. We devise two-stage games, where, at a first stage, a peer selects a scheduling discipline in terms of a convex combination of absolute priorities, and in a second stage peers play the load splitting game. The most sophisticated but computationally demanding rule for selecting a scheduling discipline is the minimization of delay at NEP after peers play the load splitting game. This leads to the best choice available to i at the moment, although future choices by others may alter the situation. We suggest various heuristics for selecting a scheduling discipline. Our models capture the dual client-server role and aim at quantifying the impact of selfish peer interaction on equilibrial operating points. In section II we present the model and assumptions. In section III we describe the client load splitting games for given scheduling disciplines and in section IV we devise games where scheduling is also part of peer strategies. Section V includes numerical results and section VI concludes our study.

II. System model

We consider a set $\mathcal{N}$ of N non-cooperative peers. No subset of peers coordinate their requests and scheduling decisions or negotiate, and they also do not conform to any incentive protocol. Each peer has some content, which it may share with others, while at the same time it requests content from others. That is, each peer acts simultaneously as client and server. In order to study the convergence properties of our approaches and the resulting equilibria we need a steady or at least a slowly varying environment. Thus, we assume that the system is closed and no peer arrivals or departures occur. However, most of the results presented in this paper hold for dynamic environments as well. The N peers engage in content exchange interactions. To show the intuition behind our models, we do not discriminate peers on the basis of content they have. Instead, we assume that requested content by a peer is always available at others. Thus, the request load splitting decision of a peer does not depend on whether peers have the content. This is the general case in every chunk-based peer-to-peer network. Apart from an initial short phase, where only a few seeds possess the content, every peer has some content that the others will be interested in (see BitTorrent). Besides, in our model this assumption could easily get discarded by introducing a $N \times N$ connectivity matrix, indicating whether peer i is interested in some of j's content.

The N peers are connected with logical links in an overlay graph. An overlay link (i, j) from i to j consists of a path of physical links from i to j. Hence, the retrieval delay of i from j would depend on congestion conditions and associated control actions along the underlying physical path. We do not consider

such scenarios, and we assume that overlay network formation is not an issue. Equivalently, this implies large link bandwidth on intermediate links between peers, such that the effect of intermediate congestion on retrieval delay is negligible; thus, only peer access link capacities come into stage.

A. Peers as clients: Request Load Splitting Strategies

Each peer i has a content request load measured in bits/sec that denotes request generation rate from higher layer applications. Each request may correspond to a distinct object. For modeling and analytical tractability, the request load generation process from peer i follows Poisson distribution with mean r_i requests/sec. The size of a requested by peer i object in bits is an exponential random variable with mean L_i and the aggregate average request rate (in bits/sec) of peer i is $r_i L_i$. Each peer j has service capacity C_j bits/sec which denotes service rate. The average request service time of client i at server j, in sec per request, is $1/\mu_{ij} = L_i/C_j$. We assume equal $L_i = 1$ size unit for all i (as in BitTorrent), yet our approach and expressions can be modified to capture the case of different average request sizes as well.

A *feasible request load splitting strategy* for client i is a vector $\boldsymbol{\lambda}_i = (\lambda_{ij} : j = 1, \ldots, N)$, with $\sum_{j=1}^{N} \lambda_{ij} = r_i$, where $\lambda_{ij} \geq 0$ is the amount of requested load of peer i from server j, and $\lambda_{ii} = 0$. Denote by $\mathcal{F}_i$ the set of feasible load splitting strategies of peer i. The fraction of requests of i addressed to peer j is λ_{ij}/r_i. Note that this also captures the scenario where a peer obtains portions of the objects from several peers. Denote by $\boldsymbol{\lambda}_{-i} = (\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_{i-1}, \boldsymbol{\lambda}_{i+1}, \ldots, \boldsymbol{\lambda}_N)$ a load splitting strategy profile of peers other than i. A feasible *network* request load splitting strategy is a matrix $\boldsymbol{\Lambda}$ with the ith row being $\boldsymbol{\lambda}_i$ and diagonal elements zero. Let $\mathcal{F}$ be the set of feasible network load splitting strategies, each of which is also assumed to satisfy the necessary and sufficient conditions for network stability, namely $\sum_{i=1}^{N} \lambda_{ij} < C_j$ for $j = 1, \ldots, N$. The system is stable if $r_i < \sum_{j=1}^{N} C_j$ for all $i = 1, \ldots, N$ and $\sum_{i=1}^{N} r_i < \sum_{j=1}^{N} C_j$, which are also assumed to hold.

B. Peers as servers : Request Scheduling Strategies

Consider a given request load splitting strategy $\boldsymbol{\Lambda} = (\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_N)$. Let $\mathcal{N}_j = \{i \in \mathcal{N} : \lambda_{ij} > 0\}$ be the set of peers from which server j receives requests. Let $\Lambda_j = \sum_{i \in \mathcal{N}_j} \lambda_{ij}$ be the total request load at server j and let Λ_j^{-i} be the load at server j from peers other than i so that $\Lambda_j = \Lambda_j^{-i} + \lambda_{ij}$. For exponentially distributed request size, service times are also exponentially distributed. Thus, each server j can be modeled as an $M/M/1$ queue with service rate C_j that serves incoming requests of rate λ_{ij}. The average retrieval delay per unit of traffic, D_{ij} of peer i when serviced by peer j is the average request queueing delay per unit of traffic at j, Q_{ij} plus $1/C_j$, the average service delay. The scheduling discipline applied by server peer j affects Q_{ij}.

1) FIFO Scheduling: For FIFO scheduling policy, the same average retrieval delay is provided to all clients j, i.e.

$$Q_{ij} = \frac{\Lambda_j}{C_j(C_j - \Lambda_j)} \ , \ D_{ij} = \frac{1}{C_j - \Lambda_j}. \tag{1}$$

2) Preemptive Priority Scheduling: A server that employs a preemptive-resume priority service discipline with absolute priorities [14], serves incoming requests based on certain priority ordering. The service of a peer is interrupted whenever a request from a peer of higher priority arrives, and it is resumed from the point it was interrupted when all higher priority requests peers have been served. Besides being a reasonable service model, this policy can be good approximation for non-preemptive priority scheduling policies in high load conditions with regard to achievable delays.

An absolute *priority scheduling policy* $\boldsymbol{\pi}_j$ at server j is a permutation of set $\mathcal{N}_j$. Let $\mathcal{M}_j$ be the set of $|\mathcal{N}_j|!$ priority orders of peers in $\mathcal{N}_j$. For a given permutation at j, let π_j^i denote the ranking of peer i. We denote the fact that a peer i is served at j with higher priority than peer ℓ with $\pi_j^i < \pi_j^\ell$. For client i and server j and given priority scheduling policy, let Λ_j^{i+} be the load that is served at server j with higher priority than i, namely $\Lambda_j^{i+} = \sum_{k:\pi_j^k < \pi_j^i} \lambda_{kj}$. Let $\boldsymbol{\pi}$ be the ensemble of scheduling disciplines of all peers and $\boldsymbol{\pi}_{-j}$ be the scheduling disciplines of peers other than j. A certain permutation corresponds to a delay vector $\mathbf{D} = (D_{1j}, \ldots, D_{Nj})$, where D_{ij} is the average delay per unit of flow that peer i experiences in peer j and depends on the priority π_j^i that peer i enjoys in server j in the scheduling policy[15], and

$$D_{ij} = \begin{cases} \dfrac{1}{C_j - \lambda_{ij}}, & \text{if } \pi_i^j = 1, \\ \dfrac{C_j}{(C_j - \Lambda_j^{i+})(C_j - \Lambda_j^{i+} - \lambda_{ij})}, & \text{if } \pi_i^j > 1. \end{cases} \tag{2}$$

Note that D_{ij} depends on λ_{ij} and on load splits of other peers with higher priority than i at server j but not on loads of lower priority peers. Also, note that the expression for D_{ij} for $\pi_i^j = 1$ emerges from that for $\pi_i^j > 1$ for $\Lambda_j^{i+} = 0$.

