Performance of Peer-to-Peer Networks: Service Capacity and Role of Resource Sharing Policies ⋆

Xiangying Yang, Gustavo de Veciana

ECE Department, University of Texas at Austin, Austin, Texas 78712, USA

Tel: (512)731-0175 (512)471-1573 FAX: (512)471-5532

Abstract

In this paper we model and study the performance of peer-to-peer (P2P) file sharing systems in terms of their 'service capacity'. We identify two regimes of interest: the transient and stationary regimes. We show that in both regimes, the performance of P2P systems exhibits a favorable scaling with the offered load. P2P systems achieve this by efficiently leveraging the service capacity of other peers, who possibly are concurrently downloading the same file. Therefore to improve the performance, it is important to design mechanisms to give peers incentives for sharing/cooperation. One approach is to introduce mechanisms for resource allocation that are 'fair', such that a peer's performance improves with his contributions. We find that some intuitive 'fairness' notions may unexpectedly lead to 'unfair' allocations, which do not provide the right incentives for peers. Thus, implementation of P2P systems may want to compromise the degree of 'fairness' in favor of maintaining system robustness and reducing overheads.

Key words: peer-to-peer, file sharing, service capacity, incentive, fairness

1 Introduction

Peer-to-peer (P2P) architectures for file sharing, e.g., Gnutella, Kazza, eDonkey and Bittorrent, among ad hoc, possibly dynamic, collections of hosts are generating an increasing fraction of the traffic on today's Internet and are reshaping the way new network applications are designed. The idea is to have hosts participate in an application level overlay network enabling signaling, routing, and searching among participating hosts. Once a host locates the document(s)

⋆ Part of this work was presented at IEEE INFOCOM 2004

Email address: {yangxy,gustavo}@ece.utexas.edu (Xiangying Yang, Gustavo de Veciana).

of interest, direct connections are established to mediate their transfer. The key principle is to allow, and, in fact, encourage participating hosts to play dual roles as servers and clients. Using similar ideas, researchers are pursuing work on "grid computing", which would enable not only file sharing, but distributed content delivery, storage, and computation over overlay networks, see e.g., [1] [2].

In this paper we model and analyze the performance associated with file transfers in these types of networks. We study the *service capacity* of P2P networks, i.e., the capability of a P2P network to provide file download service to peers. Our study highlights the service capacity in not only the stationary regime but, also, the transient regime – our interest in the latter is motivated by the, sometimes, exceedingly bursty character of loads on such networks. Moreover, for distributed systems built on peer collaboration, incentives to participate and share have a crucial influence on performance. One way to give peers incentives is to introduce '*fair*' resource allocations whereby the more a peer contributes to the system, the better download performance he can get. For example, current P2P applications mentioned above attempt to use *credit systems* to provide incentives for peers to stay online and 'increase' their upload bandwidth to serve other peers [3]. This is often done by keeping peers' credit history, and based on their contributions (e.g., upload volume), give them different priority in transfers or access to resources of their peers. Such mechanisms are geared at providing incentives for peers to cooperate. As we will see in the sequel their impact on performance may be subtle and significant. In this paper, we will define several such notions of fairness, examine their implications on the performance and discuss how they might be realized.

Related work and paper organization. Most research on P2P systems so far has emphasized design, traffic measurement and workload analysis but not performance evaluation. Early work by [4][5][6] studied traces of P2P applications like Gnutella and Napster. They focused on characterizing the overall P2P system, e.g., request patterns, traffic volume, traffic categorization and properties of shared online content as well as P2P structure and dynamics, e.g., connectivity and peer/host behaviors. Some recent research in the direction of evaluating P2P systems has focused on performance. Peer selection schemes were evaluated in [7], where measurements are used to optimize the selection of good peers so as to improve the overall system performance. A few researchers have used analytical models to study the performance of P2P networks. For example, [8] constructed a model for signaling messages in the Gnutella network and concluded that signaling might significantly compromise performance. The work in [9] is among the first to propose a model for a general P2P system and evaluate its performance. Their model, a closed queuing system, provides basic insights on the stationary performance of a P2P system; among these, the dependence of performance on parameters like the peer request rate and number of peers in the system.

By their very nature, P2P systems are built upon peer cooperation and the importance of ensuring peer incentives for cooperation in a P2P system has attracted much attention recently. In [10] game theory is used to argue that selfish peers have incentives to cooperate, similar to the situation of Prisoner's Dilemma. In [11] n-way exchanges when peers are connected in a ring are studied. The concept of an n-way exchange is proposed in order to facilitate sharing among peers when mutual interests are hard to guarantee between two peers but more likely to

exist among a group of n peers. In [12] a mechanism to control peer selection based on peer contributions to the system, e.g. the more contribution the more flexibility in peer selection, is proposed. This recent work focuses on a qualitatively study of how service policies will impact performance. The recent work in [13] focuses on maintaining 'fairness' in P2P file transfers, for which they showed a Nash equilibrium for distributed resource allocation, which is favorable for improving peer sharing incentives, exists given the global knowledge of peer contributions and bandwidth.

The rest of the paper is organized as follows. In Section 2 we discuss and analyze the performance of a P2P system in both the transient and stationary regimes. We exhibit nice performance scalability properties for such systems when they are subject to large transient, or stationary loads, and study the sensitivity of performance to user behavior characteristics, e.g., cooperativeness and heterogeneity. With a view on providing users incentives to cooperate in Section 3 we consider the role that 'fair' bandwidth allocations could play. This is an attempt at investigating the subtle relationship between 'fairness' based credit systems and performance. Such considerations will enable the design of efficient service policies that can help improve service capacity. Section 4 concludes our paper.

2 Service Capacity of P2P Systems

Our P2P model. We consider an abstract model for P2P file sharing systems, which includes the salient features of current P2P applications. We focus on peers that are sharing/downloading a single file. In doing so we ignore the complicated interactions, e.g., bandwidth sharing among concurrent downloads. Since our focus is on the file transfer performance, we assume each peer has obtained other peers' information, i.e., address and file availability, upon joining the network, e.g., in Gnutella, this information is obtained via a limited broadcast of queries, in eDonkey it is provided by the server a peer connects to, and in Bittorrent it is provided by the tracker of the file. Peers may set up concurrent upload/download streams with other peers using multisource file transmission protocol (MFTP), i.e., a peer can concurrently upload different parts of a file to other peers while it downloads other parts from its peers. Finally peers who finish downloads may randomly choose to leave the system or stay and continue sharing files, in which case we refer to them as 'seeds'. We assume for simplicity no peer leaves the system before completing its download. These features for both the file transfer and user behaviors directly impact system performance and will be analyzed in the sequel.

What is the service capacity of a P2P System? P2P systems are unique in that its service is provided by ad hoc collaborations among peers, which in turn may be heterogenous in terms of their file availability, bandwidth and user behaviors. Since peers play dual roles as servers and clients, the 'service capacity' and performance of such systems is unusual in that they depend on the offered load, i.e., the number of participant peers. Thus, it is not straightforward to define the 'service capacity' of such a system. We will view the service capacity from two perspectives: the system's and the users'. From the system's perspective we will define the following performance metric:

Definition 2.1 *The* "aggregate upload service capacity" *is the overall achievable throughput the system can offer to downloading peers interested in a given document.*

As a system level measure of the resources available to serve peers, aggregate upload service capacity accounts for the effective upload bandwidth of both seeds and downloading peers. 'Effective' here reflects the fact that seeds can upload at their available bandwidth while downloading peers may only be able to upload at a fraction of their available bandwidth because they do not have the complete copy of the document. Now, from the users' perspective, we shall consider the following measure for the service capacity:

Definition 2.2 *The* "per peer download throughput " *is the average download throughput achieved per peer, which might be roughly estimated as the aggregated upload service capacity normalized by the number of downloading peers.*

As a performance metric from the user perspective, per peer download throughput is directly related to the download delays users would see.

