Client-Server Games and their Equilibria in Peer-to-Peer Networks [☆]

Iordanis Koutsopoulos[a,c], Leandros Tassiulas[a,b], Lazaros Gkatzikis[b,*]

[a] *Centre for Research and Technology Hellas, CERTH-ITI.*
[b] *Department of Computer & Communication Engineering, University of Thessaly, Greece.*
[c] *Department of Computer Science, Athens University of Economics and Business, Greece.*

Abstract

In peer-to-peer networks, each peer acts simultaneously as client and server, by issuing and satisfying content requests respectively. In this work, we use concepts from non-cooperative game theory to model the interaction of autonomous peers. The client strategy set consists of feasible request load splits towards servers, while the server strategy set is formed out of possible service disciplines on received requests. The performance metric of interest is the average retrieval delay of content requests.

First, we assume preassigned fixed server policies (FIFO and priority) and study the emerging client request load splitting game. Peers are either egotistic that are interested only in optimizing their own delay, or altruistic ones that also take into account delay incurred to other peers. We consider best response updates to model iterative peer interaction. For egotistic peers, we show that the sequence of best responses always converges to the unique Nash Equilibrium Point (NEP). For altruistic peers, the best response updates converge to one of the multiple NEPs, with each one being a global optimum for the FIFO case and a local optimum for any other service strategy profile. We also consider mixed swarms consisting of both egotistic and (partially) altruistic peers and show an interesting transition from one to multiple NEPs. Next, we include service strategies in the peer strategy set. Though with its service policy a peer cannot directly affect its delay, it can affect the resulting NEP. We devise two-level game models, where, at a first level, a peer selects its favorable service rule and then peers play a client load splitting game.

Keywords: peer-to-peer, selfishness, two-level games.

[☆]Part of the material in this paper has appeared in a conference version, in [25].

[☆☆]I. Koutsopoulos acknowledges the support of ERC08-RECITAL project, co-financed by Greece and the European Union (European Social Fund) through the Operational Program Education and Lifelong Learning-NCRF 2007-2013.

[*]Corresponding author

Email addresses: `jordan@inf.uth.gr` (Iordanis Koutsopoulos), `leandros@inf.uth.gr` (Leandros Tassiulas), `lagatzik@inf.uth.gr` (Lazaros Gkatzikis)

1. Introduction

Peer-to-peer networks gained momentum in the last years for file sharing (BitTorrent), streaming media (SopCast), VoIP (Skype), decentralized content caching (Corelli), and recently for distributed social networking (safebook). These networks often operate autonomously without external coordination, and the network operational regime is the outcome of spontaneous peer interaction. Understanding the nature of these interactions and predicting the operational regimes of the network as a whole are two primary objectives in distributed networked environments.

In this work, we study a peer-to-peer network for content sharing. Each peer operates simultaneously as server and client, by issuing and satisfying content requests respectively. As client, it generates requests for content. Each request is followed by an advertisement about peers with that content in their possession. 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 then decides on the peers from which it will request the content. In advanced systems, a client may even download different parts of an object from different peers. If time is expanded, the sum of the portions of the various requests of a peer that are addressed towards each server forms the *portion* of the peer request load that is routed towards each server. On the other hand, a peer acts as server and serves received requests from other peers by sending them the requested content based on a service (scheduling) discipline. At the same time, a peer has its own requests served by other peers. In content sharing, a meaningful performance metric is the *average retrieval delay* of requests, namely the average time elapsed from the issue of a download request until the download has been completed.

In structured coordinated systems, a *social objective* that reflects global system operation is defined, such as minimization of total average delay, and optimization-driven algorithms delineate the means to achieve this objective through global coordination. However, peer-to-peer networks may have no central operator and usually do not adhere to global coordination towards a global social objective. Instead, peers interact autonomously and spontaneously with each other. Thus, they selfishly decide on their controls (strategies) so as to pursue their individual performance objective—in our case, minimize their average retrieval delay. Each peer has a *client strategy set consisting of feasible request load splits towards servers*, and *a server strategy set which is formed out of possible service disciplines on received requests*. By a certain client strategy, a peer affects the total loads at different servers, and hence the delays of peers that seek service at these servers. Also, by a certain server strategy a peer affects delays of peers that seek service from it. Peers compete for server capacity resources and selfishly readjust their strategies to improve their own delay. In such non-cooperative peer interactions, it makes sense to study the resulting stable operating points, namely peer strategies from which unilateral peer deviation does not improve performance. These operating points are the *Nash equilibrium points (NEPs)* of a non-cooperative game. The social objective serves as a benchmark, quantifying the performance loss of selfish interaction compared to the social optimum. In this work, we consider *not only* best response request load splitting *but also* service strategies, we investigate their impact on average retrieval delay and characterize the resulting equilibrium points.

1.1. Our contribution

Our framework comprises small to moderate-sized swarm networks consisting of nodes that engage in file sharing transactions. In that sense, the impact of a peer strategy on others is non-negligible. The starting point of our models is the *atomic splittable framework* for selfish routing over parallel links, where a link stands for a server. Each peer has a client strategy set of feasible request load splits towards servers, and a server strategy set of possible service disciplines on received requests. We consider the performance metric of average retrieval delay.

First, we assume fixed server strategies (First-In-First-Out, or absolute priority) and study the client request load splitting game. Each peer splits its request load to servers so as to minimize its own average delay. Best response updates are used to model iterative peer interaction. We consider peer profiles ranging from selfish or *egotistic*, where peers optimize only their own delay, up to benign or *altruistic*, where peers also take into account incurred delays to others. We derive conditions and properties for NEPs and discuss convergence of the best response to the NEP. For egotistic peers, the NEP is unique regardless of the server strategy, and the sequence of best responses can be implemented based on localized delay measurements. For the altruistic case under FIFO service, there exist multiple NEPs, and the sequence of best response updates converges to a NEP which is also a global optimum. For priority scheduling policies we also observe convergence of best response to one of the NEPs, which however are local minima. We show numerically that the price of anarchy is rather small. For the cases where both egotistic and β-altruistic nodes coexist we characterize the transition from one to multiple NEPs.

Next, we include server strategies in the strategy set. With its service policy, a peer cannot directly affect its own delay, but it can affect its delay at the NEP *after* peers play a client load splitting game. The idea is that a peer i should offer high priority to (and thus attract traffic from) those peers that are served with higher priority than i at other servers, and which cause large delays to i. If peer i knows that j is served with higher priority than i at a server k, it may decide to serve j with high priority. By advertising lower delays to j, peer i may attract some traffic of j that was going toward server k and thus improve its own delay at k. We devise two-level game models, where at first level, a peer selects a service rule in terms of a convex combination of absolute service priorities, and subsequently peers play a client load splitting game. A sophisticated best response for a peer is to select a service strategy such that its delay at NEP *after* peers play the client load splitting game is minimized. This is the best choice available to i at that moment. We also suggest various suboptimal heuristics for selecting a service discipline which require less information to accomplish.

The main contributions of our work can be summarized as follows:

- We formulate the double role of peers as clients (i.e. choosing the server to download from), and servers (i.e. selecting the service strategy) and investigate the impact of both roles on performance.
- We propose a game-theoretic model of peer interaction that captures a range of peer behaviors, ranging from egotistic to altruistic, and we demonstrate how peer behavior affects file retrieval delay.
- We characterize the resulting equilibrium points and quantify the performance loss arising due to selfishness of peers.

The paper is organized as follows. Section 2 provides an overview of related work. In section 3, we present our peer-to-peer model and the assumptions made. Section 4 is devoted to client load splitting games for given service disciplines, and in section 5 we study client-server games, where the service policy is also part of a peer strategy. Section 6 includes extensive numerical evaluation of the proposed schemes and section 7 concludes our study. The proofs of our theorems are provided in the appendix.

2. Related work

Modeling and performance evaluation of peer-to-peer networks can be performed through queueing theory tools under the premise that the server capacity is infinitely divisible. Initial related results on abstract scheduling problems date back to more a decade ago. Indicatively, work [1] considers minimizing the total weighted delay for a system of customers and queues under non-preemptive priority scheduling. The problem consists in i)finding an allocation of customer traffic portions to queues and ii) 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 hence the optimal solution cannot be guaranteed. The case of FIFO scheduling is studied in [2], where the major finding is that, at the optimal solution, the customer types allocated to each server have to be clustered in terms of first and second moment of service times. However, these works do not capture the specifics of peer-to-peer networking, namely the double client-server role of peers and their tendency to selfishness.

In the context of peer-to-peer file sharing, fluid models and queueing theory have been extensively used. Work [3] models the collective behavior of peers as a closed queueing network with one queue per object, a queue service rate depending on the number of object replicas, and one queue representing query processing. In [4], the authors study peer selection for downloading and streaming content to minimize required time and monetary cost for downloading, assuming that bandwidth prices are announced. In a similar setting, [5] brings the Internet Service Provider (ISP) dimension into picture and addresses the balance between minimizing the download delay in peer-to-peer communication and minimizing the tariffs paid among different ISPs that carry P2P traffic among themselves. The problem concerns coordinating the trackers in the transient swarms to control the in- and out-flow of P2P traffic from each swarm.

Given that each peer is an autonomous and self-interested entity, game theory appears as a promising tool for modeling behavior and interaction of peers. In this direction, [6] presents a fluid model for BitTorrent which analyzes average file transfer time and discusses upload bandwidth control strategies that lead to Nash equilibria, where the utility is the average download rate. Recent work [7] investigates the scenario where peers selfishly select to which other peers they connect. Compared to our approach, the considered strategy space is significantly smaller, since the exact allocation of requests to the connected peers is not captured. Besides, all these works ([3] - [7]) assume fixed service strategy.

To the best of our knowledge, only limited works have considered the server role of peers. In [8] the authors study unilateral or bilateral network formation games, where peers individually open up connections on multiple paths or mutually agree on setting up paths respectively. [9] addresses the problem of optimal multimedia P2P content exchange among a set of peers. The emphasis here is on a resource reciprocation game. In that game, the strategy of each peer amounts to appropriate placement of transmission effort (and resources) towards other peers in a group, according to the quality of multimedia content received from those other peers.

In this work, we propose a novel two-level repeated game framework that jointly captures *i*) request load splitting and *ii*) service strategy, since both affect the resulting equilibria. The problem of request load splitting to a set of server peers, faced by a set of client peers that we consider, parallels that of *selfish routing* of several input flows over shared paths, where each path link is associated with a latency function. In a non-atomic setting, each input flow represents a large population of individuals, each of which controls a negligible amount of flow and myopically follows the path of minimum cost. Macroscopically, the total input flow can be split over multiple paths, but the input flow cannot control the route of its constituent infinitesimal individuals. On the other hand, in an atomic setting, each flow is a player which routes its traffic load through only one path. For generic link cost functions, existence of a NEP is guaranteed for the non-atomic case but not for the atomic one unless all input flows are equal to each other (see [10, Chap.18] for a comprehensive survey on routing games and other results on generic and linear cost functions).

Further, in atomic splittable models, each flow is a player which may split its traffic across multiple paths under the criterion of optimizing its own cost. In the seminal work [11], the authors establish uniqueness of NEP for atomic splittable models and for a class of link cost functions with some convexity properties, among which is the average latency per unit 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.