A scheduling policy may involve mixtures of different absolute priorities. Let $\mathcal{C}_i$ be the convex hull of the set of delay vectors $\mathbf{D}_m$, each of which corresponds to a specific absolute priority order m, and let $\mathcal{C}_i^*$ be the set of achievable delay vectors by server i for all work-conserving scheduling policies. Then, $\mathcal{C}_i^* = \mathcal{C}_i$ [16]. That is, for any delay vector $\mathbf{D}$ corresponding to a work-conserving policy there exist positive reals $\{a_i^m\}$, $m = 1, \ldots, \mathcal{M}_i$ such that $\mathbf{D} = \sum_{m=1}^{\mathcal{M}_i} a_i^m \mathbf{D}_m$, with $\sum_{m=1}^{\mathcal{M}_i} a_i^m = 1$, i.e, any delay vector is achievable through a convex combination (mixture) of vectors, each corresponding to an absolute priority order. In section III we consider only absolute priorities and priority mixtures in section IV.

3) Work Conservation: The work conservation law says that, given $\{\lambda_{ij}\}$, the total average amount of work in server j, namely the total average request queue length is the same, regardless of scheduling discipline. Thus, if we reduce the request queueing delay of a client, that of another will increase. For the $M/M/1$ queue, work conservation holds for average retrieval delays, i.e. $\sum_{i=1}^{N} \lambda_{ij} D_{ij} = \Lambda_j/(C_j - \Lambda_j)$.

C. Total Average Retrieval Delay

The performance objective of peer i is the *total average delay* resulting from its request load splits and service by others,

$$D_i(\mathbf{\Lambda}, \boldsymbol{\pi}) = \sum_{j \neq i} \frac{\lambda_{ij}}{r_i} D_{ij}(\mathbf{\Lambda}, \boldsymbol{\pi}_j) \tag{3}$$

Note that, in general D_i depends on the splitting decisions of other clients and on the scheduling policies of servers. The network operating point is the vector of average retrieval delays $\mathbf{D} = (D_1, \ldots, D_N)$. The system-wide social performance metric is the total average network delay,

$$D_{\text{tot}} = \frac{1}{\sum_i r_i} \sum_{i=1}^{N} r_i D_i = \frac{1}{\sum_i r_i} \sum_{i=1}^{N} \sum_{j \neq i}^{N} \lambda_{ij} D_{ij}. \tag{4}$$

III. Client Request Load Splitting Games

First, we fix the ensemble of scheduling disciplines $\boldsymbol{\pi} = (\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_N)$. We study the following cases: (a) all servers employ FIFO scheduling, (b) each server j employs an arbitrary absolute priority scheduling $\boldsymbol{\pi}_j$ other than FIFO.

We consider best response strategy updates which naturally emerge in autonomous, spontaneously interacting non-cooperative peers. Each peer i finds the strategy $\boldsymbol{\lambda}_i$ that minimizes a certain performance objective function $F_i(\mathbf{\Lambda})$. In doing so, it affects the performance objective functions of other peers as well. These in turn need to readjust their strategies to obtain the best instantaneously achievable outcome for them, and so on. In order to capture a range of possible peer behaviors, we introduce the class of selfish load splitting policies, characterized by the following performance objective functions, indexed by parameter $\beta_i \in \mathbb{R}$,

$$F_i^{\beta_i}(\mathbf{\Lambda}, \boldsymbol{\pi}) = \frac{r_i}{\sum_k r_k} D_i(\mathbf{\Lambda}, \boldsymbol{\pi}) + \beta_i \sum_{j \neq i} \frac{r_j}{\sum_k r_k} D_j(\mathbf{\Lambda}, \boldsymbol{\pi}). \tag{5}$$

For $\beta_i = 0$, the client peer i is called *truly selfish* or *egotistic* since it attempts to find a strategy $\boldsymbol{\lambda}_i$ that minimizes its own delay D_i and is indifferent to delays of others. The corresponding best response strategy updates are called egotistic as well. For $\beta_i = 1$, the peer is *pseudo-selfish* or *altruistic*, in the sense that it also takes into account the impact of its strategy on delays D_j of other peers besides its own delay D_i. The performance objective is $F_i^1(\cdot) = D_{\text{tot}}(\cdot)$. For $0 < \beta_i < 1$, the client i objective and behavior are between the two extremes above. Parameter β_i is private for each peer.

For given ensemble of scheduling policies $\boldsymbol{\pi}$ and vector $\boldsymbol{\beta}$, a load splitting strategy $\mathbf{\Lambda}^* = (\boldsymbol{\lambda}_1^*, \ldots, \boldsymbol{\lambda}_N^*)$ is a Nash Equilibrium Point (NEP) for $\boldsymbol{\pi}$, if no peer can benefit by unilaterally deviating from the NEP, i.e if

$$F_i^{\beta_i}(\boldsymbol{\lambda}_i^*, \boldsymbol{\lambda}_{-i}^*, \boldsymbol{\pi}) \leq F_i^{\beta_i}(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i}^*, \boldsymbol{\pi}) \tag{6}$$

for all $\boldsymbol{\lambda}_i \neq \boldsymbol{\lambda}_i^*$ and all $i \in \mathcal{N}$. We denote this NEP as $\mathbf{\Lambda}^*(\boldsymbol{\beta})$. Different values of $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_N)$ yield different peer interactions and different NEPs. For instance, $\mathbf{\Lambda}^*(\mathbf{0})$ (hereforth denoted by $\mathbf{\Lambda}^*$) is the NEP for egotistic peers, where $D_i(\boldsymbol{\lambda}_i^*, \boldsymbol{\lambda}_{-i}^*, \boldsymbol{\pi}) \leq D_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i}^*, \boldsymbol{\pi})$ for all $\boldsymbol{\lambda}_i \neq \boldsymbol{\lambda}_i^*$ and $i \in \mathcal{N}$.

A. Egotistic Best Response Policies

For $\boldsymbol{\beta} = \mathbf{0}$, each peer i faces the best response problem

$$\min_{\boldsymbol{\lambda}_i \in \mathcal{F}_i} D_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i}, \boldsymbol{\pi}). \tag{7}$$

Given the splitting profile $\boldsymbol{\lambda}_{-i}$ of peers other than i, and for *any given* ensemble of scheduling policies $\boldsymbol{\pi}$ (FIFO or priority), it can be verified from (2) and (3) that the average delay $D_i(\cdot)$ of peer i is a convex function of its splitting strategy $\boldsymbol{\lambda}_i$. This establishes *existence* of a NEP in the load splitting game [17].