Note that since the service capacity evolves over time, in evaluating above performance metrics, we will consider both transient and stationary regimes. The 'transient regime' focuses on the system performance in response to a large burst of requests for a popular file when it is first introduced, or due to daily variations in load. During such periods a relatively small number of peers initially have copies of the file to serve others. However, as replication proceeds the large number of peers requesting the document can be leveraged as servers enabling an exponential growth in the service capacity of the system until the burst of requests is served. Once the intensity of requests stabilizes, the system might enter a 'stationary regime' where the throughput performance of each peer is stable. These two phases are exhibited in the representative trace shown on the left in Fig.1. This trace was obtained by monitoring the per peer download throughput for a given document, in the BitTorrent P2P system [3]. The trace begins with the addition of a new document to a P2P system, then, the solid line tracks an exponential growth in service capacity corresponding to a transient period, and finally the dotted line corresponds to fluctuations in an approximately stationary regime. Note that during the 'stationary regime' the request rate is only approximately stationary. Indeed, not shown in Fig.1 but presented subsequently (see the middle panel in Fig.5) are slow fluctuations in the demands, i.e., number of downloading peers; yet the average performance per peer is fairly stable. As will be discussed in the sequel, during the stationary regime the service capacity tends to scale with the demand. This example exhibits a desirable exponential growth, and subsequent self-scaling of a P2P system's service capacity for a given document. Although in this trace the transient period is relatively short and may not significantly impact the downloading performance of majority of peers, as exemplified on the right panel in Fig.1, for some document types, there is a dramatic fluctuation in the demands following a daily pattern, and hence the majority of peers will experience transient performance.

The service capacity in these two regimes depends on a number of factors:

- data management: a document may be partitioned into various parts permitting concurrent downloading from multiple peers; the granularity and placement of these is critical;

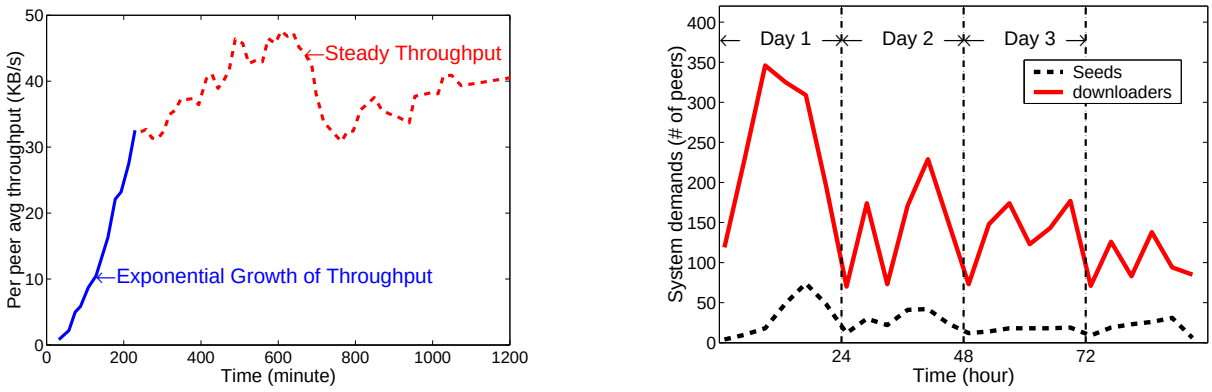

Fig. 1. Two representative traces for the BitTorrent system. On the left two-phases in the evolution of the per peer download throughput versus time after a single document is introduced into a P2P network. On the right a trace exhibiting a significant amount of variability in the number of servers (seeds) and downloading peers (roughly the demands) as a function of time.

- peer selection: the mechanism whereby a peer is selected as a server may take into account load balancing, bandwidth availability, and differentiate among peers who contribute more to the community;
- admission, scheduling and/or bandwidth allocation policy: limiting the number of concurrent downloaders and/or priority scheduling of service or differentiation in the bandwidth allocated among peers;
- traffic dynamics: the request processes for documents along with the dynamics of how peers stay online and/or delete documents.

These factors are interrelated in complex ways. For example, a good peer selection scheme may favor peers that are likely to subsequently stay as servers for the document and thus contribute to the system's service capacity. Multi-part downloads can increase the rate at which files get duplicated while at the same time allowing users to serve as peers for parts of the document they have already obtained prior to completing downloads. Allowing a large number of peers to download from one another may increase the subsequent potential service capacity for a document but may increase delays. Our goal is to explore the interactions among some of these factors and their impact on performance from the perspective of a P2P system's transient and stationary service capacity.

2.1 Transient analysis of service capacity

The purpose of our transient analysis is to investigate how quickly the service capacity of a P2P system can catch up with a burst of demands. This is crucial since popular files are often first introduced by a single peer, and may be subject to large bursts of requests far exceeding the available service capacity. Thus, our goal is to ensure a document is disseminated as quickly as possible to interested peers until the system reaches a stationary regime where the service capacity is commensurate with demands.

2.1.1 *Deterministic model*

We consider the transient regime to be associated with a large, say n, burst of roughly concurrent requests in a P2P system when there is initially a limited number of peers, say 1, able and willing to serve them. As mentioned earlier this may arise when a document is first introduced or for some types of files when the majority of peers experience transient performance due to a periodic request pattern as shown in the right panel of Fig. 1.

The underlying file sharing mechanism in the transient regime is best explained based on a deterministic model. Suppose that $n = 2^k$ users wish to acquire a document which is initially available at one peer. Assume that each peer has a limited upload capacity, say b bps, and network capacity is otherwise unconstrained, i.e., download capacity is assumed to be unconstrained.

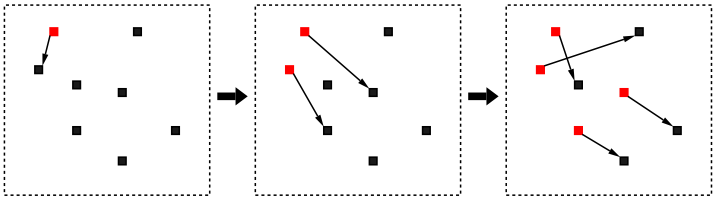

Fig. 2. File sharing in a P2P system.

Suppose the document has size s bits, and a peer can only serve a document once it has been fully downloaded. Thus to serve n requests ns bits will need to be exchanged. It should be clear that a good strategy is to first serve one user at rate b, at which point the service capacity grows to $2b$, and then have these two peers serve additional users, until the n users are served. As shown in Fig.2, under this idealized strategy, peers will complete service every $\tau = s/b$ seconds, at which point the number of peers that can serve the document doubles, leading to an exponential growth of $2^{t/\tau}$ in the number of peers available to serve the document. If the system follows these dynamics the n peers will be served by time $\tau \log_2 n = \tau k$. Thus the 'average' download delay $\bar{d}$ experienced by peers can be computed as follows. Let d_j denote the delay experienced by the jth peer to complete, and note that $2^{i-k}n$ peers complete service at time $(i+1)\tau$, giving an 'average' delay for peers in this transient regime of:

$$\bar{d} = \frac{1}{n}\sum_{j=1}^{n} d_j = \sum_{i=0}^{k-1} 2^{i-k}\tau(i+1) = k\tau - \frac{n-1}{n}\tau = \tau\big(\log_2 n - \frac{n-1}{n}\big) \approx \tau \log_2 n.$$

Hence, although the system sees an initial burst of n requests the average delay seen by peers scales as $\log_2 n$ which is favorable relative to the linear scaling of n one would obtain for a system with a fixed set of servers.