The worst-case ratio between the social objective evaluated at a NEP over that evaluated at the social optimum is called price of anarchy [12]. This is studied in [13] and [14] for routing games with non-atomic players. The price of anarchy is at most 4/3 for linear latency functions regardless of the network topology. Bounds are also derived for latency functions that correspond to $M/M/1$ and $M/G/1$ queue service (which may even grow unbounded), under some assumptions on link load that prevent latency functions from being infinite. More recently, the work [15] shows that the price of anarchy for a non-atomic game with N parallel links, a single input flow, and the class of unbounded latency functions above is N. For the atomic splittable flow case, a bound of 2.618 exists on price of anarchy [16] with linear cost functions but no such bound exists for other types of cost functions. The work [17] provides upper bounds on price of anarchy for non-atomic users whose behavioral profile ranges from selfish to altruistic, depending on whether or not the social welfare appears in their objective, whereas a recent technical report [18] provides a numerical evaluation of various regimes of peer cooperation and non-cooperation.

An important line of works study how to engineer efficient NEPs, a topic that is beyond the scope of this work. The works [20], [21] respectively consider allocating additional capacity to links and controlling some amount of flow through a central manager in the model of [11]. [22] approaches the problem of peer-to-peer content distribution and peer-to-peer services through cooperative game theory. Specifically, appropriate incentive mechanisms are designed that ensure that users contribute resources. The key challenge addressed is that the revenues for the provider (due to operational cost reduction for the system) are shared between the provider and the users according to the extent that each user's participation contributes to the cost reduction. A comparative study of different incentive schemes in P2P networks, like cooperation, payments, repeated peer interaction, intervention, and enforced full sharing can be found in [23].

Nomenclature

$\mathcal{N}$	the set of peers in a swarm
r_i	mean request load of client i (bits/sec)
L_i	mean size of requests by client i (bits/req)
C_j	upload capacity of server peer j (bits/sec)
μ_{ij}	average service rate of client peer i at server peer j (reqs/sec)
λ_{ij}	content request load of peer i addressed to server j
$\rho_{ij} = \lambda_{ij}/C_j$	traffic intensity in server j due to the load of peer i
$\Lambda_j = \sum_{i=1}^{N} \lambda_{ij}$	total content request load addressed towards server j
$\boldsymbol{\lambda}_i = [\lambda_{i1} \dots \lambda_{iN}]$	request splitting strategy of client peer i
$\boldsymbol{\lambda}_{-i}$	load splitting strategy profile of peers other than i

$\mathbf{\Lambda} = [\boldsymbol{\lambda}_1 \dots \boldsymbol{\lambda}_N]^T$	request splitting strategy of the network
$\mathbf{\Lambda}^*$	request load splitting at the NEP
$\mathbf{\Lambda}^0$	request load splitting at network optimum
$\boldsymbol{\pi} = (\boldsymbol{\pi}_1 \dots \boldsymbol{\pi}_N)$	service disciplines of the servers
D_{ij}	average retrieval delay per unit traffic of peer i at server j
$\mathbf{D} = (D_1 \dots D_N)$	vector of average retrieval delays
D_{tot}	total average network delay

3. System model

We consider a set $\mathcal{N}$ of N peers that form a small to medium sized swarm and engage in long-term content exchange transactions. Peers are non-cooperative. No subset of them coordinate or negotiate, and they do not conform to any incentive protocol. Each peer has some content in its disposal, which it may provide to others upon request. At the same time, each peer places content requests for obtaining content objects from other peers. The performance metric for each peer is the average content retrieval delay from other peers, a metric that has been extensively used in the literature for peer-to-peer file sharing applications(e.g. [5, 6, 8, 25])

Content retrieval delay between a content requester (client) peer and a provider (server) peer is the sum of network congestion delay, queueing delay and service delay. The first one is due to congestion conditions at the physical path that connect peers. The second one is due to queueing of various requests at server peers, and the third one is due to the server finite service capacity. In this work, we assume that the bandwidth available at backbone links that connect peers is large enough so that the effect of congestion on retrieval delay is negligible. We take into account the queueing and service delays which arise due to limited access link and upload bandwidth of peers. This is very common in contemporary networks, where upload capacity is much smaller than download capacity (e.g., up to 10 times smaller in ADSL). Though we assume that set $\mathcal{N}$ is fixed, the results extend to dynamic peer arrivals and departures.

3.1. Peers as Clients: Content request load splitting strategies

Each peer i as client is interested in content objects that reside in other peers. Each requested content object m by a peer i may reside in a subset of peers $\mathcal{A}_m \subseteq \mathcal{N} \backslash \{i\}$. In the long run, as the number of requested content objects grows, these become indistinguishable. The stream of content requests can be treated as a divisible fluid, and it can be assumed that content requests for peer i are addressed towards the entire peer set $\mathcal{N} \backslash \{i\}$. Such fluid models have been extensively used to model content exchange in peer-to-peer networks e.g [36].

Under the premise above, each peer i has a content request generation rate from higher layer applications. For modeling and analytical tractability, we

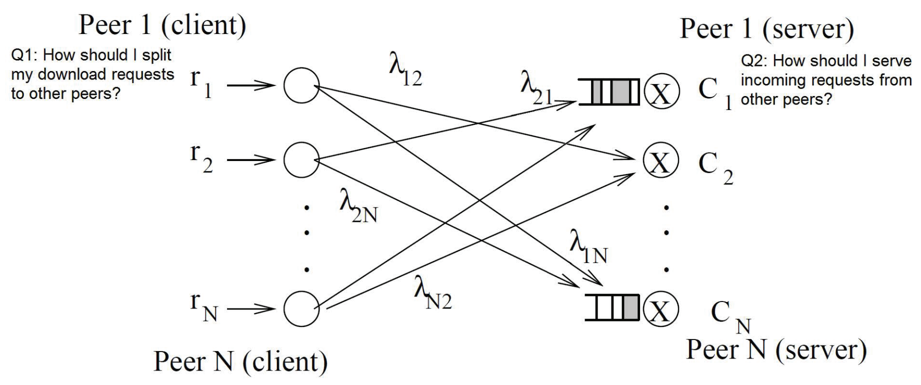

Peer 1 (client)
Q1: How should I split my download requests to other peers?
r 1
r 2
r N
λ12
λ21
λ2N
λ1N
λN2
C 1
C 2
C N
Peer 1 (server)
Q2: How should I serve incoming requests from other peers?
Peer N (client)
Peer N (server)

Figure 1: A peer-to-peer system, with each peer i being *simultaneously* client and server. As client, peer i splits its content request load of rate r_i among peers $j = 1, \dots, N, j \neq i$. As server, peer i uses service capacity C_i to serve incoming request load with a certain service policy.

assume that the request load generation process at peer i follows Poisson distribution with mean r_i bits/sec. The size of a request of peer i is an exponential random variable with mean L_i bits, and thus the request load of r_i bits/sec corresponds to a request rate equal to r_i/L_i requests/sec. Each peer j has fixed service rate (upload capacity) C_j bits/sec. The average service time per request of client peer i at server peer j is $1/\mu_{ij} = L_i/C_j$ seconds per request. Here, we assume $L_i = 1$ for all i, but our formulation can be modified to include different average request sizes.

A content request load splitting strategy for client peer i is a vector $\boldsymbol{\lambda}_i = (\lambda_{ij} : j \in \mathcal{N} \setminus \{i\})$, where $\lambda_{ij} \geq 0$ is the amount of content request load of peer i that is addressed towards server j, and $\lambda_{ii} = 0$, for $i = 1, \dots, N$. The fraction of request load of peer i addressed towards peer j is λ_{ij}/r_i. The load splitting strategy is *feasible* if $\sum_{j=1}^{N} \lambda_{ij} = r_i$. Denote by $\mathcal{F}_i$ the set of feasible load splitting strategies of peer i. The fluid model above is general enough and captures also the scenario where a peer obtains different portions of a distinct content object in parallel from different peers. A feasible request load splitting strategy for the *network* is a matrix $\boldsymbol{\Lambda}$ with the ith row being $\boldsymbol{\lambda}_i$. Let $\mathcal{F}$ be the set of feasible network load splitting strategies, each of which needs to satisfy the necessary and sufficient condition for server stability, $\sum_{i=1}^{N} \lambda_{ij} < C_j$ for $j = 1, \dots, N$.

Furthermore, we assume that network stability conditions, $\sum_{i=1}^{N} r_i < \sum_{j=1}^{N} C_j$ and $r_i < \sum_{\mathcal{N}\setminus\{i\}} C_j$ for $i = 1, \dots, N$ hold. Let $\boldsymbol{\lambda}_{-i} = (\boldsymbol{\lambda}_1, \dots, \boldsymbol{\lambda}_{i-1}, \boldsymbol{\lambda}_{i+1}, \dots, \boldsymbol{\lambda}_N)$ be a collective load splitting strategy profile of peers other than i. Figure 1 depicts the content request load splitting of clients to servers.

3.2. Peers as Servers: Service Strategies

For a given request load splitting strategy, let $\Lambda_j = \sum_{i=1}^{N} \lambda_{ij}$ be the total content request load addressed towards server j, and let Λ_j^{-i} be the request load at server j from peers other than i, so that $\Lambda_j = \Lambda_j^{-i} + \lambda_{ij}$. For exponentially distributed request size, the service times are also exponentially distributed. Thus, each server j can be modeled as an $M/M/1$ queue with service rate C_j (indicating upload capacity of peer j) that serves incoming requests load Λ_j. The traffic intensity in server j due to the load of peer i is $\rho_{ij} = \lambda_{ij}/C_j$, and the total traffic intensity at server j is $\rho_j = \sum_{i=1, i\neq j}^{N} \rho_{ij}$. The average retrieval delay per unit traffic (bit) of peer i when serviced by peer j, D_{ij}, is the average queueing delay per unit traffic at j, Q_{ij}, plus the average service delay, $1/C_j$.

For each server j, the *work conservation law* says that, given load splits $\{\lambda_{ij}\}_{i=1,\ldots,N}$ to j, the total average amount of work, namely the total average queue length of requests is the same regardless of the service discipline. Thus, if we try to reduce the request queueing delay for a client, the queueing delay of another client will increase. Work conservation holds for the $M/M/1$ queue with preemptive priority service. Work conservation also holds for average retrieval delays. That is, for each server j:

$$\sum_{i=1, i\neq j}^{N} \rho_{ij} D_{ij} = \frac{\Lambda_j}{C_j(C_j - \Lambda_j)} \,. \tag{1}$$

However, the service policy of server peer j affects the individual waiting time of peer i, Q_{ij}. The following service policies are considered:

3.2.1. FIFO Scheduling

For FIFO scheduling policy at server j, the same average retrieval delay is provided to all served clients i, i.e.

$$Q_{ij} = \frac{\Lambda_j}{C_j(C_j - \Lambda_j)} \ , \ D_{ij} = \frac{1}{C_j - \Lambda_j} \,. \tag{2}$$

3.2.2. Preemptive Priority Scheduling

A server that employs a preemptive-resume priority service discipline with given absolute priorities serves incoming requests based on a certain priority ranking. The service of peer request load is interrupted whenever request load from a peer of higher priority arrives, and it is resumed from the point it was interrupted when all higher priority request loads have been served [26]. Besides being a reasonable service model, this policy can be a good approximation for non-preemptive priority scheduling policies in high load conditions with regard to achievable delays.