The Kuhn-Tucker optimality conditions imply that $\boldsymbol{\lambda}_i$ is the best response of i to $\boldsymbol{\lambda}_{-i}$ if and only if there exist Lagrange multipliers ν_i and $\boldsymbol{\mu}_i = (\mu_{ij} : j = 1, \ldots, N, j \neq i)$ such that:

$$\frac{\partial D_i}{\partial \lambda_{ij}}(\mathbf{\Lambda}, \boldsymbol{\pi}) - \nu_i - \mu_{ij} = 0, \;\; j \neq i, \tag{8}$$

$$\sum_{j \neq i} \lambda_{ij} = r_i, \;\; \text{and} \;\; \mu_{ij} \lambda_{ij} = 0, \;\; j \neq i \tag{9}$$

with $\mu_{ij} \geq 0$, $\lambda_{ij} \geq 0$. A splitting strategy $\mathbf{\Lambda}^* = (\boldsymbol{\lambda}_1^*, \ldots, \boldsymbol{\lambda}_N^*)$ is a NEP if and only if conditions (8),(9) are satisfied for all $i = 1, \ldots, N$. Clearly, the ensemble of scheduling policies $\boldsymbol{\pi}$ plays a decisive role in the equilibrium.

A desirable attribute of the game is convergence of iterative best response load splitting to a NEP. Best response is the best myopic strategy in terms of optimizing delay, without taking into account potential updates of others' strategies in future moves, and it is a natural way of modeling spontaneous interaction. NEPs are therefore predictions of collective peer behavior. Thus, it would be desirable if we could establish that iterative best response updates converge to a NEP; the NEP would then be the natural system operating point.

1) FIFO Scheduling: Conditions (8),(9) are equivalent to:

$$\frac{C_j - \Lambda_j^{-i}}{r_i (C_j - \Lambda_j)^2} = \nu_i, \;\; \text{if } \lambda_{ij} > 0, \tag{10}$$

and $1/(r_i(C_j - \Lambda_j)) \geq \nu_i$ if $\lambda_{ij} = 0$. Uniqueness of NEP for selfish routing to parallel links with a link cost function corresponding to FIFO was shown in [5]. Interestingly, convergence of the sequence of best responses to the NEP is shown numerically but not formally, except for the two-user two-link case. Our load splitting game differs from selfish routing; in the latter, all links are available to all users, while in our game, the subset of links (peers) to which a user (client) can send load is $\mathcal{N} \setminus \{i\}$. Uniqueness of the NEP for restricted subset of links cannot be guaranteed [18]. However, in our case we can modify the proof in [5] and show that *the NEP for FIFO scheduling is unique.*

For each client peer i and server j define $C_{ij} = C_j - \Lambda_j^{-i}$ as the capacity of server j minus the total load of peers other than i to j. Note that C_{ij} depends only on $\boldsymbol{\lambda}_{-i}$. Given $\boldsymbol{\lambda}_{-i}$, assume that i ranks servers j as $C_{i1} \geq C_{i2} \geq \ldots \geq C_{iN}$. From the KKT conditions for λ_{ij}, peer i finds its best response to strategy profile $\boldsymbol{\lambda}_{-i}$ as the waterfilling solution,

$$\lambda_{ij} = C_{ij} - \frac{\sqrt{C_{ij}}}{\sum_{j \neq i}^{K_i} \sqrt{C_{ij}}} \Big(\sum_{j \neq i}^{K_i} C_{ij} - r_i \Big), \tag{11}$$

if $j \leq K_i$, and $\lambda_{ij} = 0$ otherwise, where

$$K_i = \max\{\ell : \sqrt{C_{i\ell}} \geq \frac{(\sum_{j\neq i}^{\ell} C_{ij} - r_i)}{\sum_{j\neq i}^{\ell} \sqrt{C_{ij}}}\}. \quad (12)$$

The machinery of best response is as follows. The updates take place once in a given time interval. At interval n, a peer i measures average delay per unit of flow $D_{ij}^{(n)}$ it experiences at each server peer j. This can be measured through fields in request and received data packets that contain the times when the request was released and data packets were received, and identities of servers. Peer i also knows its strategy $\boldsymbol{\lambda}_i^{(n)}$ and it can deduce C_{ij} as $C_{ij} = \lambda_{ij} + (1/D_{ij})$. It can then determine its best response, $\boldsymbol{\lambda}_i^{(n+1)}$ through (11). Note that peer i does not need any other information and may rely on local measurements for computing its best response. We have experimentally verified that the sequence of best response updates converges to the NEP, the stable operating point with respect to individual peer deviations.

If $r_i = r$, $C_i = C$ for all i, we have at NEP that $\lambda_{ij}^* = \frac{r}{N-1}$, that is, each peer equally splits its load to all servers. The client delays at NEP are $D_i(\boldsymbol{\Lambda}^*(\mathbf{0})) = 1/(C-r)$ for all i. In general however, client delays at NEP are not equal to each other.

2) Priority Scheduling: Consider now a *given* network scheduling profile $\boldsymbol{\pi}$ other than FIFO, such that the scheduling policy π_j at each server j is an absolute priority one. This case generalizes the FIFO one since now each client experiences different cost (average delay) per unit flow at different servers due to different priorities it enjoys at them. For each client i and server j define $C_{ij}^+ = C_j - \Lambda_j^{i+}$, namely the capacity of server j minus the load of peers that are served with higher priority than i at server j. Again C_{ij}^+ depends only on $\boldsymbol{\Lambda}_{-i}$. Note that it may be $\pi_i^j < \pi_i^k$ but $C_{ij}^+ < C_{ik}^+$, that is, although a peer i may enjoy higher priority at server j than at k, it may be more beneficial for i to split load to k rather than j, since the aggregate higher priority traffic at server j may be more. The delay for priority scheduling is more general than that for FIFO. However, we can prove uniqueness of the NEP even for this general case by following the structure of the proof in [5]. We provide a sketch of the proof with the main differences.

Theorem 1: For any scheduling rule $\boldsymbol{\pi}$, the NEP is unique.

Proof: Let $\boldsymbol{\Lambda}$, $\tilde{\boldsymbol{\Lambda}}$ be two NEPs. We will prove that $\boldsymbol{\Lambda} = \tilde{\boldsymbol{\Lambda}}$. The KKT conditions for $\boldsymbol{\Lambda}$ imply that

$$\frac{C_j}{r_i(C_j - \Lambda_j^{i+} - \lambda_{ij})^2} = \nu_i, \text{ if } \lambda_{ij} > 0, \quad (13)$$

and $C_j/[r_i(C_j - \Lambda_j^{i+})] \geq \nu_i$, if $\lambda_{ij} = 0$, for $i \in \mathcal{N}$. Similar conditions (with different multipliers $\tilde{\nu}_i$) hold for $\tilde{\boldsymbol{\Lambda}}$. Similarly in rationale with [5, Th.2.1], we prove that for each client i, and for each $j \neq i$, $k \neq i, j$,

$$\tilde{\nu}_i \leq \nu_i \text{ and } \tilde{\Lambda}_j^{k+} \geq \Lambda_j^{k+} \quad \text{imply that} \quad \tilde{\lambda}_{ij} \leq \lambda_{ij}$$
$$\tilde{\nu}_i \geq \nu_i \text{ and } \tilde{\Lambda}_j^{k+} \leq \Lambda_j^{k+} \quad \text{imply that} \quad \tilde{\lambda}_{ij} \geq \lambda_{ij}.$$