Next, let us consider the benefit of multi-part downloads. Suppose the file is divided into m chunks of identical size. Now, instead of waiting to finish downloading the whole file, as soon as a peer finishes downloading a file chunk it can start to serve it. Intuitively, by dividing the download process into smaller chunks, transfers can be pipelined among participating peers so performance is significantly improved. To illustrate this idea consider the following idealized strategy. We shall track service completions in time slots of size $\frac{s}{bm} = \frac{\tau}{m}$. Suppose the source of

the file sends Chunk 1 to a peer, Chunk 2 to another peer, and so on until it finishes delivering the last Chunk m in slot m. Meanwhile each chunk i is being duplicated in the system. To optimize dissemination, when possible, a peer which currently has a chunk serves another that has not yet obtained any chunk; this can be done until time slot k, at which time every peer in the system has a chunk of the file. As shown in the left panel of Fig.3, at time k, the n peers can be partitioned into k sets A_i, $i = 1,\ldots,k$, with $|A_i| = 2^{k-i}$ and A_i corresponds to peers which have only received the ith chunk. Now consider the $(k+1)$th time slot. Suppose the peers in A_1 transfer Chunk 1 to the $n/2$ peers that have not yet received it. Meanwhile the peers in $A_i, i > 1$, transfer chunk i to a node in A_1 choosing a peer that has at this point only received Chunk 1. Hence, as shown in the left of Fig.3, after the $(k+1)$th time slot, all peers have Chunk 1, $\frac{n}{2}$ peers have Chunk 2 and similarly $\frac{n}{2^{i-1}}$ peers have chunk i. Continuing this process, all chunks are eventually delivered to all users by time slot $k+m = \frac{\tau}{m}(\log_2(n-1)+m)$. This corresponds to a reduction by a factor of m versus the scheme without multi-part downloads. We can compute the average delay $\overline{d^{(m)}}$

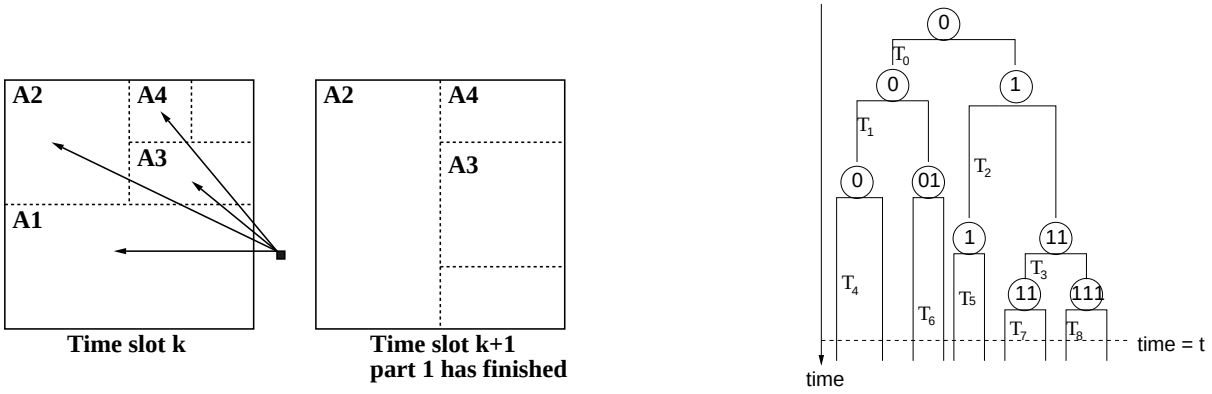

Fig. 3. (1) On the left concurrent multi-part downloads among a set of n peers. (2) On the right branching process model for file replication across a P2P system.

seen by peers in this multi-part download scenario as follows. Since half the peers have received all chunks when Chunk $m-1$ completes duplication across all peers at time slot $k+m-1$ and the rest of the peers will receive chunk m during the last time slot $k+m$. The average delay experienced by peers can be computed as follows. Let $d_j^{(m)}$ denote the delay for the jth peer that completes the download, then

$$\overline{d^{(m)}} = \frac{1}{n}\sum_{j=1}^{n} d_j^{(m)} = \frac{1}{2}((k+m-1)+(k+m))\frac{\tau}{m} = \frac{\tau}{m}\big(\log_2 n + \frac{2m-1}{2}\big) \approx \frac{\tau}{m}\log_2 n.$$

Thus a large m, i.e., small chunk size, enables pipelining of a file transfer and increases the growth exponent of completions by a factor of m, leading to a factor of m improvement in average delay for our transient regime. In practice, however, one must also take into account overheads associated with signaling and or coordinating chunk availability information and realizing the various exchanges. Thus one would expect P2P systems with multi-part downloading to see less aggressive gains in m.

The models in this section provide the basic intuition for the benefits of P2P systems during the transient regime. More rigorous formulations for this type of problem are also discussed in

[14], where a similar scaling property of the delay time for file dissemination in the total number of peers is obtained. Our models here is very idealized; we have assumed there is no congestion in the system, i.e, the upload bandwidth of a peer is not shared by peers requesting different documents, the network is not bottlenecked, and idealized scheduling and peer selection per chunk. This motivates us to consider a stochastic model that captures the variability in serving peers due to congestion and other aspects of real P2P systems. In particular, their departure from the system upon completion. To this end, we propose a branching process model for a P2P system in the transient regime. Our objective is to study the sensitivity of the exponential growth rate to system parameters and peer behavior.

2.1.2 Branching process model

Basic branching process model. Let $N(t)$ denote the number of peers available to serve a given document at time t. Note that the aggregate upload service capacity for this document should be proportional to $N(t)$, see e.g., [9]. We assume that initially there is only one copy of the document in the network, i.e., $N(0) = 1$ with probability 1, and a large number of interested peers. Right panel of Fig. 3 shows a typical evolution of the file sharing process assuming each peer serves another peer one at a time. Thus, initially Peer 0 shares its file with Peer 1. After a random service time T_0, this process completes, and Peers 0 and 1 can now serve other peers. Peer 01 and Peer 11 now download from Peer 0 and Peer 1 respectively and complete this process after some random times T_1 and T_2 respectively. This replication process continues to evolve over time, as long as there are peers still requesting the document. Suppose the times to realize a transfer between peers $T_i,\ i = 0,1,\ldots$ can be modeled as independent random variables with a common distribution, i.e., $T_i \sim T$ where $F_T(t) = P(T \leq t)$ and $E[T] = \tau = \frac{1}{\mu}$. This distribution captures the variability in the transfer time due to congestion, heterogeneity of upload bandwidth, round trip delays etc.

The model we have described corresponds to a standard age-dependent branching process with a fixed family size $v = 2$ at each new generation. General results for the evolution of the mean seed population, i.e., aggregate upload service capacity of our P2P model, for the *supercritical case* can be found in Chapter IV Theorem 3A of [15]. The supercritical case refers to the case where the branching process has mean generation size greater than one and thus the mean population size is expected to increase over time. The following is a restatement of the basic result for a branching process with i.i.d. family sizes whose distribution is that of the random variable V.

Theorem 2.1 *In the supercritical case, where the mean family size per generation satisfies* $E[V] = v$, $v > 1$ *and a non-lattice*[1] *distribution for the regeneration times, with cumulative distribution function* $F_T(t)$, *the expected population of an age dependent branching process for*

[1] A random variable X has a lattice distribution if there is $a > 0$ such that $\mathbf{Pr}\{X \in \{0, a, 2a, \ldots\}\} = 1$. Otherwise, X has a non-lattice distribution.

large time t is given by the following asymptotic result

$$E[N(t)] \sim \delta e^{\beta t}, \tag{1}$$

where $\beta > 0$ is such that $\int_0^\infty v e^{-\beta x} dF_T(x) = 1$, i.e., $d\tilde{F}(x) = v e^{-\beta x} dF_T(x)$ is a probability distribution function, whose mean we denote by $\tilde{\tau}$ and where $\delta = \frac{v-1}{\tilde{\tau}\beta v}$.

Thus for the P2P branching model in the right panel of Fig.3 the aggregate upload service capacity grows exponentially with β and δ as defined in Theorem 1 and where $v = 2$. Below we will use this model to examine the impact of variability and design choices on the transient capacity of P2P systems.

Service capacity has exponential growth under supercritical condition. As expected the service capacity will on average increase exponentially as long as there are sufficient demands in the system. As with the simple deterministic model considered earlier, one would expect that the average delay to serve a large burst of demands n would scale logarithmically in n.

In our branching process model, multi-part downloads also speed up the growth of the transient aggregate upload service capacity and reduce the average delay by a factor m, which is analogous to the previous deterministic model. Furthermore suppose a file is partitioned into m identical sized chunks, and the number of requesting peers is large, after a finite time each chunk has a distinct source peer and subsequently the m chunks are duplicated over m independent branching trees. The asymptotic growth in service capacity for a given chunk $N^{(m)}(t)$ is given by $E[N^{(m)}(t)] \approx E[N(mt)] \sim \delta e^{m\beta t}$, i.e., growth rate increases from β to βm. Given a burst of demands n, the time to complete n downloads is roughly $\frac{1}{\beta m} \ln(\frac{nm}{\delta})$. Note that while this model is quite optimistic, it gives a good sense of how much of a gain multi-part schemes would realize.