An *absolute priority service policy* $\boldsymbol{\pi}_j$ at server j is a ranking of client set $\mathcal{N} \setminus \{j\}$. Let $\mathcal{M}_j$ be the set of $(N-1)!$ possible rankings of client peers. For a given ranking applied by server j, let π_j^i denote the order of peer i in the ranking. A peer i is served with higher priority than peer ℓ at server j if

$\pi_j^i < \pi_j^\ell$. For a client i, a server j and some priority service policy, let Λ_j^{i+} be the total load that is served by server j with higher priority than the load of i, namely $\Lambda_j^{i+} = \sum_{k:\pi_j^k < \pi_j^i} \lambda_{kj}$. Let $\boldsymbol{\pi}_{-j}$ be the service disciplines of peers other than j and $\boldsymbol{\pi} = (\boldsymbol{\pi}_j, \boldsymbol{\pi}_{-j})$ denote the ensemble of service disciplines of all peers. Each priority ranking corresponds to a delay vector $(D_{1j}, \ldots, D_{Nj})$, where D_{ij} is the average delay per unit traffic that peer i experiences in server j. This depends on the priority π_j^i that peer i enjoys in server j and it is [27]

$$D_{ij} = \begin{cases} \dfrac{1}{C_j - \lambda_{ij}}, & \text{if } \pi_i^j = 1, \\ \dfrac{C_j}{(C_j - \Lambda_j^{i+})(C_j - \Lambda_j^{i+} - \lambda_{ij})}, & \text{if } \pi_i^j > 1. \end{cases} \tag{3}$$

Note that D_{ij} always depends on λ_{ij} and on loads of peers that are served at server j with higher priority than i, 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 $\Lambda_j^{i+} = 0$.

The set of possible service policies at server j is the set of all possible mixtures of different absolute priorities π_j,

$$\Pi_i = \{a_i^m \geq 0, m \in \mathcal{M}_i : \sum_{m \in \mathcal{M}_i} a_i^m = 1\}\,. \tag{4}$$

The mixture denotes the portions of time where different absolute priority policies are applied. Clearly the entire set of service policies at server j results in achievable delays that form $\mathcal{C}_j$, the convex hull of the set of client delay vectors $\mathbf{D}_m$, each of which corresponds to a specific absolute priority order $m \in \mathcal{M}_j$ of clients. Let $\mathcal{C}_j^*$ be the set of achievable delay vectors by server j for all work-conserving scheduling policies. Then, it can be shown that $\mathcal{C}_j^* = \mathcal{C}_j$ [28]. That is, for any delay vector $\mathbf{D}$ corresponding to a work-conserving policy, there exist positive reals $\{a_j^m\}$, $m = 1, \ldots, |\mathcal{M}_j|$ such that $\mathbf{D} = \sum_{m=1}^{|\mathcal{M}_j|} a_j^m \mathbf{D}_m$, with $\sum_{m=1}^{|\mathcal{M}_j|} a_j^m = 1$. Namely, any delay vector is achievable through an appropriate convex combination (mixture) of delay vectors $\{\mathbf{D}_m\}$, each of which corresponds to an absolute priority order m.

3.3. Retrieval Delay Metrics

The proposed framework is generic enough and hence enables modelling and analysis of peer-to-peer systems at different levels of abstraction. Initially, we devise a fluid approach where average retrieval delay serves as the performance metric. Next, we demonstrate that a more detailed file-level view is also feasible.

3.3.1. Average content retrieval delay

A macroscopic view of peer-to-peer systems enables us to characterize the long term behavior of the system. In this context, content exchange can be viewed as the service of the peer-generated streams of requests. Then, traffic splitting $\lambda_i j$ corresponds to the portion of file requests generated by peer i that are directed to peer j and we may denote by $D_{ij}(\mathbf{\Lambda}, \boldsymbol{\pi}_j)$ the average retrieval delay per unit traffic experienced by client i at server j. This depends on the load at server j (which in turn depends also on the loads routed by other peers towards j) and on the service discipline $\boldsymbol{\pi}_j$ of server j. In this case, the performance metric that constitutes the objective of client peer i is the *total average retrieval delay*, which depends on the network request load splitting strategy and service disciplines $\boldsymbol{\pi} = (\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_N)$ at servers j,

$$D_i(\mathbf{\Lambda}, \boldsymbol{\pi}) = \sum_{j \neq i}^{N} \frac{\lambda_{ij}}{r_i} D_{ij}(\mathbf{\Lambda}, \boldsymbol{\pi}_j) \tag{5}$$

The collective network performance 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,

$$D_{\text{tot}} = \frac{1}{\sum_i r_i} \sum_{i=1}^{N} r_i D_i = \frac{1}{\sum_i r_i} \sum_{i=1}^{N} \sum_{j \neq i}^{N} \lambda_{ij} D_{ij} \,. \tag{6}$$

3.3.2. Download time

On the other hand, analysis of content exchange at file-level calls for a different approach. In contemporary peer-to-peer file sharing systems like BitTorrent, each file is separated into several chunks and each is requested from a different peer. In this case, r_i refers to the stream of chunks that peer i has to request from other peers so as to download the file under consideration and load splitting λ_{ij} corresponds to the portion of the specific file that is acquired from peer j. Given that downloads from different peers take place in parallel, the performance metric that constitutes the objective of client peer i becomes:

$$D_i(\mathbf{\Lambda}, \boldsymbol{\pi}) = \max_{j \in \mathcal{N} \setminus \{i\}} \lambda_{ij} D_{ij}(\mathbf{\Lambda}, \boldsymbol{\pi}_j) \tag{7}$$

In the following sections, we focus on the average content retrieval delay metric. However, most of derived results also hold for the file-level approach.

4. Client (Request Load Splitting) Games

First, we fix the ensemble of service policies of servers, $\boldsymbol{\pi} = (\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_N)$ and consider the problem of request load splitting of each client. In particular, we study the following cases:

1. All servers employ FIFO service,
2. Each server j employs a given, arbitrary absolute priority scheduling $\boldsymbol{\pi}_j$.

Hence, the strategy set of each peer i consists only of set $\mathcal{F}_i$. We consider best response peer strategy updates which arise naturally in autonomous, spontaneously interacting non-cooperative peers. Each peer i computes the request load splitting strategy $\boldsymbol{\lambda}_i \in \mathcal{F}_i$ that minimizes a performance objective function, call it $U_i(\boldsymbol{\Lambda})$. In doing so, it affects loads at different servers, and hence 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, parameterized by $\beta_i \in \mathbb{R}$,

$$U_i^{\beta_i}(\boldsymbol{\Lambda}, \boldsymbol{\pi}) = \frac{r_i}{\sum_k r_k} D_i(\boldsymbol{\Lambda}, \boldsymbol{\pi}) + \beta_i \sum_{j \neq i} \frac{r_j}{\sum_k r_k} D_j(\boldsymbol{\Lambda}, \boldsymbol{\pi}). \qquad (8)$$

For $\beta_i = 0$, client peer i is selfish or *egotistic* since it attempts to find a strategy $\boldsymbol{\lambda}_i$ that minimizes its own average delay D_i (i.e. first term) and is indifferent to delays caused to others (i.e. second term). The corresponding best response strategy updates are called egotistic. For $\beta_i = 1$, the peer is benign or *altruistic*; when it computes its optimal strategy, it also takes into account the impact of the strategy on delays $D_j, j \neq i$ of other peers besides its own, D_i. In this case, the performance objective of each peer i coincides with the system objective, i.e. $U_i^1(\cdot) = D_{\text{tot}}(\cdot)$. For $0 < \beta_i < 1$, client i's behavior is between the two extremes above. Parameter β_i is private for each peer.

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

$$U_i^{\beta_i}(\boldsymbol{\lambda}_i^*, \boldsymbol{\lambda}_{-i}^*, \boldsymbol{\pi}) \leq U_i^{\beta_i}(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i}^*, \boldsymbol{\pi}) \qquad (9)$$

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

4.1. Egotistic Best Response Strategies

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

$$\min_{\boldsymbol{\lambda}_i \in \mathcal{F}_i} D_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i}, \boldsymbol{\pi})\,. \qquad (10)$$

Given the strategies $\boldsymbol{\lambda}_{-i}$ of peers other than i, and for *any given* ensemble of service policies $\boldsymbol{\pi}$ (e.g., FIFO or priority), it can be verified from (3) and (5) that the average delay $D_i(\cdot)$ of peer i is a convex function of its strategy $\boldsymbol{\lambda}_i$. This establishes *existence* of a NEP in the load splitting game [29]. 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 $\boldsymbol{\mu}_i = (\mu_{ij} : j \in \mathcal{N} \setminus \{i\})$ such that:

$$\frac{\partial D_i(\boldsymbol{\Lambda}, \boldsymbol{\pi})}{\partial \lambda_{ij}} - \nu_i - \mu_{ij} = 0, \;\; j \neq i, \qquad (11)$$

$$\sum_{j \neq i} \lambda_{ij} = r_i, \;\; \text{and} \;\; \mu_{ij} \lambda_{ij} = 0, \;\; j \neq i \qquad (12)$$

with $\mu_{ij} \geq 0$, $\lambda_{ij} \geq 0$ and ν_i the Lagrange multiplier that corresponds to constraint $\sum_{j\neq i} \lambda_{ij} = r_i$. A splitting strategy $\mathbf{\Lambda}^* = (\boldsymbol{\lambda}_1^*, \ldots, \boldsymbol{\lambda}_N^*)$ is a NEP if and only if conditions (11)-(12) are satisfied for all $i = 1, \ldots, N$. Clearly, service policies $\boldsymbol{\pi}$ play a decisive role in the NEP. In general, for each service policy $\boldsymbol{\pi}$, a different NEP emerges for the client game.

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

4.1.1. FIFO Service

Conditions (11), (12) reduce to:

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

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 a FIFO queue service was shown in [11]. Convergence of the sequence of best responses to NEP is shown numerically but not formally, except for the two-user two-link case. Our client load splitting game, although conceptually similar to selfish routing over parallel links, differs in the following. In [11], all links are available to all users, while in our game the subset of links (servers) to which a user (client) can send flow is $\mathcal{N} \setminus \{i\}$. In general, uniqueness of NEP for the case of restricted subsets of links to which each user can split its flow cannot be guaranteed [19]. However, for our game, it turns out we can slightly modify the proof in [11] and can show that *the NEP 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 destined towards 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 KKT conditions for λ_{ij}, peer i finds its best response to the strategy $\boldsymbol{\lambda}_{-i}$ of other peers as the waterfilling solution:

$$\lambda_{ij} = C_{ij} - \frac{\sqrt{C_{ij}}}{\sum_{j\neq i}^{K_i} \sqrt{C_{ij}}} \Big(\sum_{j\neq i}^{K_i} C_{ij} - r_i\Big), \text{ if } j \leq K_i \tag{14}$$

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

$$K_i = \max\{\ell : \sqrt{C_{i\ell}} \geq \frac{(\sum_{j\neq i}^{\ell} C_{ij} - r_i)}{\sum_{j\neq i}^{\ell} \sqrt{C_{ij}}}\}. \tag{15}$$

The machinery of best response goes as follows. Strategy update by each peer takes place once in a given time interval. The interval can be taken to be large enough so that the condition above is satisfied. Synchronism in updates of peers is not required. At interval n, a peer i measures average delay per unit flow, $D_{ij}^{(n)}$ that it experiences at each server peer j. This can be measured easily through certain fields in request and received data packets that contain time instants when the request load packets were released and when content packets were received, as well as the identities of servers. Peer i also knows its strategy $\boldsymbol{\lambda}_i^{(n)} = (\lambda_{i1}^{(n)}, \ldots, \lambda_{iN}^{(n)})$ and it can deduce $C_{ij}^{(n)}$ as $C_{ij}^{(n)} = \lambda_{ij}^{(n)} + (1/D_{ij}^{(n)})$. It can then determine its best response, $\boldsymbol{\lambda}_i^{(n+1)}$ at $n+1$ through (14). Peer i relies only 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 unilateral peer deviations.