Next, the proof differs from [5]. For some client k, define server sets $\mathcal{N}_k^1 = \{j : \tilde{\Lambda}_j^{k+} > \Lambda_j^{k+}\}$, $\mathcal{N}_k^2 = \mathcal{N} - \mathcal{N}_k^1$. Let $\mathcal{N}' = \{i \in \mathcal{N} : \tilde{\nu}_i > \nu_i\}$. For $i \in \mathcal{N}'$, and $i \notin \mathcal{N}'$ we have, respectively,

$$\sum_{j\in\mathcal{N}_k^1} \tilde{\lambda}_{ij} = r_i - \sum_{j\in\mathcal{N}_k^2} \tilde{\lambda}_{ij} \leq r_i - \sum_{j\in\mathcal{N}_k^2} \lambda_{ij} = \sum_{j\in\mathcal{N}_k^1} \lambda_{ij} \quad (14)$$

$$\sum_{j\in\mathcal{N}_k^1} \tilde{\Lambda}_j^{k+} = \sum_{j\in\mathcal{N}_k^1} \sum_{i:\pi_i^j<\pi_k^j} \tilde{\lambda}_{ij} \leq \sum_{j\in\mathcal{N}_k^1} \sum_{i:\pi_i^j<\pi_k^j} \lambda_{ij} = \sum_{j\in\mathcal{N}_k^1} \Lambda_j^{k+}. \quad (15)$$

The last inequality contradicts the definition of $\mathcal{N}_k^1$, which means that $\mathcal{N}_k^1$ is empty set. Similarly, set $\{j : \tilde{\Lambda}_j^{k+} < \Lambda_j^{k+}\}$ is empty. Thus, we have that for every server j, it is $\tilde{\Lambda}_j^{k+} = \Lambda_j^{k+}$ for any client k. We proceed as in [5] to show that $\tilde{\nu}_i = \nu_i$ for each i. Thus, we get that $\tilde{\lambda}_{ij} = \lambda_{ij}$ for every i and j, which implies that the NEP is unique. ■

To compute its best response, each peer i needs to know capacities C_j of servers j, which is information available in contemporary peer-to-peer systems. Similar to FIFO, at each iteration n, a peer i can measure $D_{ij}^{(n)}$ from different servers j. Since it knows $\lambda_{ij}^{(n)}$, it can find C_{ij}^+ as the positive root of equation $D_{ij}^{(n)}x^2 - D_{ij}^{(n)}\lambda_{ij}^{(n)}x - C_j = 0$. It then ranks servers j in decreasing order of $C_{ij}^+/\sqrt{C_j}$ and derives its best response strategy $\boldsymbol{\lambda}_i^{(n+1)}$ to a given strategy $\boldsymbol{\lambda}_{-i}$ of others as,

$$\lambda_{ij} = C_{ij}^+ - \frac{\sqrt{C_j}}{\sum_{j\neq i}^{K_i^+} \sqrt{C_j}} (\sum_{j\neq i}^{K_i^+} C_{ij}^+ - r_i) \text{ if } j \leq K_i^+ \quad (16)$$

and $\lambda_{ij} = 0$ otherwise, where

$$K_i^+ = \max\{\ell : C_{i\ell}^+ \geq \frac{\sqrt{C_\ell}}{\sum_{j\neq i}^{\ell} \sqrt{C_j}} (\sum_{j\neq i}^{\ell} C_{ij}^+ - r_i)\}. \quad (17)$$

The sequence of best response updates is numerically verified to converge to the NEP.

B. Altruistic Best Response Policies

We now consider altruistic peers, namely $\beta_i = 1$ for all i. All peers know that they are altruistic. We study best response policies where each peer finds its strategy $\boldsymbol{\lambda}_i$ by considering also its impact on average delays of others. For given strategy profile $\boldsymbol{\Lambda}_{-i}$, the altruistic best response policy of peer i emerges as the solution of:

$$\min_{\boldsymbol{\lambda}_i \in \mathcal{F}_i} \frac{1}{\sum_k r_k} \sum_{j\neq i} \left(\lambda_{ij} D_{ij} + \sum_{k\neq i,j} \lambda_{kj} D_{kj}\right) = \min_{\boldsymbol{\lambda}_i \in \mathcal{F}_i} D_{\text{tot}}(\boldsymbol{\Lambda}, \boldsymbol{\pi}) \quad (18)$$

1) FIFO Scheduling: Since $D_{\text{tot}}(\cdot)$ is convex function of $\boldsymbol{\lambda}_i$, a NEP exists. One can show that $D_{\text{tot}}(\cdot)$ is jointly convex in $\boldsymbol{\Lambda}$, since the Hessian matrix of $D_{\text{tot}}(\cdot)$ is positive definite.

The KKT conditions for peer i are:

$$\frac{C_j}{(C_j - \Lambda_j)^2} = \nu_i, \quad \text{if } \lambda_{ij} > 0, \quad (19)$$
$$\frac{C_j}{(C_j - \Lambda_j)^2} \geq \nu_i \quad \text{if } \lambda_{ij} = 0.$$

Peer i finds its best response to $\boldsymbol{\lambda}_{-i}$ as,

$$\lambda_{ij} = C_{ij} - \frac{\sqrt{C_j}}{\sum_{j\neq i}^{L_i}\sqrt{C_j}}(\sum_{j\neq i}^{L_i} C_{ij} - r_i), \tag{20}$$

if $j \leq L_i$, and $\lambda_{ij} = 0$ otherwise, where

$$L_i = \max\{\ell : C_{i\ell} \geq \frac{(\sum_{j\neq i}^{\ell} C_{ij} - r_i)}{\sum_{j\neq i}^{\ell}\sqrt{C_j}}\sqrt{C_\ell}\,\}. \tag{21}$$

Each client i needs to know capacities C_j of all servers j. At each iteration n, a peer i can measure $D_{ij}^{(n)}$ from different clients j. Since it knows $\lambda_{ij}^{(n)}$, it can find C_{ij} as in FIFO and then it ranks servers in decreasing order of $C_{ij}/\sqrt{C_j}$. Then it derives its best response strategy $\boldsymbol{\lambda}_i^{(n+1)}$ from KKT conditions above. We now study NEPs and best response updates.

Equal Capacities: If $C_i = C$ for all i, the global problem,

$$\min_{\boldsymbol{\Lambda}\in\mathcal{F}} D_{\text{tot}}(\boldsymbol{\Lambda}) = \frac{1}{\sum_k r_k}\sum_{j=1}^{N}\frac{\Lambda_j}{C-\Lambda_j}. \tag{22}$$

falls within the class of problems:

$$\text{Problem (P)}: \quad \min_{\boldsymbol{\Lambda}\in\mathcal{F}} D(\boldsymbol{\Lambda}) = \sum_{j=1}^{N} G(\Lambda_j) \tag{23}$$

where $G(\cdot)$ is a strictly nondecreasing convex function of load Λ_j at server j. For problem (P), we have [19, Corollary 4]:

Fact 1: A feasible load splitting policy $\boldsymbol{\Lambda}$ is solution to (P) for a given strictly convex function $G(\cdot)$ if and only if it is a solution to problem (P) for all convex functions.

In the sequel, we use function $G(x) = x^2$. A solution to problem (P) with $G(x) = x^2$ is solution to (P) for any other convex function $G(\cdot)$ and thus a solution to (22).