Increased parallelism typically decreases the growth exponent. Most P2P applications allow nodes to simultaneously serve a number, say $v-1$, of peers interested in the same document. Thus, in a saturated network, peers may compete for upload bandwidth or CPU resources at other peers resulting in longer service times. As a simple model for systems allowing parallel uploads, consider our branching process model, with a fixed family size $v > 2$. Suppose the distribution for transfer time between two peers is slowed down by a factor $v-1$ causing the mean download to increase by a factor of $v-1$. On one hand, this process will have longer regeneration times. Yet, on the other hand, each time it regenerates, a larger number $v-1$ of peers will have completed downloads and become available to serve others. Thus one might ask whether parallel uploading leads to faster growth rates. If we consider a generation time distribution defined on $[c, \infty)$, where $c > 0$, and it has tail decaying exponentially (or faster) or has a finite mean τ, one can then bound the growth exponent by studying two deterministic branching processes with generation time c and τ, which correspond to upper and lower bounds respectively. The upper bound is obvious since no generation time can be less than c. The lower bound is from the fact that among all distributions, the deterministic generation time gives the slowest growth exponent given the same mean generation time [16]. Note that deterministic generation time is a lattice distribution and strictly speaking one can not use the result in Theorem 1. For a branching process with deterministic generation time τ, one can show that the growth

will be $E[N(t)] = v^{\frac{t}{(v-1)\tau}} = e^{\frac{\ln(v)t}{(v-1)\tau}}$, i.e., for the deterministic case, $\delta^{(d)} = 1$ and $\beta^{(d)} = \frac{\ln(v)}{(v-1)\tau}$. Thus the growth exponent β for the above re-scaled branching process with generation time distribution defined on $[c,\infty)$, is bounded by $\frac{\ln(v)}{(v-1)\tau} < \beta < \frac{\ln(v)}{(v-1)c}$ and is likely to decrease in the parallelism parameter v. Moreover, considering the overheads associated with each transfer and non-linearities in performance degradation, when $v > 2$ the actual performance with increased parallel uploading could be worse. Thus for an application in which peers are unlikely to exit the system, e.g., a P2P based content delivery server system, it makes sense to limit the number of peers that a node serves concurrently in order to obtain the fastest growth in service capacity.

Parallel uploads improve transient capacity when peers are uncooperative. In practice, peers that have completed a transfer may leave the system or delete the file. In contrast to the above result, when peers exhibit such uncooperative behavior, parallel uploading may indeed help achieve higher growth rates. Suppose upon completing a download, each peer deletes the file or leaves the system with probability $1-\zeta$. In this case one must ensure that $v\zeta > 1$ to maintain the branching process in the super critical regime. Otherwise the process becomes extinct, i.e., eventually no peers are available to serve the document, with probability 1 if $v\zeta < 1$, see Chapter IV of [15]. One can further show that it is no longer the case that the maximal growth rate is achieved when v is as small as possible. For example, assuming exponential peer to peer transfer times, consider our peer departure model via an equivalent branching process, for which the transfer time is exponentially distributed with mean $(v-1)\tau$ and the mean generation size is $v\zeta$. According to Theorem 1, the growth exponent β must be such that $d\tilde{F}(x) = ve^{-\beta x}dF_T(x)$, i.e., for the re-scaled branching process $d\tilde{F}(x) = \frac{v\zeta}{(v-1)\tau}e^{-\beta x - \frac{x}{(v-1)\tau}}$, is a probability distribution function, which requires that $\beta = (\zeta - \frac{1-\zeta}{v-1})\mu$ and results in $\delta = 1 + \frac{1-\zeta}{v\zeta-1}$. Left panel of Fig. 4 shows different mean growth trajectories for various choices of v when $\zeta = 0.6$ and $\mu = 1$. When $v = 2$, the service capacity has the smallest growth exponent β. When $v > 2$, β increases, albeit slowly, in v. Thus when some fraction of peers exit the system upon completion of a regenera-

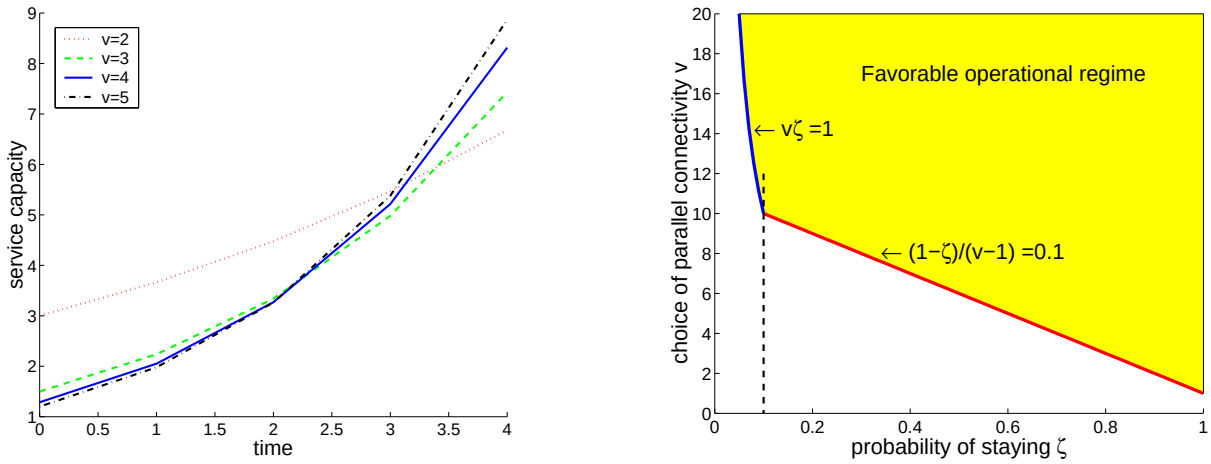

Fig. 4. On the left mean growth in service capacity in a system with uncooperative peers and parallel uploading: various choices of v are shown for $\zeta = 0.6$ and $\mu = 1$ fixed. On the right choice of v to ensure exponential growth with 90% growth rate for a given ζ.

tion, allowing parallel uploads may help assure document availability and improve the overall

exponential growth of the system's service capacity. On the other hand, parallel uploads increase overheads, and for example from the left panel in Fig.4 the improvement in the growth rate is marginal after $\nu = 4$. Hence for design purposes it is interesting to choose only the necessary amount of parallelism, in order to keep overheads low while achieving good service capacity. For example, if we choose a ν such that 90% of the highest growth rate can be achieved while the process is still in the supercritical regime, we require both $\frac{1-\zeta}{\nu-1} < 0.1$ and $\nu\zeta > 1$. As shown on the right panel in Fig.4, when $\zeta < 0.1$, the choice of ν is very sensitive to ζ and a peer generally needs to maintain a relatively large number of parallel uploads to ensure the system is in the supercritical regime; once $\zeta > 0.1$, good choices for ν can be less than 10 and linearly decrease in ζ, i.e., when peers are relatively cooperative, one need only maintain a moderate connectivity and still leverage most of the service capacity.

In summary, the transient regime for a P2P system exhibits excellent scalability properties, in the sense that average delays for service grow logarithmically in the burst size. In addition further speedups can be obtained by enabling multi-part file sharing. When uncooperative peers exist, in order to optimize the growth rate for the transient regime it may be worthwhile to have peers serve others in parallel to ensure a sufficient number of peers remain in each generation.

2.2 Stationary regime analysis of service capacity

For some types of files, after an initial transient phase, the demand may be fairly stationary and sustained. If this is the case it makes sense to consider performance for the stationary regime. Below we briefly describe a model for such a regime and some empirical results. Our goal is to analyze how parameters such as, the offered load and rate at which peers exit the system, impact the average delay to service requests.