If $r_i = r$, $C_i = C$ for all i, at the NEP it is $\lambda_{ij}^* = \frac{r}{N-1}$ for all $i \in \mathcal{N}$, $j \neq i$, that is, each peer splits its request load equally to all servers. The client delays at NEP are $D_i(\boldsymbol{\Lambda}^*(\mathbf{0})) = 1/(C-r)$ for all i. In general, client delays at NEP are not equal to each other. Note that the unsplittable one-to-one assignment, where each client sends all load r to one server, and each server receives the load of one client is not NEP.

4.1.2. Priority Service

Consider a *given* network service policy $\boldsymbol{\pi}$ other than FIFO, so that service policy π_j at server j is an absolute priority ordering. This is different than FIFO, since now each client experiences different cost (average delay per unit traffic) at different servers due to different service priorities at them. For each client i and server j define $C_{ij}^+ = C_j - \Lambda_j^{i+}$ to be the capacity of server j minus the peer load routed to server j, which is served with higher priority than i. Again C_{ij}^+ depends only on $\boldsymbol{\lambda}_{-i}$. Note that it may be $\pi_i^j < \pi_i^k$ but $C_{ij}^+ < C_{ik}^+$, for some servers j, k. That is, although peer i may enjoy higher priority at server j than at k, it may be more beneficial for i to allocate more load to k rather than j, since the aggregate higher priority traffic at server j may be more. The delay expression for fixed absolute priority service is different than that for FIFO. However, we can prove uniqueness of the NEP even for this general case. Thus, we have:

Theorem 1. *For any network service policy $\boldsymbol{\pi}$, the NEP for the client request load splitting game is unique.*

PROOF. The proof is presented in the appendix.

To compute its best response, each peer i needs to know capacities C_j of servers j. This information is available in contemporary peer-to-peer systems. Similarly to FIFO, at each iteration interval n, a peer i can measure $D_{ij}^{(n)}$ from different servers j. Since it knows $\lambda_{ij}^{(n)}$, it can find $C_{ij}^{(n)+}$ 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

$\boldsymbol{\lambda}_{-i}^{(n)}$ of others as:

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

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

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

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

4.2. Altruistic Best Response Strategies

We now consider altruistic peers, i.e. $\beta_i = 1$ 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 other peers. For given strategy profile $\boldsymbol{\lambda}_{-i}$, the best response policy of altruistic peer i emerges as the solution of:

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

4.2.1. FIFO Service

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 give:

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

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

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

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 $C_{ij}^{(n)}$ 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 Server Capacities

If $C_i = C$ for all i, the global problem can be also expressed as:

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

For this special case, the following theorem characterizes the limit points of the sequence of best response updates.

Theorem 2. *Given that $C_i = C$ for all i, any limit point of the sequence of best response updates, $\{\boldsymbol{\Lambda}^{(k)}\}_{k=1,2,\ldots}$, is a NEP of the altruistic client load splitting game and an optimal solution to the global problem described by (22).*

Proof. The proof is presented in the appendix.

The altruistic game leads to *multiple* NEPs $\boldsymbol{\Lambda}^*$ that all minimize $D_{\text{tot}}(\cdot)$, and all NEPs induce the same server load vector $\mathbf{v} = (\Lambda_1^*, \ldots, \Lambda_N^*)$. Vector $\mathbf{v}$ has some interesting properties.

First we give a few definitions. For an N-dimensional vector $\boldsymbol{\alpha}$, let $\boldsymbol{\alpha}'$ denote the vector with the entries of $\boldsymbol{\alpha}$ arranged in decreasing order, i.e $\alpha_1' = \max_i \alpha_i$, etc. Vector $\boldsymbol{\alpha}$ is called *lexicographically smaller* than vector $\mathbf{b}$, if either $\alpha_1' < b_1'$ or, for some i, $1 \le i < N$ it is $\alpha_j' = b_j'$ for $1 \le j \le i$ and $\alpha_{i+1}' < b_{i+1}'$. A vector $\boldsymbol{\alpha}$ is called *more balanced* than $\mathbf{b}$ if $\sum_{\ell=1}^{i} \alpha_\ell' \le \sum_{\ell=1}^{i} b_\ell'$, for all $i = 1, \ldots, N$.

The server load vector $\mathbf{v}$ at the NEP is *the most balanced* load vector, since it minimizes $\sum_{i=1}^{N} G(\Lambda_j)$ where $G(\cdot)$ is any nondecreasing strictly convex function [30], [31, Theorem 7]. Furthermore, $\mathbf{v}$ is the *lexicographically minimal* vector.

If $r_i = r$ for all i, the optimal average delays of all clients are all equal to $1/(C-r)$.

Unequal Server Capacities

The proof outlined for the previous case cannot be applied here.However, the sequence of best response updates is in essence a sequence of Gauss-Seidel iterations for problem (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 $D_{\text{tot}}(\cdot)$ [32, pp.219-221] and also NEPs of the altruistic game.

4.2.2. Priority Service

For a given service profile $\boldsymbol{\pi}$ other than FIFO, altruistic best response updates by peer i are optimization problems that are convex in peer i's strategy $\boldsymbol{\lambda}_i$. However, now the expression for delay becomes more complicated and may not be jointly convex in $\mathbf{\Lambda}$. In its best response, each peer i minimizes its own delay plus 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. A Gauss-Seidel descent argument guarantees the existence of limit points of the best response sequence. The limit points of best response updates are again NEPs of the game, but they are now local minima of $D_{\text{tot}}(\cdot)$. Our intensive numerical experiments indeed verify the existence of multiple NEPs with suboptimal performance, which are limit points of the best response sequence.

4.3. Best Response Strategies for Heterogenous Swarms

The assumption of selfishness has been repeatedly questioned by economists and psychologists due to innate altruism observed in human beings ([34],[35]). Additionally, the lack of any incentive mechanism makes the scenario of heterogenous peers, i.e. peers characterized by different levels of altruism, the most common in practice. Thus, a swarm may consist of some egotistic and some altruistic peers, or even peers exhibiting behavior between the two extremes. We call the latter β-altruists with $0 < \beta_i < 1$.

Our previous analysis can be straightforwardly extended to capture such scenarios, with the difference that uniqueness of NEP cannot be guaranteed. However, our numerical results indicate that *any swarm including at least two fully altruistic peers has multiple NEPs*, since the existence of the fully altruistic peers guarantees that equivalent traffic splitting strategies exist. For swarms consisting only of β-altruists, i.e. $\beta_i = \beta \ \forall i \in \mathcal{N}$, we have identified that when the FIFO service discipline is applied we have a unique NEP for any $\beta < 0.5$, whereas for the priority service the transition to multiple NEPs can occur even for smaller β values.

The cases of $\beta_i > 1$ and $\beta_i < 0$ may also arise in a network, indicating benign and malicious users respectively. Finally, a user may exhibit different behavior against different peers, i.e. peer i may be altruistic regarding the delay of user j and egotistic towards another peer k. Such scenarios are beyond the scope of this work and thus not considered here.

5. Client - Server Games

We now consider the case where peers can change their service strategy in addition to the client request load splitting strategy. Our objective is to devise game theoretic models that capture the joint client-server strategy space of peers. We consider egotistic peers. A change of service rule at servers leads to different advertised delays to clients, who in turn need to readjust their request load splits in a best response fashion, so that they minimize their delay for the new service regime. In section 4, a peer could affect its delay by controlling the portions of content request load addressed to different servers. But, how can a peer really affect its own delay by altering its service strategy?

Although a peer cannot affect its delay directly by changing its service policy, it may select a service policy such that the NEP *after* the next client request load splitting game is most beneficial for the peer in terms of delay. Suppose that at a NEP, peer i observes large delay D_{ij}^* at some peer j, and 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 service discipline. At the next round of egotistic load splits, peer i will attract part of the request load of k, since k will selfishly split its request load toward peers that advertise low delays. As a result, peer i will have less load of higher priority of k to confront at server j. Hence, delay D_{ij}^* (and therefore D_i^*) will be reduced at the resulting NEP .

In Section 4, we defined the NEP for the client load splitting game for given server strategy profile $\boldsymbol{\pi}$, call it $\boldsymbol{\Lambda}^{\boldsymbol{\pi}}$. We now generalize this definition as follows. A client load splitting strategy and server 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 for all $\boldsymbol{\lambda}_i \neq \boldsymbol{\lambda}_i^*$, $\boldsymbol{\pi}_i \neq \boldsymbol{\pi}_i^*$, and all $i \in \mathcal{N}$, $D_i(\boldsymbol{\lambda}_i^*, \boldsymbol{\lambda}_{-i}^*, \boldsymbol{\pi}_i^*, \boldsymbol{\pi}_{-i}^*) \leq D_i(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i}^*, \boldsymbol{\pi}_i, \boldsymbol{\pi}_{-i}^*)$ or equivalently, if

$$D_i(\boldsymbol{\Lambda}^{(\boldsymbol{\pi}_i^*, \boldsymbol{\pi}_{-i}^*)}, \boldsymbol{\pi}_i^*, \boldsymbol{\pi}_{-i}^*) \leq D_i(\boldsymbol{\Lambda}^{(\boldsymbol{\pi}_i, \boldsymbol{\pi}_{-i}^*)}, \boldsymbol{\pi}_i, \boldsymbol{\pi}_{-i}^*) \tag{23}$$

for all $\boldsymbol{\pi}_i \neq \boldsymbol{\pi}_i^*$, and all $i \in \mathcal{N}$. That is, any change in the service strategy of any peer i will result to a client game whose NEP is characterized by larger average delay for i than the current one $D_i(\boldsymbol{\Lambda}^{(\boldsymbol{\pi}_i^*, \boldsymbol{\pi}_{-i}^*)}, \boldsymbol{\pi}_i^*, \boldsymbol{\pi}_{-i}^*)$.

5.1. Service strategy selection in Client-Server Games

We devise two-level games to capture the double client-server role of a peer. Peers take turns and the game goes as follows. At each iteration, a peer determines a best response *service* strategy and announces it to all. This service strategy is found so that the delay of that peer at the NEP *after* peers play the client game is as small as possible. Subsequently, peers play a client load splitting game based on this service profile and perform best response request load splitting updates until convergence to the (pre-computed) NEP. In the next round, another peer adjusts its service policy, and peers again play the load splitting game based on the new ensemble of server strategies, and so on.