For a feasible splitting strategy $\boldsymbol{\Lambda} \in \mathcal{F}$ and a client i, the altruistic best response is a strategy $\mathcal{T}_i\boldsymbol{\Lambda}$, where $\mathcal{T}_i$ is an operator on $\boldsymbol{\Lambda}$, and $\mathcal{T}_i\boldsymbol{\Lambda} = \arg\min_{\boldsymbol{\lambda}_i\in\mathcal{F}_i} D(\boldsymbol{\Lambda})$, while strategies $\boldsymbol{\lambda}_{-i}$ are kept fixed. Let $\boldsymbol{\Lambda}^{(0)}$ be an initial load splitting strategy, and let $\{i_k\}_{k\geq 1}$ be a sequence of client indices. Assume there exists an integer Δ such that for any integer ℓ and any $i' \in \mathcal{N}$, it is $i_k = i'$ for some k with $\ell \leq k \leq \ell + \Delta$. In other words, all clients should make an update within a finite period of Δ iterations. Define the sequence of assignments $\{\boldsymbol{\Lambda}^{(k)}, k \geq 0\}$ recursively as $\boldsymbol{\Lambda}^{(k+1)} = \mathcal{T}_{i_k}\boldsymbol{\Lambda}^{(k)}$. We follow the rationale in [19] with all necessary modifications. We first prove the following lemma.

Lemma 1: Suppose $C_i = C$ for all i. Let $\boldsymbol{\Lambda}$, $\tilde{\boldsymbol{\Lambda}}$ be feasible strategies based on $\boldsymbol{\Lambda}$ and $\mathcal{T}_i\boldsymbol{\Lambda}$ respectively. For $\boldsymbol{\Lambda} \in \mathcal{F}$ and $i \in \mathcal{N}$, define $||\boldsymbol{\Lambda} - \mathcal{T}_i\boldsymbol{\Lambda}|| = \max_{j\in\mathcal{N}} |\lambda_{ij}(\boldsymbol{\Lambda}) - \lambda_{ij}(\mathcal{T}_i\boldsymbol{\Lambda})|$. Then, $||\boldsymbol{\Lambda} - \mathcal{T}_i\boldsymbol{\Lambda}||^2 \leq D(\boldsymbol{\Lambda}) - D(\tilde{\boldsymbol{\Lambda}})$.

Proof: Let Λ_j and $\tilde{\Lambda}_j$ be the loads of server j according to $\boldsymbol{\Lambda}$ and $\tilde{\boldsymbol{\Lambda}}$. Since it is $\sum_{j\neq i}\lambda_{ij} = \sum_{j\neq i}\tilde{\lambda}_{ij} = r_i$, we get $\sum_{j\in\mathcal{N}}(\Lambda_j - \tilde{\Lambda}_j) = 0$. Define $\sigma = \min_j \Lambda_j$. We have,

$$\begin{aligned}
D(\boldsymbol{\Lambda}) - D(\tilde{\boldsymbol{\Lambda}}) &= \sum_{j\in\mathcal{N}\setminus\{i\}} (\Lambda_j^2 - \tilde{\Lambda}_j^2) \\
&= \sum_{j\in\mathcal{N}\setminus\{i\}} \left[\Lambda_j^2 - \tilde{\Lambda}_j^2 - 2\sigma(\Lambda_j - \tilde{\Lambda}_j)\right] \\
&= \sum_{j\in\mathcal{N}\setminus\{i\}} \left[(\Lambda_j - \sigma)^2 - (\tilde{\Lambda}_j - \sigma)^2\right] \\
&\geq \sum_{j\in\mathcal{N}\setminus\{i\}} (\Lambda_j - \tilde{\Lambda}_j)^2 \geq \max_{j\in\mathcal{N}\setminus\{i\}} |\Lambda_j - \tilde{\Lambda}_j|^2 \\
&= \max_{j\in\mathcal{N}\setminus\{i\}} |\lambda_{ij} - \tilde{\lambda}_{ij}|^2 = ||\boldsymbol{\Lambda} - \mathcal{T}_i\boldsymbol{\Lambda}||^2,
\end{aligned}$$

where the first inequality holds since $-\sigma \geq -\tilde{\Lambda}_j$ for all j. ■

For $C_i = C$, $D(\boldsymbol{\Lambda}^{(k)})$ is monotone non-increasing with k, since $D(\boldsymbol{\Lambda}) - D(\tilde{T}_i\boldsymbol{\Lambda}) \geq 0$ from Lemma 1. The following theorem shows the convergence of the iterative procedure.

Theorem 2: Suppose that $C_i = C$ for all i. Then, any limit point of the sequence of best response updates, $\{\boldsymbol{\Lambda}^{(k)}\}_{k=1,2,\ldots}$, is a NEP of the altruistic load splitting game and an optimal solution to Problem (P).

Proof: Since $D(\cdot)$ is a nonnegative and continuous function, we have that $\lim_{k\to\infty}(D(\boldsymbol{\Lambda}^{(k+1)}) - D(\boldsymbol{\Lambda}^{(k)})) = 0$. In addition, we have $\lim_{k\to\infty} D(\boldsymbol{\Lambda}^{(k)}) = D(\boldsymbol{\Lambda}^*)$ for any limit point $\boldsymbol{\Lambda}^*$ of $\boldsymbol{\Lambda}^{(k)}$, $k \geq 0$. From Lemma 1, it follows that $\lim_{k\to\infty}|\boldsymbol{\Lambda}^{(k+1)} - \boldsymbol{\Lambda}^{(k)}| = 0$. Since mapping $\mathcal{T}_i$ is continuous for all i, we have $\mathcal{T}_i\boldsymbol{\Lambda}^* = \boldsymbol{\Lambda}^*$ for any i and any limit point $\boldsymbol{\Lambda}^*$. This means that no peer i can benefit from unilaterally deviating from $\boldsymbol{\Lambda}^*$, hence $\boldsymbol{\Lambda}^*$ is a NEP for the altruistic splitting game. Since $\mathcal{T}_i\boldsymbol{\Lambda}^* = \boldsymbol{\Lambda}^*$ for all i, $\boldsymbol{\Lambda}^*$ is a stationary point of $D_{\text{tot}}(\cdot)$. Since $D_{\text{tot}}(\cdot)$ is jointly convex with respect to $\boldsymbol{\Lambda}$, a stationary point of $D_{\text{tot}}(\cdot)$ is a global optimal solution to problem (P). ■

The altruistic game leads to multiple NEPs $\boldsymbol{\Lambda}^*$ that minimize $D_{\text{tot}}(\cdot)$, and all induce the same server load vector $\mathbf{v} = (\Lambda_1^*, \ldots, \Lambda_N^*)$. Load vector $\mathbf{v}$ has some interesting properties. First we give a few definitions. For vector $\boldsymbol{\alpha}$, let $\boldsymbol{\alpha}'$ denote the vector with the entries of $\boldsymbol{\alpha}$ arranged in decreasing order, i.e $\alpha_1' = \max_i \alpha_i$. Vector $\boldsymbol{\alpha}$ is *lexicographically smaller* than vector $\mathbf{b}$, if either $\alpha_1' < b_1'$ or for some i, $1 \leq i < N$ it is $\alpha_j' = b_j'$ for $1 \leq j \leq i$ and $\alpha_{i+1}' < b_{i+1}'$. A vector $\boldsymbol{\alpha}$ is *more balanced* than $\mathbf{b}$ if $\sum_{\ell=1}^{i}\alpha_\ell' \leq \sum_{\ell=1}^{i} b_\ell'$, $i = 1, \ldots, N$. The server load vector $\mathbf{v}$ is *the most balanced* vector, since it minimizes $\sum_{i=1}^{N} G(\Lambda_j)$ where $G(\cdot)$ is any nondecreasing strictly convex function [19], [21, Theorem 7]. Furthermore, $\mathbf{v}$ is the *lexicographically minimal* vector.

If $r_i = r$ for all i, all client delays are $1/(C-r)$.