We shall model all peers in a P2P system which are interested in, or serving, a particular document and assume that there is always at least one peer serving the document. Suppose new requests follow a Poisson process with rate λ. The system's state is a pair $(x,y) \in \mathbb{N} \times \mathbb{Z}^+$, where x denotes the number of peer requests currently in progress and y denotes the number of peers that have finished downloading and remain in the system, i.e., contributing to the system's service capacity. We further assume that the file is partitioned into chunks, allowing multi-part downloading, thus peers which are in the process of downloading, but already have part of a file, can serve this part to other peers. Thus, a downloading peer also contributes to the system's service capacity, but its contribution is only a fraction η of that of a peer who has already downloaded the full document. The aggregate upload service capacity in the system is thus proportional to the effective number of servers in the system, which we denote by $\mu \times (\eta x + y)$, where μ denotes the service rate for a request at a typical seed. Each time a peer completes downloading the document it becomes a seed in the system, but each such seed may leave the system at rate γ. Thus, the service time for a request at a single peer and the time until a peer that has completed a download leaves the system, are independently and exponentially distributed with rates μ and γ. The evolution for the state of this system can be described by a continuous

time Markov chain with a rate transition matrix Q over the state space $\mathbb{N} \times \mathbb{Z}^+$ given by :

$$\begin{aligned} q\big((x,y),(x+1,y)\big) &= \lambda && \text{new request;} \\ q\big((x,y),(x-1,y+1)\big) &= \mu \times (\eta x + y) && \text{service a peer;} \\ q\big((x,y),(x,y-1)\big) &= \gamma y && \text{exit system.} \end{aligned}$$

We numerically computed the stationary distribution for this Markov chain to find mean number of jobs, servers, and delay for this system. The left panel in Fig.5 exhibits the average delay in the system for a range of parameters; specifically $\mu = 4.0$ $\eta = 0.5$ and the values of λ and γ varied from 4.0 to 12.0 and 2.0 to 8.0, respectively. The average delay depends only on the ratios λ/μ, the offered load and γ/μ, the rate at which peers exit the system, as long as delays are measured in the units of holding times μ^{-1}. As can be seen the mean delay seen by peers in this system model may increase or decrease with the offered load depending on the rate γ/μ at which peers exit. Not shown in the figure, is the fact that this threshold depends on the effectiveness η of document sharing in the P2P system.

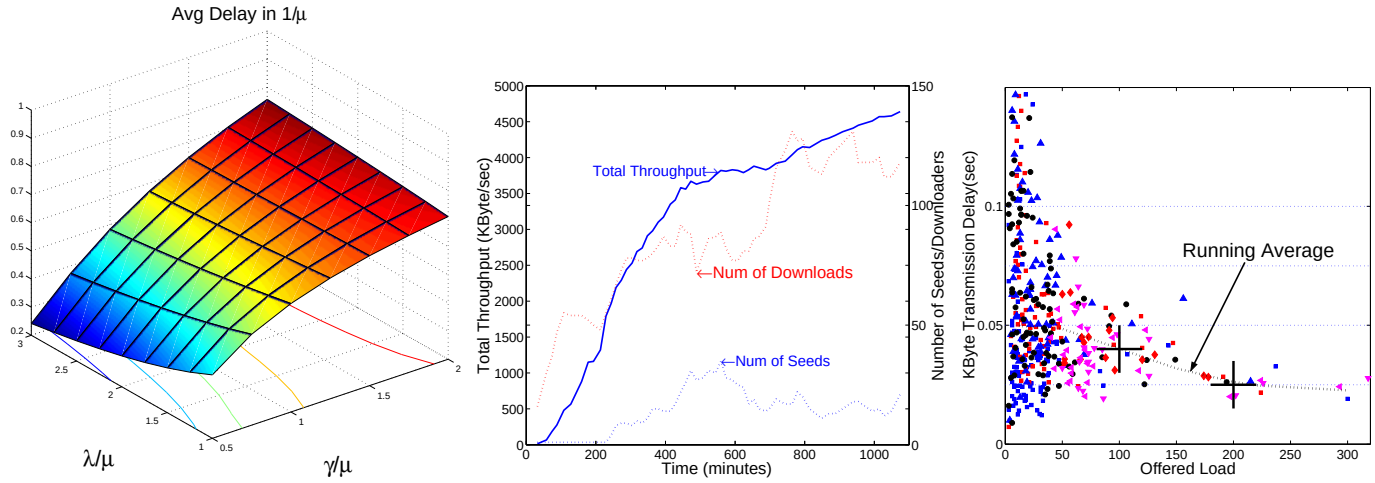

Fig. 5. On the left the average delay for the Markov model with $\eta = 0.5$ for a range of system parameters. At the middle, in the transient regime, the trace measurement for total transfer throughput (i.e., aggregate upload service capacity), number of seeds, and downloaders associated with a file in the BT system. On the right in the stationary regime, the trace measurement of the KByte transmission delay versus offered load(estimated by summing over servers and downloaders associated with a given file in the BitTorrent P2P system.

2.3 Trace Measurements

In order to validate the previous analytical results in a real P2P system, we sampled system data on a Bittorrent(BT) network over a period of several days – see [17] for details on this open-source project. Many aspects of BT's architecture are captured by the models we have been discussing. Specifically a document is introduced by a single peer which is encouraged

insert time	file	size	#seeds	#downloaders	#finished	TX vol	throughput	life
...	...	678MB	2	8	104	75.05GB	265.31KB/s	3 10:43

Table 1
Format of BT trace file

to stay in system for a long period of time, i.e., even after subsequent peers have successfully downloaded the document. BT supports multi-part downloads with a chunk size of roughly 2^{20} bytes, allowing peers to do parallel uploading on a fairly fine granularity. A distributed credit system in BT keeps track of upload and download volumes among peers and tries to achieve fairness in service such that upload and download volumes are equalized.

We collected a trace of network performance reports generated by a program called BT tracker, which has the format exhibited in Table.1. Here *# seeds* refers to the number of peers with complete replicas of a document that are currently on line; *# downloaders* is the number of peers currently downloading the document; *# finished* is the number of completed downloads so far; *TX vol* is the cumulative data volume transferred associated with the given document; *throughput* is the sum of the throughputs seen by peers currently downloading a document, i.e., aggregate upload service capacity; and *life* is the time that has elapsed since the document was first introduced in the system. This data is updated approximately every 5 minutes. Thus a trace permits one to evaluate how the system capacity for an individual file evolves over time.

Transient growth in service capacity. The middle panel of Fig.5, shows the total throughput (aggregate upload service capacity), number of seeds and number of downloads (demands) for a representative (popular) document of size 1310 MBytes over time. We note that in the first 200 minutes or so, the number of seeds stays fixed at 1, although the total throughput increases exponentially. This clearly exhibits the fast increase in service capacity in the initial transient mode, which is enabled by multi-part downloading, i.e., downloading peers are making significant contributions to the service capacity. We note that at around 500 minutes the number of seeds in the system peaks, and subsequently decreases to a steady state of roughly 20 seeds so uncooperative peers are exiting the system quickly. Meanwhile the number of peers downloading the document increases in bursts from 50 to 75 to about 125. The total throughput in the system continues increasing after an initial exponential growth although it tracks the bursty increases in number of downloaders (e.g. the upsurge mentioned at time 500 minutes) slowly instead of exponentially fast. This suggests that the ability of the system to leverage the dynamic service capacity offered by a large number of concurrently downloading peers may not scale as effectively as it did at the start. We suspect this is due to the impact of the credit system as discussed in the sequel. In particular it would give priority to peers that have been in the system and downloading for quite sometime at the expense of new peers (with low credit) and thus reduce the growth rate. This might also be due to a poor scaling of the signaling overheads among larger number of peers.

Impact of offered load on average throughput performance. The data shown on the right of Fig.5 corresponds to a sample of 500 files with file sizes ranging from 400MBytes to 1.1GBytes, for which the system capacity appeared to be in the steady state, i.e., one to four days have elapsed since these documents were introduced to the system and the service capacity and per

peer download throughput should be representative of their popularity/offered loads. For each file, we plot the KByte transmission delay, i.e., inverse of the per peer download throughput (in KByte/sec), versus the number of seeds and downloaders participating in the system. The number of participants is roughly linear in the offered load for a each file, i.e., a proxy for the popularity of the document. For files with less than 50 peers participating in the system, i.e., not very popular, the performance is seen to be quite unpredictable. Intuitively, this big variance is due to the fact that the number of peers is small and heterogeneity among peers is reflected in differences in performance. However, for files that are very popular, the performance improves, albeit slowly, in the number of participants. This matches our analytical results very well, i.e., average delays might go down with the offered load as shown on the left of Fig.5. For example, as marked on the right of Fig.5, when the number of peers is 200, the per peer download throughput is roughly 40KBytes/sec, i.e., the delay to transmit 1Kbyte is 0.025sec and when the number of peers is 100 the per peer download throughput is only about 25Kbyte/sec, i.e., a delay of 0.04sec per 1Kbyte. This improvement as the number of peers grows, is less significant when the number of peers exceeds 200.