The model we consider is reminiscent of an extended version of two-level Stackelberg games [33]. In classic Stackelberg games there exists one fixed dominant player, *the leader*, and other players, *the followers.* At the upper layer, the leader attempts to minimize its cost subject to all followers being in competitive equilibrium. After the leader selects its strategy, the followers at the lower layer play their game and reach equilibrium. The initial strategy selection by the leader is done by foreseeing the equilibrium strategy of followers under a given leader strategy and then optimizing the leader strategy. Clearly, the leader strategy affects the follower equilibrium.

In our case, *a leader is a peer that chooses its service policy. The followers are the clients* that interact and reach equilibrium in their content request load splitting game, under a given selected service strategy. The migration from the classical Stackelberg model lies in the fact that leaders vary at each stage and also that leaders are part of the lower level game. Each time a different peer takes turn and selects its service strategy by optimizing its own derived delay

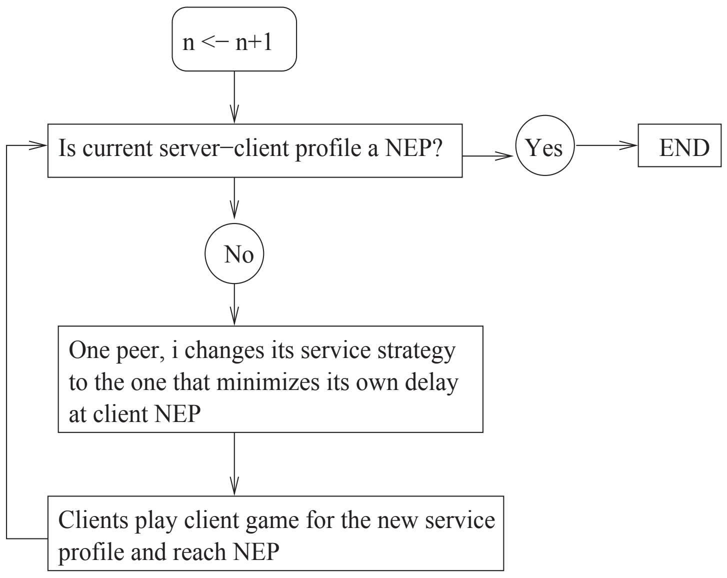

n <− n+1
Is current server−client profile a NEP?
Yes
END
No
One peer, i changes its service strategy to the one that minimizes its own delay at client NEP
Clients play client game for the new service profile and reach NEP

Figure 2: Diagram of two-level game-theoretic server-client interaction.

at the NEP that will emerge after peers play the client game. The sequence of moves in shown in Figure 2.

Given the service policies $\boldsymbol{\pi}_{-i}$ of peers other than i, consider the problem faced by peer i. This consists in finding a service policy $\boldsymbol{\pi}_i$ out of the set of possible ones,

$$\Pi_i = \{a_i^m \geq 0, m \in \mathcal{M}_i : \sum_{m \in \mathcal{M}_i} a_i^m = 1\},$$

such that the delay D_i^* at the NEP after the forthcoming client load splitting game is minimized. Set Π_i encompasses absolute priority rankings of peers in $\mathcal{N}\backslash\{i\}$ (out of set $\mathcal{M}_i$ of the $(N-1)!$ available ones), or *mixtures* of absolute priority rankings.

The delay of peer k at server i is $D_{ki} = \sum_m a_i^m D_{ki}^m$, where, according to (3),

$$D_{ki}^m = \frac{C_i}{(C_i - \Lambda_i^{mk+})(C_i - \Lambda_i^{mk+} - \lambda_{ki})} \tag{24}$$

is the delay experienced by client k at server i for the specific service priority ranking m, and Λ_i^{mk+} the request load of peers of higher priority than k at server i for the priority ranking m.

Peer i needs to go one step ahead and *predict* the client NEP strategy, *assuming that he selects service strategy* $\boldsymbol{\pi}_i$. Namely, it needs to find the NEP splits $\boldsymbol{\Lambda}^{(\boldsymbol{\pi}_i, \boldsymbol{\pi}_{-i})}$, namely the splits $(\lambda_{k\ell}^* : k = 1, \ldots, N, \ell \neq k)$ for *each* peer k to each server $\ell \neq k$ (including server i) for the subsequent client game. To do so, it uses KKT conditions (Appendix A.1) for each pair of client k and server

$\ell \neq k$ and solves them to derive the client NEP as function of $\boldsymbol{\pi}_i$. At round n, a server i chooses service policy

$$\boldsymbol{\pi}_i^{(n)} = \arg \min_{\boldsymbol{\pi}_i \in \Pi_i} D_i(\boldsymbol{\Lambda}^{(\boldsymbol{\pi}_i, \boldsymbol{\pi}_{-i}^{(n-1)})}, \boldsymbol{\pi}_i, \boldsymbol{\pi}_{-i}^{(n-1)}) \,. \tag{25}$$

Thus, peer i wishes to determine the best response *server* strategy $\boldsymbol{\pi}_i^{(n)}$ consisting of the fractions allocated to different priority rankings, $\{a_i^m\}, m = 1, \ldots, |\mathcal{M}_i|$, with $\sum_{m \in \mathcal{M}_i} a_i^m = 1$, so that its delay at NEP,

$$D_i^* = \frac{1}{r_i} \sum_{j \neq i} \sum_{m \in \mathcal{M}_j} a_j^m D_{ij}^{m*} \tag{26}$$

is minimum. Note that fractions $\{a_i^m\}$ are implicit in expressions $D_{ij}^{m*} = C_j \big[(C_j - \Lambda_j^{mi+*})(C_j - \Lambda_j^{mi+*} - \lambda_{ij})\big]^{-1}$, since they affect allocated loads of all peers at different servers j.

Peer i announces strategy $\boldsymbol{\pi}_i^{(n)}$ to all others. After that, peers play the client load splitting game and yield the (anticipated) NEP. Next, at round $n+1$, another peer computes its best response service strategy by finding its own fractions of different priority rankings so as to minimize its own delay at the forthcoming client NEP, then peers play the splitting game, and so on. We assume that each peer chooses his service policy at least once within a given time horizon. A NEP $(\boldsymbol{\Lambda}^{\boldsymbol{\pi}^*}, \boldsymbol{\pi}^*)$ consists of a splitting strategy $\boldsymbol{\Lambda}^*$ and priority fractions $\boldsymbol{\pi}^* = \{a_i^{m\,*} : m \in \mathcal{M}_i, i \in \mathcal{N}\}$ 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 possible priority rankings. Since the strategy space of each peer i, $\Pi_i \cup \mathcal{F}_i$ is compact and convex and the delay D_i^* of peer i at the client NEP is continuous function in Π, the sequence of server best responses delineated by client-server games in Figure 2 converges to an equilibrium.

Although for NEP $(\boldsymbol{\Lambda}^{\boldsymbol{\pi}^*}, \boldsymbol{\pi}^*)$ with $\boldsymbol{\Lambda}^{\boldsymbol{\pi}^*} = (\boldsymbol{\lambda}_1^{\boldsymbol{\pi}^*}, \ldots, \boldsymbol{\lambda}_N^{\boldsymbol{\pi}^*})$ we have that each $\boldsymbol{\lambda}_i^{\boldsymbol{\pi}^*}$ is a continuous function of $\boldsymbol{\pi}^*$ on $\Pi_1 \times \ldots \times \Pi_N$, the computation of the best response server strategy $\boldsymbol{\pi}_i$ by each peer i may require several computations and information that needs to be circulated. To reduce computational burden, peers could consider only pure service strategies, namely absolute priority rankings. In that case, Π_i includes only the $(N-1)!$ possible priority rankings $m = 1, \ldots |\mathcal{M}_i|$. For each such priority ranking m, a peer i runs a provisional client game and computes delay at the NEP, $D_i^*(m)$. Then, it selects the service ranking $m^* = \arg\min_m D_i^*(m)$. However, existence of NEP is not guaranteed in that case.

Though a peer may improve its delay by modifying its service discipline, the best response service strategy may not be applicable due to increased complexity. Thus, in a real system each peer i may perform *heuristically good response* service strategy updates based on heuristic metrics for prioritizing peers, where no a priori computation of the system NEP is required. Peer i may compute these metrics for $j \neq i$ and find an absolute or mixed priority ordering according to them. Such metrics are:

- A metric that quantifies total *hindrance* that peer j causes to i in all servers where j enjoys higher priority than i:

$$\Theta_{ij} = \sum_{\ell:\pi_\ell^j < \pi_\ell^i} \frac{\lambda_{j\ell}}{C_\ell} . \tag{27}$$

  Note that we normalize the load by the server capacity.

- A metric that captures total absolute delay reduction for peer i *per unit of removed flow of peer* j at various servers, given that j is served with higher priority than i at these servers. This metric is given by:

$$H_{ij} = \sum_{\ell:\pi_\ell^j < \pi_\ell^i} \left|\frac{\partial D_{i\ell}}{\partial \lambda_{j\ell}}\right| . \tag{28}$$

Both metrics attempt to quantify the gain of prioritizing a specific peer. Thus, each server peer i should prioritize peers in decreasing order of metrics Θ_{ij} or H_{ij}. That is, the highest priority should be provided to the peer with the highest metric value and so on. Alternatively, i may find a mixture of absolute priorities with priority portions analogous to the metrics above e.g. by assigning the priority fractions such that $\sum_{m\in\mathcal{P}_i^k} a_i^m = \frac{\Theta_{ik}}{\sum_{j=1}^N \Theta_{ij}}$ $\forall k \in \mathcal{N} \setminus \{i\}$, with $\mathcal{P}_i^k \subset \mathcal{M}_i$ the set of priority orderings where peer k enjoys the highest priority and so on. Alternatively, i could choose only a subset of peers, e.g. those with the larger metrics and apply mixtures of priority orderings based on the metrics above.

6. Numerical Results

First, in order to obtain insight on the proposed game models and the impact of various parameters on performance, we simulate using MATLAB a small system of $N = 5$ peers. Further on in this section, we consider also swarms of up to 50 peers, in order to demonstrate scalability of the proposed approach and to observe the impact of swarm size on system performance.

For better tractability of results we first consider a network where each peer i has service capacity $C_i = 256$ Kbps and average request file size $L_i =$ 1024KB. These are typical values for home users (e.g. ADSL upload rate is of the order of 1 Mbps) requesting small files or even chunks of a bigger file. For request load vector $\boldsymbol{r} = [1.6875, 1.6875, 1.1250, 0.5625, 0.5625]$ (in requests/min), the global optimal solution balances the load across servers, i.e. it is $\Lambda_j =$ 1.125 requests per minute for all j. However, this is not always the case. For load vector $\boldsymbol{r} = [5.5125, 0.2625, 0.2625, 0.2625, 0.2625]$, at the optimal solution, the load is not totally balanced, but is distributed across servers as follows: $[1.05, 1.38, 1.38, 1.38, 1.38]$, which is *the most balanced* traffic splitting. Actually in this case peers $2-5$ do not generate enough load to balance the effect of the actions of peer 1.