Unequal Capacities: In that case, the proof procedure outlined above cannot be applied, since the global problem does not belong to the class of problems represented by problem (P). However, the sequence of best response updates is in essence a sequence of Gauss-Seidel iterations(18). Since $D_{\text{tot}}(\cdot)$ is jointly convex with respect to $\boldsymbol{\Lambda}$ and convex with respect to $\boldsymbol{\lambda}_i$ for fixed $\boldsymbol{\Lambda}_{-i}$, this sequence converges to limit points that are global minima of $D_{\text{tot}}(\cdot)$ [20, pp.219-221] and also NEPs of the altruistic game.

2) Priority Scheduling: For a given priority profile $\boldsymbol{\pi}$, altruistic best response updates by peer i are optimization problems that are convex in peer i's strategy $\boldsymbol{\lambda}_i$. Now, the expression for delay becomes more complicated and may not be jointly convex in $\boldsymbol{\Lambda}$. Each peer i minimizes its own delay plus the delays of peers that are served with lower priority than i at different servers. Besides server capacities, peer i needs to know the amount of higher priority traffic of each client ℓ at each server j. The limit points of best response updates are NEPs of the game and local minima of $D_{\text{tot}}(\cdot)$. Our intensive experiments show existence of multiple NEPs with suboptimal performance, which are limit points of the sequence of best responses.

IV. Client - Server Games

We now consider the case where peers may change their scheduling discipline in addition to their load splits. We consider egotistic rules. A change of schedule at one or more servers leads to different advertised delays to clients. These in turn need to readjust their load splits in a best response fashion, so that they minimize their delay for the new scheduling regime.

In section III, a peer could affect its delay by controlling its load splits. But, can a peer affect its own delay by altering its scheduling policy? Although it cannot do so directly, it may select a scheduling policy so that the NEP *after* the subsequent load splitting game is the most beneficial for it. Suppose that at a NEP, peer i observes large delay D_{ij}^* at some peer j and that it realizes that this is mainly due to a peer k with large load λ_{kj} which is served by j with higher priority than i. Then, peer i may decide to offer peer k higher priority than before by changing its schedule. At the next round of egotistic load splits, i will attract part of the load of k, since k will selfishly split its traffic toward peers that advertise low delays. Thus, delay D_{ij}^* at the next NEP (and therefore D_i^*) will be less since i will have less higher priority load at server j to compete with.

In section III, we defined the load splitting NEP for given scheduling profile $\boldsymbol{\pi}$. Let $\boldsymbol{\Lambda}^{\boldsymbol{\pi}}$ be the load splitting NEP for $\boldsymbol{\pi}$. We generalize the definition as follows. A load splitting and scheduling strategy, $(\boldsymbol{\Lambda}^*, \boldsymbol{\pi}^*) = (\boldsymbol{\lambda}_1^*, \ldots, \boldsymbol{\lambda}_N^*, \boldsymbol{\pi}_1^*, \ldots, \boldsymbol{\pi}_N^*) \equiv (\boldsymbol{\Lambda}^{\boldsymbol{\pi}^*}, \boldsymbol{\pi}^*)$ is a NEP if $D_i(\boldsymbol{\lambda}_i^{\boldsymbol{\pi}^*}, \boldsymbol{\lambda}_{-i}^{\boldsymbol{\pi}^*}, \boldsymbol{\pi}^*) \leq D_i(\boldsymbol{\lambda}_i^{\boldsymbol{\pi}}, \boldsymbol{\lambda}_{-i}^{\boldsymbol{\pi}}, \boldsymbol{\pi})$. for all $\boldsymbol{\pi} \neq \boldsymbol{\pi}^*$ and all $i \in \mathcal{N}$.

A. Two-stage Client-Server Games

We devise two-stage games that capture the dual client-server role of a peer. Suppose that, at first stage, a peer determines a best response scheduling policy as server. Subsequently, peers play a request load splitting game as clients based on this scheduling profile and perform best response load split updates until convergence to NEP. In the next round, another peer adjusts its scheduling policy and peers again play the load splitting game based on the new policy and so on. Selection of a scheduling policy by server i consists in finding either an *absolute* priority ordering m^* of peers $\mathcal{N} \setminus \{i\}$ or a *mixture* of absolute priorities, represented by a set of portions $\{a_i^m\}, m = 1, \ldots, \mathcal{M}_i$, such that the delay D_i^* at NEP after the load splitting game is minimized. Consider finding a mixture of absolute priorities. Then the delay of peer k at server ℓ is $D_{k\ell} = \sum_m a_\ell^m D_{k\ell}^m$, where, according to (2), $D_{k\ell}^m = C_\ell[(C_\ell - \Lambda_\ell^{mk+})(C_\ell - \Lambda_\ell^{mk+} - \lambda_{k\ell})]^{-1}$ is the delay experienced by client k at server ℓ based on priority order $m \in \mathcal{M}_\ell$, and Λ_ℓ^{mk+} is the load of peers of higher priority than k at server ℓ for the priority order m.

Peer i needs to go one step ahead and compute the *provisional* best response splits $\{\lambda_{k\ell}^*\}$ of peer k to each server $\ell \neq k$ (including server i) at the NEP of the subsequent load splitting game. As in (13), the KKT conditions for each peer k yield

$$\sum_{m \in \mathcal{M}_\ell} a_\ell^m \frac{C_\ell}{\left(C_\ell - \Lambda_\ell^{mk+} - \lambda_{k\ell}\right)^2} = r_k \nu_k, \text{ if } \lambda_{k\ell} > 0. \quad (24)$$

for $\ell \neq k$. The NEP splits $\{\lambda_{k\ell}^*\}$, (as well NEP splits $\{\lambda_{ki}^*\}$ that peer i attracts), are affected by portions $\{a_i^m\}_{m \in \mathcal{M}_i}$, since the splits depend on Lagrange multipliers $\{\nu_k\}$ which in turn depend on portions $\{a_\ell^m\}$ through (24) and the constraint $\sum_{\ell \neq k} \lambda_{k\ell}^* = r_k$. Thus, peer i wishes to determine best response priority portions $\{a_i^m\}, m = 1, \ldots, \mathcal{M}_i$ so that its delay at NEP,

$$D_i^* = \frac{1}{r_i} \sum_{j \neq i} \lambda_{ij}^* \frac{C_j}{(C_j - \Lambda_j^{i+*})(C_j - \Lambda_j^{i+*} - \lambda_{ij}^*)} \quad (25)$$

is minimum. Note that portions $\{a_i^m\}$ are hidden in the expressions of Λ_j^{i+*}. After the change in priority portions $\{a_i^m\}$, peers play the splitting game anew and yield another NEP. Next, another peer computes its best response priority portions to minimize its own delay, then peers play the splitting game, and so on. A NEP $(\boldsymbol{\Lambda}^*, \boldsymbol{\pi}^*)$ consists of a splitting strategy $\boldsymbol{\Lambda}^*$ and priority portions $\{a_i^{m*} : m \in \mathcal{M}_i, i \in \mathcal{N}\}$ and is guaranteed to exist, since $\boldsymbol{\pi}^*$ can be viewed as a *mixed* strategy over the finite set of pure strategies, i.e, the different priority orders.

The computation of $\{a_i^{m*}\}, m = 1, \ldots, \mathcal{M}_i$ by each peer i is a computationally intensive problem, for which information from servers is needed in terms of load splits of peers and priority portions, which is difficult to obtain. To reduce complexity, one could instead consider only absolute priority policies. For each of the $|\mathcal{M}_i|!$, possible priority orders, $m = 1, \ldots |\mathcal{M}_i|!$, peer i runs the provisional splitting game and records delay $D_i^*(m)$. Then, it selects order $m^* = \arg\min_n D_i^*(m)$. However, existence of NEP is not guaranteed in that case.