Additional empirical data [16] suggest that a P2P system in stationary regime will likely exhibit fairly good performance. As with the transient regime, a significant amount of the service capacity is leveraged from peers that are concurrently downloading the file. It is, however, useful to provide incentives for the latter to stay in the system. Indeed we note that if the rate at which peers leave the system is low (i.e., γ/μ is small), then documents with a high offered load may see improved average performance versus those with lower loads, see Fig.5. This self-scaling characteristic would in practice be highly desirable since it achieves better performance under higher offered loads.

3 Fairness, incentives and their implications on performance

The previous discussion on the service capacity of P2P systems, suggests that the extent to which peers collaborate is crucial to the performance in both the transient and stationary regimes. Thus it is desirable to carefully devise peer incentives to collaborate, i.e., to increase upload bandwidth, share documents that are in demand, and stay in the system longer. One way to improve peer incentives is to reward good contributors to the system by improving their download performance. With this in mind, we introduce a notion of 'fairness' for P2P systems, which would give peers incentives to contribute. The general idea can be described as follows: 'the more a peer uploads the better his download throughput should be', e.g., the download throughput a peer receives should be proportional to his upload throughput. Note that this type of 'reciprocal proportional fairness' among peers can be realized by running a distributed mechanism to allocate resources among peers. This notion of reciprocity distinguishes our notions of 'fairness' from those traditionally considered proportional fairness for resource allocation in networks, see e.g. [18]. We believe such a 'fairness' notion will improve peer incentives in particular when peers can only measure their own upload and download throughput and accordingly determine their contributions to the P2P system. Indeed as mentioned in the introduction, similar ideas have already been implemented in current P2P file sharing applications and are

known as 'credit systems'. In the sequel, we investigate several notions of 'fairness' that fit P2P systems. Surprisingly, some seemingly straightforward criteria have unexpected outcomes and lead to unfair bandwidth allocation, i.e., they may not provide the right incentives to peers and may lead to poor performance. To study this problem, we consider first a stationary regime were all peers are saturated with demands.

3.1 Notions of fairness in stationary regime

Consider the following model for service (bandwidth) allocation. We let N denote a set of peers with total number $|N| = n$ and $\mathbf{x} = (x_{ij} | i \neq j, i, j \in N)$ denote the service provided by each peer, i.e., x_{ij} is a measure of the allocation of resources from i to j. For simplicity let us assume that the access rates are such that each peer is not download constrained but has an upload constraint captured by $\mathbf{b} = (b_i | i \in N)$ where b_i denotes an upload bandwidth constraint for peer i. This constraint may be placed by the communication system or may be artificially placed by the user. Below we will assume that peers are fully connected and share a common interest, i.e., place demands on each other. We let the set N_i denote the neighbors of peer i, which, in the case of fully connected nodes, would be given by $N_i = N \setminus \{i\}$.

Global proportional fairness. The first notion of fairness for P2P systems we consider is one whereby the service, i.e., bandwidth, a peer receives from another is proportional to its own overall contribution to the system. We shall say an allocation $\mathbf{x}$ is *globally proportionally fair* if for all $i, j \in N$ where $i \neq j$ it satisfies

$$x_{ij} = \frac{u_j}{\sum_{k \in N_i} u_k} b_i \text{ where } u_j = \sum_{i \in N_j} x_{ji}.$$

The idea is to allocate outgoing bandwidth from peer i to peer j in proportion to peer j's overall contribution to the system, which is denoted by u_j. If there is demand from each peer, and there are no binding download constraints then $u_j = b_j$ and it follows that

$$x_{ij} = \frac{b_j}{\sum_{k \in N_i} b_k} b_i, \text{ and } d_j = \sum_{i \in N_j} x_{ij} = b_j \sum_{i \in N_j} \left[\frac{b_i}{\sum_{k \in N_i} b_k} \right].$$

Here d_j denotes the *aggregate download bandwidth* that j obtains. Global proportional fairness has several desirable properties. It gives users appropriate incentives by allocating more download throughput to peers who contribute more upload to the system. The globally proportionally fair allocation is unique and gives a positive allocation to all peers. Thus, it ensures the peers in the system are 'fully connected', i.e., when the system is fully loaded any two peers have positive mutual uploads. This is likely to be beneficial to performance particularly when a high degree of sharing and load balancing is desired due to peer heterogeneity. Unfortunately this criterion, while appealing, requires tracking the overall contributions of all peers, which may be hard to verify and implement in a distributed manner.

Peerwise proportional fairness. An alternative notion would be for peers to allocate their upload capacity to other peers based directly on the service they receive from them. This would only require peers to keep track of their downloads from the peers who have served them. This is fairly straightforward to do, and to verify, in a distributed manner. We say that an allocation $\mathbf{x}$ is *peerwise proportionally fair* if for all $i, j \in N$ where $i \neq j$ it satisfies

$$x_{ij} = \frac{x_{ji}}{\sum_{k \in N_i} x_{ki}} b_i. \tag{2}$$

Although such allocations intuitively look 'fair', in general they are not unique and may be, perhaps unexpectedly, unfair. To understand the characteristics of peerwise proportionally fair allocations, we shall consider several examples with increasing generality. First we define an important condition that will be used in the sequel.

Definition 3.1 *Consider n nodes, where $n \geq 3$, ordered by increasing upload bandwidths $b_1 \leq b_2 \leq \ldots \leq b_n$. When the upload bandwidths satisfy*

$$b_n < \sum_{i=1}^{n-1} b_i, \tag{3}$$

we say the peers satisfy the non-dominant condition. Note that (3) implies that $b_j < \sum_{i \neq j} b_i$, $\forall j$, i.e., the upload bandwidth of any single peer does not exceed the aggregate of the others.

The non-dominant condition certainly holds when all peers have roughly the same upload bandwidth, or when there are a large number of peers to compensate wide heterogeneity in the upload bandwidth.

Fact 1 *Consider the three-node case and suppose the non-dominant condition is satisfied. Then there is a* unique, strictly positive *allocation, i.e., with $x_{ij} > 0$, $\forall i \neq j$, which is peerwise proportionally fair:*

$$\mathbf{x} = \left[x_{ij}\right] = \begin{bmatrix} 0 & \frac{b_1+b_2-b_3}{2} & \frac{b_1+b_3-b_2}{2} \\ \frac{b_2+b_1-b_3}{2} & 0 & \frac{b_2+b_3-b_1}{2} \\ \frac{b_3+b_1-b_2}{2} & \frac{b_3+b_2-b_1}{2} & 0 \end{bmatrix}.$$

Fact 1 is straightforward to show by solving the equations (2) associated with the definition of peerwise proportional fairness and constraints. The uniqueness of positive allocations, however, does not follow when there are more than three peers. The following theorem, proved in the appendix, summarizes some of the characteristics of strictly positive allocations.

Theorem 3.1 *Under a peerwise proportionally fair allocation which is strictly positive i.e., $x_{ij} > 0, \forall i \neq j$, the bandwidth allocation must be symmetric $x_{ij} = x_{ji}$. Conversely, symmetric strictly positive allocations imply peerwise proportional fairness. A necessary and sufficient condition for the existence of a convex set of strictly positive peerwise proportionally fair allocations is that their upload bandwidths b_i, $i = 1, \ldots, n$ satisfy the non-dominant condition.*

Note that strictly positive peerwise proportionally fair allocations are thus symmetric, and will satisfy $u_i = d_i = b_i$, i.e., a peer's download bandwidth is equal to the upload bandwidth it contributes to the system. Symmetric allocations are desirable for providing peers incentives in P2P systems because they guarantee that a peer's download performance is equal to its upload. Other than the convex set of symmetric strictly positive allocations, there also exist allocations that are not strictly positive irrespective of the non-dominant condition, i.e., for which some of the allocations among peers are zero and thus peers are not fully connected. In this case symmetry need not hold, and a peer's download allocation need not be equal to its upload contributions. For example, in the 3-node case, the following allocations are also feasible:

$$\mathbf{x} = \begin{bmatrix} 0 & \frac{b_1 b_2}{b_2+b_3} & \frac{b_1 b_3}{b_2+b_3} \\ b_2 & 0 & 0 \\ b_3 & 0 & 0 \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} 0 & b_1 & 0 \\ \frac{b_2 b_1}{b_1+b_3} & 0 & \frac{b_2 b_3}{b_1+b_3} \\ 0 & b_3 & 0 \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} 0 & 0 & b_1 \\ 0 & 0 & b_2 \\ \frac{b_3 b_1}{b_1+b_2} & \frac{b_3 b_2}{b_1+b_2} & 0 \end{bmatrix}$$

Although these are peerwise fair allocations, they would not give users the right incentives and therefore may not be desirable. For example, it is possible that a node contributing a large total upload bandwidth only gets rewarded by a small download throughput from peers with low upload bandwidths. Indeed, when we consider the first allocation above where $b_1 < b_2 + b_3$, peer 1's total download throughput $d_1 = b_2 + b_3$ is larger than the summation of peer 2 and 3's download throughput, i.e., $d_2 + d_3 = b_1$, even if peer 1 has the smallest upload bandwidth. In a large system peerwise proportionally fair allocations can also lead to poor connectivity or even disconnected cliques of peers. This in turn would have a negative impact on the performance since the ability to leverage distributed service capacity would be reduced, and a peer may in practice have trouble finding the file or chunk it requires.