In order to quantify the performance of the proposed schemes, 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 the 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., it is

$$\text{PoA} = \frac{D_{\text{tot}}(\mathbf{\Lambda}^*)}{D_{\text{tot}}(\mathbf{\Lambda}^0)}. \tag{29}$$

  If several equilibria exist, the worst one, namely the one with the largest delay, is considered.
- **Fairness Index**, F. This is a measure of the dispersion of delay values $D_i(\mathbf{\Lambda}^*)$ and is defined as follows:

$$F = \frac{\left[\sum_{i=1}^{N} D_i(\mathbf{\Lambda}^*)\right]^2}{N\sum_{i=1}^{N} D_i^2(\mathbf{\Lambda}^*)}\ , \tag{30}$$

Notice that under FIFO scheduling optimum delay can be easily calculated, since it coincides with the performance of the altruistic scheme. This is also the optimum of any priority service discipline, if the mean size of requests L_i is the same for all peers. For the general case though, the optimum cannot be always calculated. Instead, the delay of an $M/M/1$ queue of service capacity $C = \sum_{j=1}^{N} C_j$ that serves all the requests provides a lower bound of total average network delay D_{tot}. This bound is generally accurate for small swarms in the heavy load regime.

For comparison purposes, we consider also a one-shot client load splitting strategy, namely a *Proportional Scheme*, where each client allocates its request load to servers in proportion to server capacities, i.e $\lambda_{ij} = \frac{r_i C_j}{\sum_{k\neq i}^{N} C_k}$ for $j \neq i$.

Whenever we refer to absolute priority service, we imply without loss of generality that priorities are assigned according to the index of each peer. Thus, peer 1 always enjoys the highest priority and peer N the lowest one. The system utilization factor, $\rho = \frac{\sum_{i=1}^{N} r_i L_i}{\sum_{j=1}^{N} C_j}$ will also be used in our simulations to determine the request load vector. Finally, whenever we have multiple equilibria, depicted values are the averages of several simulation runs. For the rest of this section, peers have different upload capacities $\boldsymbol{C} = [128, 256, 640, 768, 768]$ (in Kbps), which for an average request file size of $L = 1024$KB lead to a maximum supported load of [5.625,5.625,3.75,1.875,1.875] requests/min and equivalently to a utilization factor of $\rho = 1$.

6.1. Client Games with unconstrained requests

Initially, we consider the scenario of unconstrained requests, i.e. each peer can potentially address a portion of its request load to any other peer. In

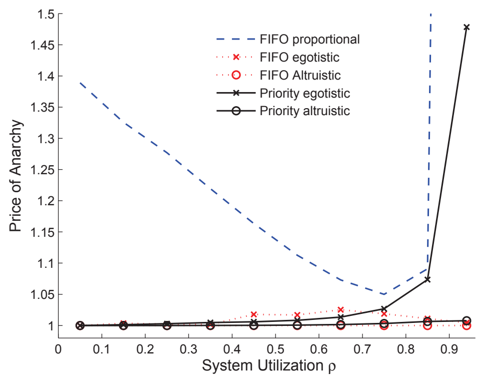
Table 1: Total average retrieval delay (D_{tot}) at NEP for different scenarios

	FIFO			Priority	
	Egot.	Altr.	Prop.	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	74.150	71.457
$\rho = 0.4(\mathbf{\Psi})$	21.328	21.156	27.275	21.734	21.162
$\rho = 0.6(\mathbf{\Psi})$	34.644	34.126	45.882	35.516	34.188
$\rho = 0.8(\mathbf{\Psi})$	72.109	71.188	1958.6	84.557	71.648

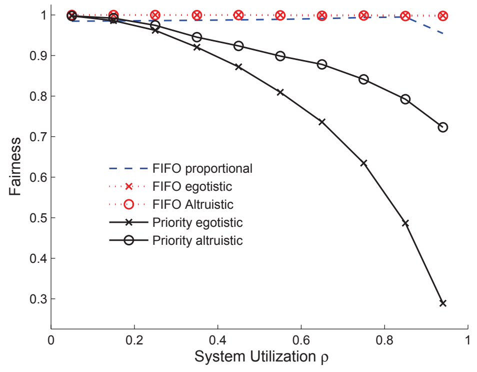
the upper part of Table 1, we show the total delay performance at NEP for different load splitting strategies and different but fixed service strategies as function of the request load. As we have shown in our analysis, the altruistic FIFO game gives the system-wide optimal (minimum delay) performance. Our simulations indicate that the egotistic FIFO game leads to a NEP of near optimal performance (e.g. within $1-2\%$). This can be justified since all client peers, as they contend for the resources of a specific server peer, experience same average retrieval delay. Thus, whenever a peer selects its traffic load splits, it considers the total load in each server and tries to keep it somewhat balanced. Eventually although each peer attempts to minimize its own delay, an underlying coordination of peers occurs. This is not case for the priority egotistic game though. He we considered a swarm of users that are characterized by diverse load and upload capacity profiles. The corresponding delay performance of a uniform swarm would be the order of [26.7, 40, 80] minutes for $\rho = [0.4, 0.6, 0.8]$ respectively. Thus, it is verified that diverse load/capacity profiles are beneficial for the system.

For high enough values of system utilization ρ, we noticed that a high priority peer may select a strategy that, though feasible for itself may be infeasible for the system, leading thus to instability. This is indeed possible, since the delay of a high priority peer does not depend on others' selections, while his actions affect the delay of lower priority peers. Thus, for $\rho > 0.9$, it is quite common that the set of feasible network load splitting strategies becomes empty, since peers' selected strategies do not satisfy necessary and sufficient conditions for stability. In other words, the system converges to an equilibrium point, where high priority peers experience extremely low delays, while lower priority ones experience infinite delays which in turn leads to infinite delay for the entire system as well. This is also indicated in the following figures by the *absence* of the corresponding points for high values of ρ. This phenomenon does not appear in the FIFO egotistic scenario, since all peers work towards guaranteeing stability of each server.

Figure 3(a) depicts the PoA of the client games as a function of ρ. A value of PoA close to 1 means less divergence from the social optimum due to selfishness.



(a) Price of Anarchy metric



(b) Fairness metric

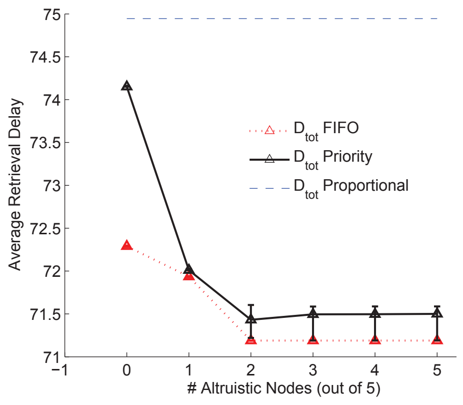
Figure 3: Performance of the network for different values of system utilization factor ρ

For $\rho > 0.94$, no feasible traffic splitting can be found in the priority egotistic scenario. As expected, altruistic games are characterized by smaller PoA values, since each peer considers the effect of its actions on others. All games have a PoA of less than 1.46, i.e. lie within 46% of the optimum, whenever a feasible solution can be found. The proportional scheme performs significantly worse.

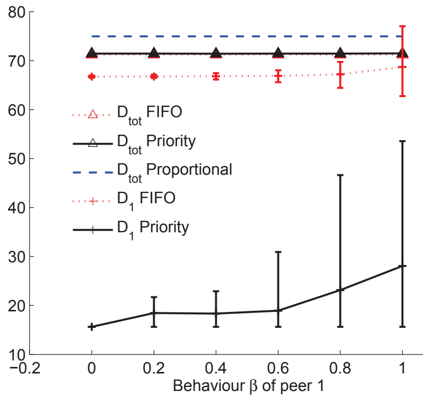
On the other hand, from Figure 3(b), it can be deduced that priority-based schemes become quite unfair as the load increases, since they tend to prioritize some peers at the expense of others. For example, although Figure 3(a) shows that for $\rho = 0.6$, the altruistic games for FIFO and priority service have identical performance, individual delays are quite different. Delay vectors of peers at the NEP are: $\mathbf{D}_{\mathrm{FIFO}} = [32.3, 33.7, 35.7, 38.3, 33.3]$ and $\mathbf{D}_{\mathrm{Pr}} = [15.3, 33.4, 42, 55.7, 55.7]$ respectively. Thus, high priority peers experience extremely low delays, compared to the FIFO case where delays are quite balanced. This difference becomes more evident as the load increases.

6.2. The impact of altruism-parameter β

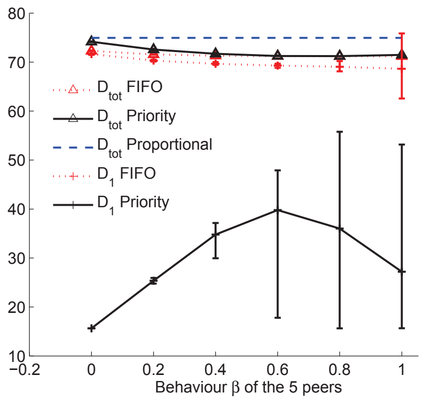
In this section, we consider the impact of peer behavioral profiles, as these are captured by the β parameters, on the resulting equilibria. In the following figures, besides the arithmetic mean of the considered performance metric, we depict the highest and the lowest values out of 2000 simulation runs. This 'variance' serves among others as an indication of the existence of multiple NEPs. Initially, we assume that only two types of peers exist, the egotistic and the fully altruistic ones of $\beta = 1$. In Figure 4(a) we show that the average retrieval delay of a P2P system decreases with the number of altruistic peers. The 'variance' of D_{tot} is a sufficient condition for the existence of multiple NEPs. In general, the existence of at least two altruistic nodes leads to multiple NEPs. This is also evident from the plot for the priority service discipline. In the FIFO case, multiple NEPs appear, but all have the same D_{tot} performance.



(a) Number of altruists



(b) A swarm with a unique β-altruist



(c) A swarm of β-altruists

Figure 4: Average Retrieval Delay vs. Behavior of nodes (β parameter)

Next, we investigate the scenario of users exhibiting behavior between the two extremes of egoism and full altruism. We consider a swarm consisting of a number of fully altruistic peers and some β-altruists. Figure 4(b) depicts the scenario of four altruists and peer 1 characterized by $0 \leq \beta \leq 1$. We plot the D_{tot} metric and the delay of peer 1, i.e. the peer of highest priority. As the level of altruism increases the individual performance degrades but the total delay remains unchanged. Interestingly, increasing altruism causes also the increased variance for both the FIFO and priority schemes.

Figure 4(c) depicts the delay performance of a network of 5 β-altruists. Here, a more significant impact on the total delay can be observed. In addition, the delay of the highest priority peer is no more monotonous in β, with maximum average delay observed for $\beta = 0.6$. Subfigures 4(b), 4(c) indicate that uniqueness of NEP is guaranteed for $\beta < 0.5$ in the FIFO scenario, while the transition to multiple NEPs occurs earlier for the priority scheme.