Instead of best response schedule rule adaptation that involves a priori computation of NEPs, each peer i may perform *heuristically good response* schedule updates through heuristic metrics for prioritizing peers. Peer i computes these metrics for $j \neq i$ and finds an absolute or mixed priority ordering according to them. Such metrics are:

- A metric denoting total hurdle that peer j causes to peer i in all servers where j enjoys higher priority than i, $\Theta_{ij} = \sum_{\ell : \pi_\ell^j < \pi_\ell^i} (\lambda_{j\ell}/C_\ell)$.
- A metric that captures the total reduction of the delay of peer i at various servers, per unit of removed flow of peer j, if it is served with higher priority than i, $H_{ij} = \sum_{\ell : \pi_\ell^j < \pi_\ell^i} |(\partial D_{i\ell}/\partial \lambda_{j\ell})|$.

Peer i should prioritize peers in decreasing order of Θ_{ij} or H_{ij}. Alternatively, i may find a mixture of absolute priorities with priority portions based on the metrics above. In the numerical results section we provide some rules for finding good enough

mixtures of priorities. Alternatively, i could choose a subset of peers, e.g. those with large metrics and apply mixtures of priority orderings based on the metrics above.

V. Numerical Results

To obtain insight about the proposed game models and impact of various parameters on performance, we simulate a toy system of $N = 5$ peers. For better tractability of results we first consider a network where each peer has capacity $C = 256$kbps and average request file size $L = 1024$kB. For request load vector $\mathrm{r} = [1.6875, 1.6875, 1.1250, 0.5625, 0.5625]$ (in requests/min), the optimal solution balances the load across servers, i.e. it is $\Lambda_j = 1.125$ requests/min for all j. However, this is not always the case. For load vector $\mathrm{r} = [5.5125, 0.2625, 0.2625, 0.2625, 0.2625]$, at the optimal solution the load is not balanced but is distributed across servers as $[1.0500, 1.3800, 1.3800, 1.3800, 1.3800]$. This is due to the fact that peers other than the first one do not generate enough requests to balance the effect of its actions. We define the following performance metrics:

- Total Delay, $D_{\text{tot}}(\mathbf{\Lambda}^*)$. This is the total average network delay at the NEP.
- Price of Anarchy PoA. This is a metric of proximity of the value of $D_{\text{tot}}(\cdot)$ at NEP $\mathbf{\Lambda}^*$ to its value at the global optimum $\mathbf{\Lambda}^0$, i.e., PoA $= D_{\text{tot}}(\mathbf{\Lambda}^*)/D_{\text{tot}}(\mathbf{\Lambda}^0)$. If several equilibria exist, the worst one is considered.
- Fairness Index, F. This is a measure of dispersion of values of $D_i(\mathbf{\Lambda}^*)$ and is defined as

$$F = \frac{\left[\sum_{i=1}^{N} D_i(\mathbf{\Lambda}^*)\right]^2}{N \sum_{i=1}^{N} D_i^2(\mathbf{\Lambda}^*)}, \tag{26}$$

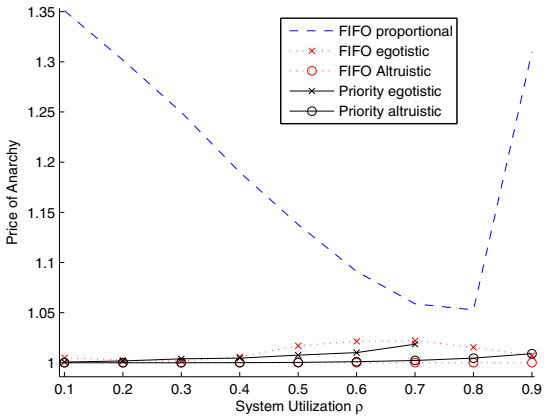

Fig. 1. Price of Anarchy vs. System utilization factor ρ

For comparison we also use a non-iterative splitting strategy, the *Proportional Scheme*, where each peer allocates its requests to servers in proportion to their capacities, i.e $\lambda_{ij} = \frac{C_j}{\sum_{k \neq i}^{N} C_k}$ for $j \neq i$. Whenever we refer to absolute priority schemes, we will imply that priorities are assigned according to the index of each peer. Thus, peer 1 always has highest priority and peer N lowest. The system utilization factor, $\rho = \frac{\sum_{i=1}^{N} r_i}{\sum_{j=1}^{N} C_j}$ will be used in our simulations to determine the request load vector. Finally, wherever we have multiple equilibria, depicted values are mean values of several simulation runs. For the rest of the section peers have different capacities, $C = [128, 256, 640, 768, 768]$.

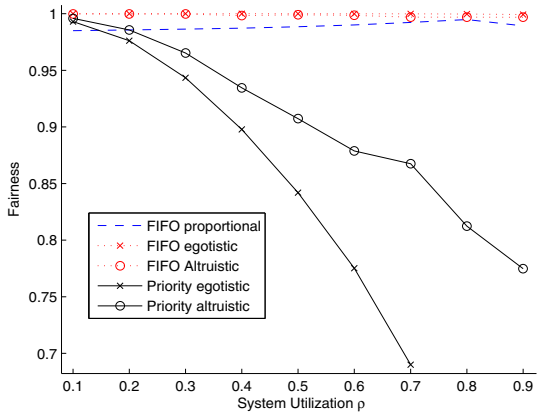

Fig. 2. Fairness vs. System utilization factor ρ

TABLE I
Total Delay Performance at NEP for different games

	FIFO			Preempt. Prior.	
	Egot.	Altr.	Proport.	Egot.	Altr.
$\rho = 0.4$	21.284	21.156	25.175	21.255	21.159
$\rho = 0.6$	34.857	34.125	37.232	34.473	34.162
$\rho = 0.8$	72.289	71.188	74.944	-	71.464

In Table I, we show total delay performance at NEP for different games as function of input load. The altruistic FIFO game always gives the system-wide optimal performance, while egotistic FIFO performs close to that (e.g within 1%). For given load, all games have similar performance, with the exception of priority egotistic for high loads. In such cases, a high priority peer k selects a traffic split that leads the system to instability, a phenomenon quite common for $\rho > 0.5$. This is possible, since the delay of the high priority peers does not depend on others' selections, while their actions affect the delay of the lower priority peers. In the following figures this is denoted by the absence of points for high values of ρ .

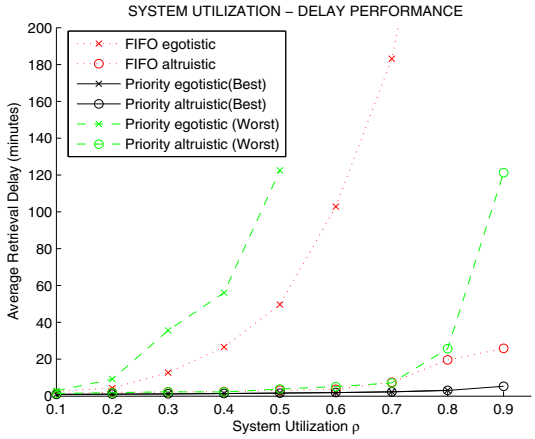

Fig. 3. Total Delay vs. System utilization factor ρ for the case of different file sizes.