Thus only symmetric peerwise proportional allocations, i.e., where the upload throughput and download throughput between any two peers are balanced, would naturally lead to the right peer incentives. A sufficient condition for that, according to Theorem 3.1, is to maintain full connectivity in the system. However, in a real P2P system, full connectivity is unlikely to hold given the huge number of peers. Without full connectivity in the system, is there still a possibility to maintain balanced upload and download throughput between any two peers? The following corollary proved in the appendix gives a sufficient condition for symmetric peerwise proportional fairness under a relaxed connectivity requirement.

Corollary 3.1 *Let the graph $\mathcal{G}(N,E)$ represent the connectivity of a P2P system in a stationary regime, where N is a set of all peers with $|N| \geq 3$ and E is the set of links among peers such that $x_{ij} > 0,\ \forall i,j \in N$. Suppose we cover $\mathcal{G}(N,E)$ by* **fully connected** *subgraphs $\mathcal{G}_k(N_k,E_k)$, i.e.,$\mathcal{G}(N,E) = \bigcup_k \mathcal{G}_k(N_k,E_k)$ and $E_k = N_k \times N_k$. Suppose node i's bandwidth on a subgraph k, denoted by $b_i^{(k)}$, is such that $b_i^{(k)} > 0$ if $i \in N_k$ and is zero otherwise, and $\sum_k b_i^{(k)} = b_i$. Further, suppose that within each $\mathcal{G}_k(N_k,E_k)$, the non-dominant condition holds for $\{b_i^{(k)},\ \forall i \in N_k\}$. Then the overall allocation to the peers must be symmetric and peerwise proportional fair, i.e., the throughput between any two peers is such that $x_{ij} = x_{ji}$, and $b_i = d_i = \sum_{j \in N_i} x_{ji},\ \forall i \in N$.*

Corollary 3.1 states that overlapping fully connected subgraphs suffices to ensure symmetric peerwise proportionally fair allocations. This suggests a scalable approach to achieving peerwise fairness, which only requires peers to manage a restricted amount of connectivity, and assuming that the non-dominant condition is true when each subgraph includes a sufficient number of peers.

ε-peerwise proportional fairness. The above considerations still suggest these notions may not be suitable for a robust implementation in P2P systems. Maintaining full connectivity within a group of peers was found to be crucial, yet it can be hard in a dynamic context where peers join and leave the system. Another issue, mentioned earlier is that a peerwise proportionally fair allocation need not to be unique, i.e., the system does not have a stable operating state. To avoid problems associated with peerwise proportional fairness, it may instead be preferable to compromise the notion of peerwise proportional fairness to ensure that there exists a unique positive solution and distributed mechanisms to reach it. A simple idea is to ensure that peers persistently attempt exchanges with one another thereby avoiding becoming disconnected. This strategy is used in the BitTorrent P2P system[3]. We can model this by ensuring there is a minimal exchange among all peers, say a persistent allocation of resources which has an long term average bandwidth usage ε. With this in mind we define a bandwidth allocation $\mathbf{x}$ as being ε-*peerwise proportionally fair* if for all $i, j \in N$ where $i \neq j$ it satisfies:

$$x_{i,j} = \frac{x_{j,i} + \varepsilon}{\sum_{k \in N_i}(x_{k,i} + \varepsilon)} b_i.$$

The existence of ε-peerwise proportionally fair allocations follows by the Brouwer fixed point theorem in Chapter 2.6 of [19]. However showing uniqueness does not appear to be straightforward. Our simulations suggest that indeed such allocations are unique even when b_i are not equal. Assuming allocations are unique and the non-dominant condition holds, we can show the following desirable properties of ε-peerwise proportionally fair allocation. First, for ε small enough, one would expect the resulting allocation to be close to one which is strictly positive and satisfies the peerwise proportionally fair requirement. If, however, ε is large a peer would allocate its upload bandwidth equally among its neighbors. Second, a peer's download throughput d_i increases with its upload bandwidth b_i as is the case for the globally proportionally fair allocation. Third, depending on ε, the overall bandwidth allocation will be biased, in that a peer with limited upload bandwidth is likely to get a higher relative aggregate download throughput versus one with a higher upload bandwidth. Clearly the system need not be exactly 'fair' to improve incentives, i.e., our notion of 'fairness' need only loosely require that 'more upload results in better download performance', and we believe this will be sufficient to improve user incentives if we assume a user behavior is only controlled by his/her perceived download performance with respect to his/her upload throughput.

3.2 How to improve incentives under traffic dynamics?

A practical P2P system will be dynamic with peers joining and leaving over time. This complicates considerations on how to provide appropriate incentives. For example, what kind

of resource allocation is ‘fair’ when there are both new peers who just joined the system without any upload/download history and peers that have participated in sharing for some time. Clearly the system should not give little or no services to new peers just because they do not have any history of sharing. Instead, it makes sense to assign relatively high priority to new peers so that they can participate in sharing as quickly as possible and their service capacity can be subsequently leveraged by others. Intuitively, with this approach, the overall system’s service capacity will respond faster to bursty traffic.

In the long run, however, it makes sense to maintain fairness among peers who have participated in sharing based on their uploading history. For example, peers can proportionally allocate resources, e.g., based on the ε-proportional fairness criteria, to others according to their measured uploads. We believe peers should see two stages in terms of getting incentives for sharing from a P2P system, (1) a transient phase, when they first enter the system, in which they get high priority in downloading, and can then participate sharing as soon as possible, and (2) a stationary phase in which ‘fairness’ is maintained based on proportional resource allocation according to their longer term actual uploads.

An important aspect, that has not been explicitly addressed so far, is how to make peers, in particular seeds who are not downloading other files, stay in the system. It is obvious that the reward to a seed’s contribution can only be granted next time the peer starts downloading a new file. Therefore, we believe that it is reasonable to associate a peer’s initial credit in the transient regime with his upload history since a peer’s initial credit is directly associated with his transient performance.

4 Conclusion

In summary, current P2P applications have started to implement service policies, generally referred as *credit systems*, to maintain ‘fairness’ so as to improve peer incentives to cooperate. From our analysis of different notions of fairness and their impacts on the performance, a distributed implementation to realize fairness is not straightforward. Unexpected allocations may result. Moreover, system dynamics further complicate the interaction between performance and timescales on which incentives are realized. A credit system should be carefully implemented in order to effectively leverage the service capacity of peers and do so in a manner that is consistent with both transient and stationary regimes. Indeed one of the key features and advantages of P2P systems is their ability to respond to bursty offered loads by leveraging the capacity of ongoing downloaders. It would be undesirable for mechanisms that provide incentives for peers to collaborate to reduce their capability to do so.

5 Appendix

Proof of Theorem 3.1. To show the first part of Theorem 3.1, note that given a strictly positive solution, a peerwise proportional fairness allocation guarantees that for any nodes i, j and k,

$$\frac{x_{ij}}{x_{ji}} = \frac{x_{ik}}{x_{ki}} \text{ at node } i, \ \frac{x_{ki}}{x_{ik}} = \frac{x_{kj}}{x_{jk}} \text{ at node } k, \text{ and } \frac{x_{jk}}{x_{kj}} = \frac{x_{ji}}{x_{ij}} \text{ at node } j, \text{ i.e., } \frac{x_{ij}}{x_{ji}} = \frac{x_{ik}}{x_{ki}} = \frac{x_{jk}}{x_{kj}} = \frac{x_{ji}}{x_{ij}}.$$

It must then be true that $x_{ij} = x_{ji}, \ \forall i \neq j$. Conversely, given a symmetric strictly positive allocation $\mathbf{x}$, we have $x_{ij} = x_{ji}$ and $b_i = \sum_{j \neq i} x_{ij} = \sum_{j \neq i} x_{ji}$. With these relations, it is straightforward to verify that a symmetric strictly positive allocation satisfies the peerwise proportional fairness criterion in Equations 2.