6.3. Client Games with constrained requests

Previous results have been derived under the assumption of unconstrained requests, i.e. a peer is interested in content of any other peer. However, in reality, this is usually not the case. For example, there may be some rare files that are available only at a small number of peers. Such scenarios arise when a new file appears in the system, or when a file is not popular. We capture such constrained scenarios by a binary $N \times N$ matrix $\mathbf{\Psi}$ of zero diagonal. In essence, this is the adjacency matrix of the overlay graph. Element Ψ_{ij} indicates whether peer i is interested in j's content. Thus, each client peer i may split its request load only to the set of peers $\{j : \Psi_{ij} = 1\}$. For the arbitrary matrix

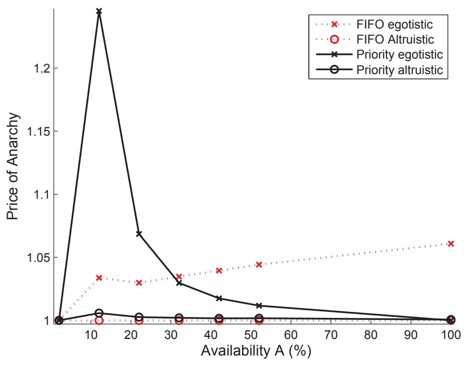
$$\mathbf{\Psi} = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix},$$

we get the delays shown at the bottom part of Table 1. These results are not directly comparable to the previous ones, since they refer to a different, sparser overlay graph. The delay generally increases, since each peer has fewer candidate neighbors to split its traffic to. Note that the slight decrease of delay in some cases is attributed to the fact that the system converges to a different NEP, since the NEP is not unique anymore.

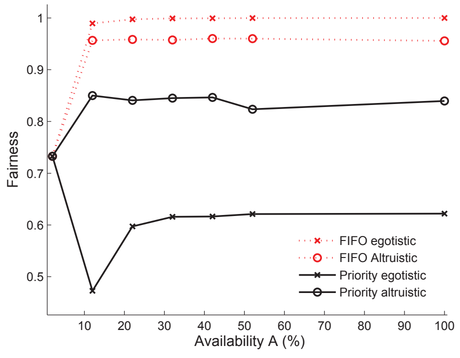
Next, in order to investigate the impact of content availability on system performance, we consider a swarm of $N = 50$ peers. The upload capacity C of each peer is uniformly distributed in [128,1024]kbps and connectivity is randomly generated. The percentage of 1's in matrix $\mathbf{\Psi}$ is indicative of content availability and we denote it with A.

Figure 5 depicts how PoA, fairness and average retrieval delay evolve as content progressively becomes available. We start from a limited connectivity scenario of $A = 2\%$ with $\Psi_{i\,i+1} = 1\ \forall i$, whereas the rightmost point corresponds to the unconstrained case. Figure 5(a) reveals that all schemes perform close to the optimal. There are instances though, where limited content availability causes noticeable performance degradation to the egotistic approaches. Performance degradation of selfish approaches holds in general for scenarios of low to medium availability. Interestingly, at the extreme where only a few peers posses each object, load splitting cannot be performed in many ways, and hence selfish splitting is very similar to the optimal one. The same holds also for the case of full availability, where the load is equally balanced even under selfishness.

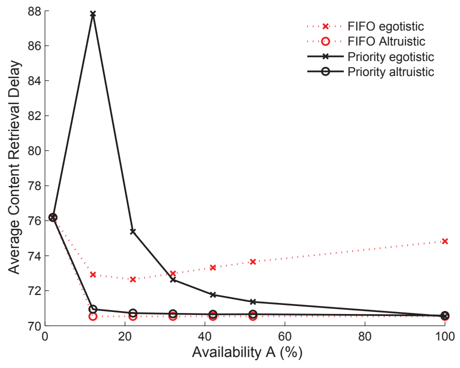
In Figure 5(b), we see that all schemes generally become more fair as availability increases. This is expected, since each peer has more options to split its traffic to, and hence the traffic at NEP becomes more balanced among servers.



(a) Price of Anarchy



(b) Fairness



(c) Delay

Figure 5: Performance vs. Availability A. Higher A indicates greater content availability.

The relative fairness of the various schemes is not affected by content availability, except for the extremely sparse scenarios where the choices are so limited that all schemes perform identically. Figure 5(c) we depict the impact of content availability on average retrieval delay. As availability increases, the average delay decreases significantly. However, under priority service and egotistic behavior, increasing availability may also have the opposite result. Such a scenario arises when high priority peers exploit all the available servers and hence cause significant performance degradation to lower priority ones (as depicted for availability of 12%) .

Notice that availability matrix is a structure that allows us to model evolution of contacts among peers. Each peer needs to know only its own row, i.e the set of peers that it can split its requests to; this information is available at contemporary peer-to-peer networks through content discovery mechanisms. The introduction of matrix $\mathbf{\Psi}$ may also capture *peer dynamics*, i.e. peers entering or leaving the system. When a newcomer k requests an object, a new row is added

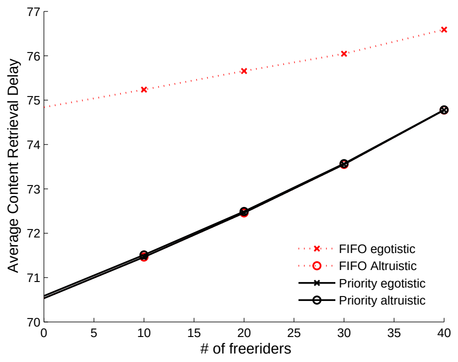


Figure 6: The impact of freeriding on delay performance D_{tot}

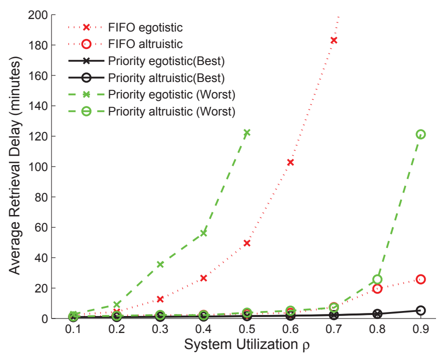
in $\mathbf{\Psi}$. Besides, future requests for k's content will update the matrix by adding 1's in respective positions. The event of a peer leaving the system is equivalent to elimination of its row and column.

In addition, availability matrix enables us to model freeriding behavior, i.e. peers that are not willing to provide any content/upload capacity to others. In particular, for each freerider the corresponding column is a zero vector since no other peer can have part of his requests satisfied there. As expected, we observe in Figure 6 that freeriding causes significant performance degradation for all the schemes.

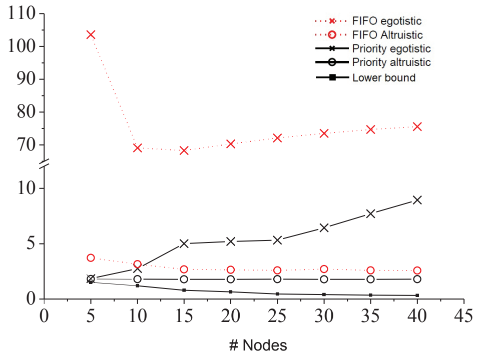
6.4. The impact of service strategy

From the hitherto numerical study, one might claim that the FIFO egotistic approach always exhibits near-optimal performance, since it seems to guarantee both near-optimal delay and fairness. However, if we consider peers of *different average request sizes*, say $\boldsymbol{L} = [0.5, 7, 30, 200, 700]$MB, corresponding to users that are interested in different types of files ranging from simple documents up to movies, we get significantly higher delays at the FIFO NEP than that of the priority schemes (see Figure 7(a)). By prioritizing peers that request small files as $c\mu$ rule dictates [1], we may get significantly lower delay per request. In other words, in order to minimize the mean waiting time per request (or equivalently maximize the number of requests that get served per unit of time), we need to apply the Shortest Expected Processing Time (SEPT) service discipline, i.e prioritize peers in order of increasing L_i. However, this approach may cause significant increase in the waiting time of requests for bigger files.

On the other hand, if we cannot provide incentives to the peers to assign the priorities the proper way, we may experience significant performance degradation. For example, in Figure 7(a) we present the worst case performance, achieved by assigning the priorities inversely to what the $c\mu$ rule dictates. Finally, in Figure 7(b) we demonstrate that all the proposed schemes are also



(a) As an expression of the system utilization factor ρ



(b) As an expression of the number of peers in the swarm for $\rho = 0.6$

Figure 7: Average retrieval delay for requests of different sizes

applicable in larger networks. For comparison purposes, we depict also the lower bound of delay resulting from the single server abstraction of the P2P system. Although the number of participating peers has minor impact on system performance at the resulting equilibria, we observe that the lower bound is only meaningful for small enough swarms.

6.5. Client-Server Games

For the two-level client-server games, we need a small enough network so as to be able to track the two-level peer interactions. Thus, we simulate a network of $N = 3$ peers with capacities $\boldsymbol{C} = [384, 256, 128]$ and low enough loads, $\boldsymbol{r} = [0.6563, 1.0313, 1.1250]$ to guarantee stability. In Table 2, we depict results on total average and individual peer delays at the NEP. All peers are fully egotistic, but for comparison purposes we also provide the NEP delays for the altruistic scenario.

Initially, peers have no information about the network and hence all apply FIFO discipline. The resulting delays at NEP are presented in the second column. Once a peer (say peer 1) has collected sufficient data, it may decide to change its service strategy. Peer 1 determines its best response service policy from 25 and subsequently clients play the load splitting game. The resulting delays at NEP are shown in column BR1. Our simulations show that peer 1 will give absolute priority to peer 3. Thus, at the resulting NEP peer 1 manages to improve his delay (49.23 units) by modifying his service discipline. We obtain similar results when peer 1 selects his new service discipline based on the proposed heuristics (this is not depicted in the table). In the next round, another peer may modify its service discipline until convergence. Such games in general have multiple equilibria to which the client-server best response(BR) updates converge. As anticipated, the Θ_{ij}-based heuristic does not always converge to an equilibrium, while the H_{ij}-based one has a stationary point, albeit different

Table 2: Client (FIFO) games and Client-Server games

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

from the BR equilibrium. In the specific scenario though all games converge, and the performance at the resulting NEPs is depicted in the last three columns.

From the above, it can be deduced that if each peer is allowed to selfishly select its service strategy, the system performance degrades, but interestingly, not much. On the other hand, the performance of the individuals changes significantly, but in general we cannot predict who will finally benefit. Each peer enjoys some performance gain only for limited time, namely until the rest of peers reply by changing their own service disciplines.

7. Conclusion

In this work, we investigated the double client-server role of peers and characterized the NEPs emerging from best response strategy updates. The peer strategy set comprises client request load splits alone, or together with service strategies. Interestingly, we have demonstrated that the service discipline of a peer implicitly determines its own performance.The proposed framework can model swarms of any size, as long as each peer controls a positive, non-negligible amount of request rate. This holds for the initial dissemination phase of a file and the final stages where most seeders have left the swarm, which are the most crucial ones due to the limited availability of content.

In addition, our model captures a range of peer behaviors, ranging from egotistic up to altruistic. Our results reveal how that range is depicted into file retrieval delay. We also dealt with the system collective behavior in terms of equilibria for different populations of altruistic and egotistic peers and we demonstrated that introducing altruism causes a transition from one to multiple equilibrium points. In this work, we have assumed that peer behaviour is determined by their level of altruism. However, the employment of reputation methods for gradually revealing peer profiles and guiding the system to efficient equilibria is interesting topic for future study.

[1] J. Sethuraman and M. Squillante, “Optimal stochastic scheduling in multi-class parallel queues”, *Proc. ACM SIGMETRICS,* 1995.

[2] S. Borst, “Optimal probabilistic allocation of customer types to servers”, *Proc. ACM SIGMETRICS,* 1995.

[3] Z. Ge, D.R. Figueiredo, S. Jaiswal, J. Kurose and D. Towsley, “Modeling peer-peer file sharing systems”, *Proc. IEEE INFOCOM,* 2003.

[4] 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.