In Fig. 1 we depict PoA for client games. A value of PoA close to 1 means less divergence from social optimum due to

TABLE II
CLIENT (FIFO) GAMES AND CLIENT-SERVER (2-STAGE) GAMES

	FIFO		Two-Stage			
	Egot.	Altr.	BR1	BR	H1: Θ_{ij}	H2: H_{ij}
D_{tot}	58.39	58.25	62.48	60.02	63.06	60.24
D_1	66.42	62.54	49.23	56.14	51.54	57.51
D_2	55.83	60.47	100.3	91.21	44.56	50.68
D_3	56.05	53.71	35.55	33.68	86.75	70.60

selfishness. As expected, altruistic games have smaller PoA values, since each peer considers the effect of its actions on others. All games have PoA less than 3% above the optimal. The proportional scheme has the poorest performance.

On the other hand, from Fig.2, it can be deduced that priority schemes become quite unfair as load increases. They tend to prioritize some peers at the expense of others. From 1 we see that for $\rho = 0.6$, the FIFO and the priority altruistic schemes have identical performance. However, the delay vectors of each individual peer are $\mathbf{D_{FIFO}} = [32.3, 33.7, 35.7, 38.3, 33.3]$ and $\mathbf{D_{Pr}} = [15.3, 33.4, 42, 55.7, 55.7]$ respectively. As we can see high priority peers experience extremely low delays, compared to the FIFO case where the delays are quite balanced.

From the above, one might claim that the FIFO egotistic approach has near-optimal performance,since it seems to guarantee near-optimal delay and fairness. However, for different average file of sizes, e.g. $L = [0.5, 7, 30, 200, 700]$MB, we get significantly lower performance than priority schemes (Fig. 3). By prioritizing based on small file sizes, as the μc rule dictates, we may get significantly lower delay. However, if we cannot incentivize peers to assign priorities properly, we may experience significant performance degradation.

For the two-stage client-server game, we need a small enough network, so as to be able to get some insight about the interactions among the peers. Thus, we simulate a network of $N = 3$ peers with capacities $C = [384, 256, 128]$ and low enough loads, $r = [0.6563, 1.0313, 1.1250]$ to guarantee stability. Such a small network is easily tractable. Initially all peers apply the FIFO policy. In Table II, we depict results on total and individual delays at equilibrium. Then, a peer (say peer 1) determines his best response scheduling policy and lets the clients play the load splitting game. Our simulations showed that peer 1 would give absolute priority to peer 3. The resulting delays at the equilibrium are shown in the column titled "BR1". Thus, peer 1 improves his delay. We get similar results when peer 1 selects his new service discipline based on the proposed heuristics. However, our heuristics do not guarantee the improvement of the delay. Then, another peer modifies its service discipline and so on. Such games in general have multiple equilibria to which the client-server best response(BR) game converges. Our Θ_{ij}–heuristic does not lead always to equilibrium, while H_{ij} one has a stationary point (different from the BR equilibrium point). As expected, if each peer is allowed to selfishly select its service discipline as well, the system performance is degraded, but interestingly, degradation is not severe. On the other hand, the individual performance of each user changes significantly, but we cannot predict who will get a benefit and who will lose.

VI. CONCLUSION

This work is a first step towards characterizing peer interactions by capturing the peers' dual client-server role and different behavioral profiles, ranging from egotistic to altruistic ones. We characterized NEPs emerging from best response strategy updates, where the peer strategy set comprised the client request load splits either alone or in conjunction with server schedules. In this work, we adhered mainly to a model with no distinction on the basis of availability of requested item. As we showed the latter can be easily included in our model, if each peer restricts its load splits to a subset of peers that possess the item.

VII. ACKNOWLEDGEMENTS

The authors acknowledge support from the European Commission through the STREP project NET-REFOUND (FP6-IST-034413), STREP project NADA (FP7-ICT-223850), and the NoE project NEWCOM++ (FP7-ICT-216715).

REFERENCES

[1] J. Sethuraman and M. Squillante, "Optimal stochastic scheduling in multiclass parallel queues", *Proc. ACM SIGMETRICS,* 1995.
[2] S. Borst, "Optimal probabilistic allocation of customer types to servers", *Proc. ACM SIGMETRICS,* 1995.
[3] M. Adler, R. Kumar, K. Ross, D. Rubenstein, T. Suel and D.D. Yao, "Optimal peer selection for p2p downloading and streaming", *Proc. IEEE INFOCOM,* 2005.
[4] H. Zhang, G. Neglia, D. Towsley, G. Lo Presti, "On Unstructured File Sharing Networks", *Proc. IEEE INFOCOM,* 2007.
[5] A. Orda, R. Rom and N. Shimkin, "Competitive routing in multiuser communication networks", *IEEE/ACM Trans. Networking,* vol.1, no.5, pp.510-521, Oct. 1993.
[6] E. Altman, T. Basar, T. Jimenez and N. Shimkin, "Routing into two parallel links: Game-theoretic distributed algorithms", *J. Parallel Distr. Comput.,* vol.61, no.9, pp. 1367-1381, Sept. 2001.
[7] R.J. La and V. Anantharam, "Optimal Routing Control: Repeated game approach", *IEEE Trans. Aut. Contr.,* vol.47, no.3, pp.437-450, March 2002.
[8] Y.A. Korilis, A.A. Lazar and A. Orda, "Capacity allocation under noncooperative routing", *IEEE Trans. Aut. Contr.,* vol.42, no.3, pp.309-325, March 1997.
[9] Y.A. Korilis, A.A. Lazar and A. Orda, "Achieving Network Optimal using Stackelberg routing strategies", *IEEE/ACM Trans. Networking,* vol.5, no.1, pp. 161-173, Feb. 1997.
[10] C. Papadimitriou, "Algorithms, games and the internet", *Proc. ACM Symp. Th. of Comput.,* 2001.
[11] T. Roughgarden and E. Tardos, "How bad is selfish routing", *Journal of ACM,* vol.29, pp.235-259, 2002.
[12] T. Roughgarden, "The price of anarchy is independent of the network topology", *Journal of Comput. System Sci.,* vol. 67, no.2, pp.341-364, 2003.
[13] T. Wu and D. Starobinski, "On the price of anarchy in unbounded delay networks", *Proc. ACM GameNets,* 2006.
[14] D. Bertsekas and R. Gallager, *Data Networks,* Prentice Hall, 1992.
[15] I. Adan, Queueing Theory Course : Lecture Notes, *Eindhoven Univ. of Technology,* available online : http://www.win.tue.nl/∼iadan/que/.
[16] S. Borst, L.C.M. Kallenberg and G.M. Koole, "Stochastic Operations Research 2 : Stochastic Dynamic Programming and Control of Queues", *Lecture Notes*, CWI, Amsterdam.
[17] J.B. Rosen, "Existence and uniqueness of equilibrium points for concave n-person games", *Automatica,* vol.33, no.3, pp.520-534, July 1965.
[18] L. Libman and A. Orda, "Atomic resource sharing in non-cooperative networks", *Telecommun. Sys.,* vol. 17, no. 4, pp. 385-409, 2001.
[19] B. Hajek, "Performance of global load balancing by local adjustment", *IEEE Trans. Inf. Theory,* vol.36, no.6, pp.1398-1414, Nov. 1990.
[20] D. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computation : Numerical Methods*, Athena Scientific, 1997.
[21] L. Georgiadis and L. Tassiulas, "Optimal overload response in sensor networks", *IEEE Trans. Inf. Theory,* vol.52, no.6, pp.2684-2696, June 2006.