To show a sufficient condition for existence of strictly positive peerwise proportional fairness allocation, we will construct a peerwise proportionally fair allocation in the form of matrix $\mathbf{x} = [x_{ij}]$ step by step starting from the peer with the lowest upload bandwidth b_1. We start by constructing the first column and first row of $\mathbf{x}$, such that they are symmetric and positive, i.e., $x_{1j} = x_{j1} > 0, \ \forall j = 2, \ldots, n$. Our goal is to ensure that after this allocation, the remaining nodes $2, \ldots, n$ with residual bandwidth $b_j^{(1)} = b_j - x_{j1}, \ j = 2, \ldots, n$ are still ordered, i.e., $b_2^{(1)} \leq b_3^{(1)} \ldots \leq b_n^{(1)}$, and satisfy the non-dominant condition, i.e., $b_n^{(1)} < \sum_{j=2}^{n-1} b_j^{(1)}$. The allocation problem over the remaining $n-1$ peers can then be considered. Two cases need to be considered in our first step. First, if $b_n > b_{n-1} + b_1$, we will make the following allocation: $x_{11} = 0$, $x_{1k} = x_{k1} = \delta, \ k = 2, \ldots, n-1$, and $x_{1n} = x_{n1} = b_1 - (n-2)\delta$, where $0 < \delta < \frac{\sum_{i=1}^{n-1} b_i - b_n}{(n-1)(n-2)}$ is small enough such that at the next step the non-dominant condition still holds. The second case is when $b_n < b_{n-1} + b_1$, then we make the following allocation: $x_{11} = 0$, $x_{1k} = x_{k1} = \frac{b_1}{n-1} - \frac{(b_n - b_{n-1})}{n-1}, \ k = 2, \ldots, n-1$, and $x_{1n} = x_{n1} = \frac{b_1}{n-1} + \frac{(n-2)(b_n - b_{n-1})}{n-1}$. We can continue this reduction iteratively until the problem has dimension three, where the feasibility of each step is guaranteed by ensuring δ is selected at each step to ensure the non-dominant condition holds. For the case of dimension three, the non-dominant condition still holds, i.e., $b_n^{(n-3)} < b_{n-2}^{(n-3)} + b_{n-1}^{(n-3)}$, and we can use Fact 1 to make a final allocation. Therefore we have shown that we can construct a strictly positive symmetric allocation and it satisfies the peerwise proportional fairness.

One can construct a set of solutions based on the above approach, e.g., arbitrarily making symmetric allocations as long as the non-dominant condition is satisfied at every step. This solution space is indeed a convex set. Given allocations $\mathbf{x}$ and $\mathbf{x}'$ are both feasible strictly positive peerwise proportionally fair allocations, we want to show that $\mathbf{x}'' = \alpha\mathbf{x} + (1-\alpha)\mathbf{x}', \ \forall 0 < \alpha < 1$ is still a feasible strictly positive peer wise proportionally fair allocation. Note that $x_{ij} = x_{ji} > 0$, $x'_{ij} = x'_{ji} > 0$ and $x''_{ij} = \alpha x_{ij} + (1-\alpha)x'_{ij}, \ \forall i \neq j$. Therefore $x''_{ij} = x''_{ji} > 0$, i.e., the allocation $\mathbf{x}''$ is also symmetric and all positive for $i \neq j$, which satisfies the definition of peerwise proportional fairness, i.e., $\mathbf{x}''$ is also a feasible strictly positive peer wise proportionally fair allocation.

To show that this is a necessary condition, simply consider a counter example of three nodes with bandwidth b_1, b_2 and b_3 and $b_3 > b_1 + b_2$. There is no way to construct a strict

positive allocation that is symmetric. Otherwise $b_3 = d_3 \leq b_1 + b_2$ and it is a contradiction to the condition $b_3 > b_1 + b_2$. Similar argument can be easily generalized to n-node case. Therefore the necessary condition is proved. □

Proof of Corollary 3.1. For each fully connected subgraph $\mathcal{G}_k(N_k)$ with the set of vertices N_k, we have $x_{ij}^{(k)} = x_{ji}^{(k)} > 0$ if $i, j \in N_k$ by Theorem 3.1 since the non-dominant condition and full connectivity hold locally in $\mathcal{G}_k(N_k)$. If $i \notin N_k$ or $j \notin N_k$, we still have $x_{ij}^{(k)} = x_{ji}^{(k)} = 0$. Since $\mathcal{G}(N,E)$ is covered by $\bigcup_k \mathcal{G}_k(N_k)$, $\sum_k x_{ij}^{(k)} = x_{ij},\ \forall i \neq j$. Therefore, it is straightforward to see $x_{ij} = x_{ji},\ \forall i, j \in \mathcal{G}$. □

References

[1] S. R. et. al., Maintenance-free global data storage, in: IEEE Internet Computing, pages 40–49, September-October 2001.

[2] S. Graupner, W. Kalfa, C. Reimann, Modeling and simulation of media-on-demand services - evaluating a digital media grid architecture, Tech. Rep. HPL-2002-192, HP Laboratories (2002).

[3] B. Cohen, Incentives build robustness in BitTorrent, in: Proceedings of Workshop on Economics of Peer-to-Peer Systems, 2003.

[4] M. Ripeanu, I. Foster, A. Iamnitchi, Mapping the Gnutella network: properties of large-scale peer-to-peer systems and implications for system design, IEEE Internet Computing 6 (1) (Jan.-Feb.2002) 50–57.

[5] M. Ripeanu, Peer-to-peer architecture case study: Gnutella network, in: Proceedings of First International Conference on Peer-to-Peer Computing, pages 99–100, 2001.

[6] S. Saroiu, P. Gummadi, S. Gribble, A measurement study of peer-to-peer file sharing systems, in: Proceedings of Multimedia Computing and Networking, San Jose, January 2002.

[7] T. E. N. et. al., Measurement-based optimization techniques for bandwidth-demanding peer-to-peer systems, in: proceedings of IEEE INFOCOM03, San Francisco, April 2003.

[8] J. Ritter, Why Gnutella can't scale. No, really, in: URL http://www.tch.org/gnutella.html, 2001.

[9] Z. G. et. al., Modeling peer-peer file sharing systems, in: Proceedings of IEEE INFOCOM, San Francisco, April 2003.

[10] K. L. et. al., Incentives for cooperation in peer-to-peer networks, in: Proceedings of Workshop on Economics of Peer-to-Peer Systems, 2003.

[11] K. Anagnostakis, M. Greenwald, Exchange-based incentive mechanisms for peer-to-peer file sharing, in: Proceedings of 24th IEEE International Conference of Distributed Computing (ICDCS), March 2004.

[12] A. Habib, J. Chuang, Service differentiated peer selection: an incentive mechanism for peer-to-peer media streaming, Tech. Rep. TR-2004-2-HC, UC Berkeley (February 2004).

[13] R. Ma, S. Lee, D. Yau, A game theoretic approach to provide incentive and service differentiation in P2P networks, in: Proceedings of ACE SIGMETRICS/Performance Evaluation Review, Vol. 32, 2004.

[14] J. Mundinger, R. Weber, Efficient file dissemination using peer-to-peer technology, Private communication .

[15] K.B.Athreya, P.E.Ney, Branching processes, Springer-Verlag New York Heidelberg Berlin, 1972.

[16] X. Yang, G. de Veciana, Service capacity of peer-to-peer networks, in: Proceedings of IEEE INFOCOM, March 2004.

[17] `http://bitconjurer.org/BitTorrent/`.

[18] A. M. F.P. Kelly, D. Tan, Rate control in communication networks: shadow prices, proportional fairness and stability, Journal of the Operational Research Society 49 (1998) 237–252.

[19] D. Bertsekas, J. Tsitsiklis, Parallel and distributed computation: numerical methods, Athena Scientific, 1997.