[5] P. Parag, S. Shakkottai, and I. Menache, “Service Routing in Multi-ISP Peer-to-Peer Content Distribution: Local or Remote?” *Proc. ACM Gamenets*, pp 353-368, 2011.

[6] D. Qiu and R. Srikant, “Modeling and performance analysis of BitTorrent-like peer-to-peer networks”, *Proc. ACM SIGCOMM,* 2004.

[7] T. Moscibroda, S. Schmid and R. Wattenhofer, “Topological Implications of Selfish Neighbor Selection in Unstructured Peer-to-Peer Networks” *Algorithmica*, vol 61 no. 2, pp 419-446 , 2011.

[8] H. Zhang, G. Neglia, D. Towsley, G. Lo Presti, “On Unstructured File Sharing Networks”, *Proc. IEEE INFOCOM,* 2007.

[9] E. Maani Z. Chen and A.K. Katsaggelos, “A game theoretic approach to video streaming over peer-to-peer networks,” , *Signal Processing: Image Communication* vol. 27, no.5 pp. 545-554, 2012.

[10] N.Nisan, T. Roughgarden, E. Tardos and V.V. Vazirani (*eds*), *Algorithmic Game theory,* Cambridge University Press, 2007.

[11] 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.

[12] C. Papadimitriou, “Algorithms, games and the internet”, *Proc. ACM Symp. Th. of Comput.,* 2001.

[13] T. Roughgarden and E. Tardos, “How bad is selfish routing”, *Journal of ACM,* vol.29, pp.235-259, 2002.

[14] 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.

[15] T. Wu and D. Starobinski, “On the price of anarchy in unbounded delay networks”, *Proc. ACM GameNets,* 2006.

[16] B. Awerbuch, Y. Azar and A. Epstein, "The price of routing unsplittable flow", *Proc. ACM Symp. on Theory of Comp. (STOC),* 2005.

[17] P.-A. Chen and D. Kempe, "Altruism, selfishness, and spite in traffic routing", *In Proc. 9th Conf. Electr. Commerce (EC)*, 2008.

[18] A. Prakash Azad, E. Altman and R. El-Azouzi, "Routing games: From Egoism to Altruism", *INRIA Tech. Report No. 7059,* Oct. 2009.

[19] L. Libman and A. Orda, "Atomic resource sharing in non-cooperative networks", *Telecommun. Sys.,* vol. 17, no. 4, pp. 385-409, 2001.

[20] 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.

[21] 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.

[22] V. Misra, S. Ioannidis, A. Chaintreau, and L. Massoulie, "Incentivizing peer-assisted services: a fluid shapley value approach". *Proc. ACM SIGMETRICS Performance Evaluation Review*, vol. 38, no. 1, pp. 215-226, 2010.

[23] J. Park, and M. van der Schaar,"A Game Theoretic Analysis of Incentives in Content Production and Sharing Over Peer-to-Peer Networks", *IEEE Journal of Selected Topics in Signal Processing*, vol.4, no.4, pp.704-717, 2010.

[24] 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.

[25] I. Koutsopoulos, L. Tassiulas, L. Gkatzikis, "Client and server games in peer-to-peer networks", *In Proc. IEEE Int. Workshop on Quality of Service (IWQoS),* 2009.

[26] D. Bertsekas and R. Gallager, *Data Networks,* Prentice Hall, 1992.

[27] I. Adan, Queueing Theory Course : Lecture Notes, *Eindhoven Univ. of Technology,* available online: http://www.win.tue.nl/∼iadan/que/.

[28] 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.

[29] J.B. Rosen, "Existence and uniqueness of equilibrium points for concave n-person games", *Automatica,* vol.33, no.3, pp.520-534, July 1965.

[30] B. Hajek, "Performance of global load balancing by local adjustment", *IEEE Trans. Inf. Theory,* vol.36, no.6, pp.1398-1414, Nov. 1990.

[31] L. Georgiadis and L. Tassiulas, "Optimal overload response in sensor networks", *IEEE Trans. Inf. Theory,* vol.52, no.6, pp.2684-2696, June 2006.

[32] D. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computation : Numerical Methods*, Athena Scientific, 1997.

[33] H. van Stackelberg, *The Theory of Market Economy,* Oxford University Press, 1952.

[34] S.C. Kolm and J.M. Ythier,"Handbook of the Economics of Giving, Altruism and Reciprocity: Foundations", North Holland, 2006.

[35] H. Gintis,"Moral sentiments and material interests: The foundations of cooperation in economic life", The MIT Press, 2005.

[36] W. Saad, Z. Han, T. Basar, M. Debbah, and A. Hjorungnes, "A Coalition Formation Game in Partition Form for Peer-to-Peer File Sharing Networks". *Proc. IEEE GLOBECOM* pp. 1-5. 2010.

Appendix A. Proof of Theorem 1

Let $\mathbf{\Lambda}$, $\tilde{\mathbf{\Lambda}}$ be two NEPs. We will prove that $\mathbf{\Lambda} = \tilde{\mathbf{\Lambda}}$. The KKT conditions for $\mathbf{\Lambda}$ for the average delay imply that

$$\frac{C_j}{r_i(C_j - \Lambda_j^{i+} - \lambda_{ij})^2} = \nu_i, \text{ if } \lambda_{ij} > 0, \qquad \text{(Appendix A.1)}$$

and $C_j/[r_i(C_j - \Lambda_j^{i+})^2] \geq \nu_i$, if $\lambda_{ij} = 0$, for $i \in \mathcal{N}$. Similar conditions (with different multipliers $\tilde{\nu}_i$) hold for $\tilde{\mathbf{\Lambda}}$.

Similarly in rationale with [11, Th.2.1], we prove that for each client i, and for each $j \neq i$,

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

Next, the proof differs from [11]. 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 $i \in \mathcal{N}'$, $i \notin \mathcal{N}'$ respectively, it is:

$$\sum_{j \in \mathcal{N}_k^1} \tilde{\lambda}_{ij} = r_i - \sum_{j \in \mathcal{N}_k^2} \tilde{\lambda}_{ij} \leq r_i - \sum_{j \in \mathcal{N}_k^2} \lambda_{ij} = \sum_{j \in \mathcal{N}_k^1} \lambda_{ij} \qquad \text{(Appendix A.2)}$$

$$\sum_{j \in \mathcal{N}_k^1} \tilde{\Lambda}_j^{k+} = \sum_{j \in \mathcal{N}_k^1} \sum_{i : \pi_i^j < \pi_k^j} \tilde{\lambda}_{ij} \leq \sum_{j \in \mathcal{N}_k^1} \sum_{i : \pi_i^j < \pi_k^j} \lambda_{ij} = \sum_{j \in \mathcal{N}_k^1} \Lambda_j^{k+}. \qquad \text{(Appendix A.3)}$$

The last inequality contradicts the definition of $\mathcal{N}_k^1$, which means that $\mathcal{N}_k^1$ is an 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 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.

Appendix B. Proof of Theorem 2

The problem described by (22) falls within the class of problems:

$$\text{Problem (P)}: \quad \min_{\mathbf{\Lambda} \in \mathcal{F}} D(\mathbf{\Lambda}) = \min_{\mathbf{\Lambda} \in \mathcal{F}} \sum_{j=1}^{N} G(\Lambda_j) \qquad \text{(Appendix B.1)}$$

where $G(\cdot)$ is a strictly nondecreasing convex function of load Λ_j at server j. For problem (P), we have [30, Corollary 4]:

Fact 1. *A feasible network load splitting policy $\mathbf{\Lambda}$ 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 network splitting strategy $\mathbf{\Lambda} \in \mathcal{F}$ and client i, the altruistic best response is a strategy $\mathcal{T}_i\mathbf{\Lambda}$, where $\mathcal{T}_i$ is an operator on $\mathbf{\Lambda}$, with $\mathcal{T}_i\mathbf{\Lambda} = \arg\min_{\boldsymbol{\lambda}_i \in \mathcal{F}_i} D(\mathbf{\Lambda})$, while strategies $\boldsymbol{\lambda}_{-i}$ are kept fixed. Let $\mathbf{\Lambda}^{(0)}$ 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 $\{\mathbf{\Lambda}^{(k)}, k \geq 0\}$ recursively as $\mathbf{\Lambda}^{(k+1)} = \mathcal{T}_{i_k}\mathbf{\Lambda}^{(k)}$. We follow the rationale in [30] with all necessary modifications. We first prove the following lemma.

Lemma 3. *Suppose $C_i = C$ for all i. Let $\mathbf{\Lambda}$, $\tilde{\mathbf{\Lambda}}$ be feasible strategies based on $\mathbf{\Lambda}$ and $\mathcal{T}_i\mathbf{\Lambda}$ respectively. For $\mathbf{\Lambda} \in \mathcal{F}$ and $i \in \mathcal{N}$, define $||\mathbf{\Lambda} - \mathcal{T}_i\mathbf{\Lambda}|| = \max\limits_{j\in\mathcal{N}} |\lambda_{ij}(\mathbf{\Lambda}) - \lambda_{ij}(\mathcal{T}_i\mathbf{\Lambda})|$. Then,*

$$||\mathbf{\Lambda} - \mathcal{T}_i\mathbf{\Lambda}||^2 \leq D(\mathbf{\Lambda}) - D(\tilde{\mathbf{\Lambda}}). \qquad \text{(Appendix B.2)}$$

Proof. Let Λ_j and $\tilde{\Lambda}_j$ be the loads of server j according to $\mathbf{\Lambda}$ and $\tilde{\mathbf{\Lambda}}$. 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,

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

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

For $C_i = C$, $D(\mathbf{\Lambda}^{(k)})$ is monotone non-increasing with k, since $D(\mathbf{\Lambda}) - D(\tilde{T}_i\mathbf{\Lambda}) \geq 0$ from Lemma 3.

Then we show the convergence of the iterative procedure. Since $D(\cdot)$ is a non-negative and continuous function, we have that $\lim_{k\to\infty}(D(\mathbf{\Lambda}^{(k+1)}) - D(\mathbf{\Lambda}^{(k)})) = 0$. In addition, we have $\lim_{k\to\infty} D(\mathbf{\Lambda}^{(k)}) = D(\mathbf{\Lambda}^*)$ for any limit point $\mathbf{\Lambda}^*$ of $\mathbf{\Lambda}^{(k)}$,

$k \geq 0$. From Lemma 3, it follows that $\lim_{k\to\infty} |\mathbf{\Lambda}^{(k+1)} - \mathbf{\Lambda}^{(k)}| = 0$. Since mapping $\mathcal{T}_i$ is continuous for all i, we have $\mathcal{T}_i\mathbf{\Lambda}^* = \mathbf{\Lambda}^*$ for any i and any limit point $\mathbf{\Lambda}^*$. This means that no peer i can benefit from unilaterally deviating from $\mathbf{\Lambda}^*$, hence $\mathbf{\Lambda}^*$ is a NEP for the altruistic splitting game. Since $\mathcal{T}_i\mathbf{\Lambda}^* = \mathbf{\Lambda}^*$ for all i, $\mathbf{\Lambda}^*$ is a stationary point of $D_{\text{tot}}(\cdot)$. Since $D_{\text{tot}}(\cdot)$ is jointly convex with respect to $\mathbf{\Lambda}$, a stationary point of $D_{\text{tot}}(\cdot)$ is a global optimal solution to problem described by (22).