Service Capacity of Peer to Peer Networks

Xiangying Yang and Gustavo de Veciana
Department of Electrical and Computer Engineering
The University of Texas at Austin
Austin, TX 78712
{yangxy,gustavo}@ece.utexas.edu

***Abstract*— In this paper we study the 'service capacity' of peer to peer (P2P) file sharing applications. We begin by considering a transient regime which is key to capturing the ability of such systems to handle bursty traffic, e.g., flash crowds. In this context our models, based on age dependent branching processes, exhibit exponential growth in service capacity, and permit the study of sensitivity of this growth to system policies and parameters. Then we consider a model for such systems in steady state and show how the average delay seen by peers would scale in the offered load and rate at which peers exit the system. We find that the average delays scale well in the offered load. In particular the delays are upper bounded by some constant given any offered load and even decrease in the offered load if peers exit the system slowly. We validate many of our findings by analyzing traces obtained from a second generation P2P application called BitTorrent.**

***Index Terms*— system design, network measurements, peer to peer applications, flash crowds, service capacity, performance evaluation, mathematical modeling**

I. Introduction

Peer-to-peer (P2P) architectures for file sharing 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) 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 – thus hosts are considered peers.

P2P file sharing applications first became prominent with the introduction of Napster, which allowed users to share MP3 formatted music files. In February 2001, Napster boasted a peak of 1.5 million simultaneous users[1], but subsequently suffered legal setbacks. However new P2P file sharing applications such as Gnutella, KaZaA, Morpheus, eDonkey and BitTorrent continue emerging and the number of users is growing faster than ever[2]. Indeed, in March 2002 2.9 million simultaneous online users were reported [3], and 6.2 million users in January 2003, see *http://www.slyck.com*. According to the SD–NAP trace [3], the dominant traffic type observed by Internet service providers (ISPs) is associated with P2P file sharing applications. Perhaps driven by the growth in broadband services, e.g., cable and ADSL, the average document size exchanged on P2P networks is growing, e.g., enabling the sharing of video files. Thus it is reasonable to expect the predominance of P2P traffic on the Internet to grow further. In addition to file sharing, P2P overlays have also been proposed as part of solutions to handle Internet 'flash crowds,' i.e., unexpected rapid increases in the demand for particular objects, which challenge content delivery network infrastructure such as Akamai[4]. Indeed many researchers, including [5], [6], [7] have proposed the use of P2P overlay networks on top of online clients as supplementary means for providing web content in order to alleviate the traffic burden on content servers and smooth/balance traffic on networks when flash crowds occur. In fact researchers are developing a broader framework called "grid computing" which enables distributed content delivery, storage, computation and file sharing over overlay networks, for example UC Berkeley's OceanStore project[8] and HP Lab's Digital Media Grid project[9].

This work is supported in part National Science Foundation Grant ECS-0225448.

A. What is the service capacity of a P2P system?

In addition to enabling the sharing of CPU, storage and bandwidth resources, P2P architectures excel in serving bursty requests. For example, if a host has a popular file and many peers are requesting it they are likely to see poor performance. However, as soon as one of the downloading hosts finishes, the system has two 'servers', with which it can serve other peers. As this process evolves it is easy to see that the number of servers in the system grows exponentially and the throughput seen by peers improves. When the number of servers becomes large enough to serve the intensity of requests, the system enters a 'steady state' where the throughput performance of each peer is stable. These two phases are exhibited in the representative trace shown in Fig.1. It begins with the addition of a new document to a P2P system. The solid line tracks an exponential growth in service capacity corresponding to a transient period, and the dotted line corresponds to fluctuations in a steady state. Note that during the 'steady state' the request rate need not be stationary. Indeed, not shown in Fig.1, the offered load may fluctuate yet the average performance per peer is fairly stable. As will discussed in the sequel, during the steady state period the service capacity tends to scale with the offered load.

This example exhibits a desirable exponential growth, and subsequent self-scaling (based on popularity) of a P2P system's *service capacity* for a given document. Ignoring heterogeneity in the upload bandwidth and computing capacity of peers, in both

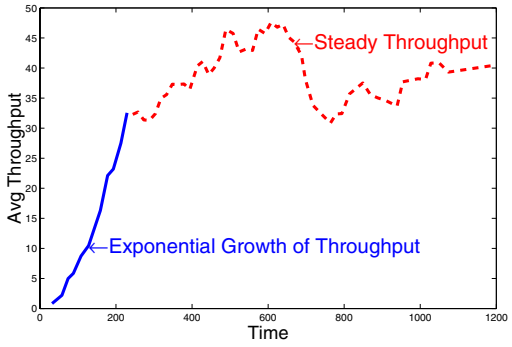


Fig. 1. Two-phases in the evolution of the average throughput per peer versus time for a single document introduced into a P2P network.

cases the number of peers willing to serve the document is the driver. Yet service capacity should be viewed in two regimes: the 'transient' and 'steady state'. In the transient regime one might wish to assess how quickly a P2P network starting with a limited number of servers can grow its service capacity to meet the load associated with a burst of demands. While in the steady state regime, one might wish to assess the average throughput/delay performance seen by a typical peer. Note that in the transient regime, the system is server constrained, i.e., requests are backlogged and hence the service capacity increases exponentially fast with a rate that reflects the system's intrinsic ability to leverage its resources to share limited service capacity. By contrast, in steady state the service capacity depends and/or adapts to fluctuations in demand; in this regime the service capacity is demand constrained.

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;
- 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 and scheduling policy: limiting the number of concurrent downloaders and/or scheduling to provide differentiation/priority among them;
- traffic: 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 they have already obtained prior to completing downloads. Allowing large numbers of peers to download from one another may increase the subsequent potential service capacity for a document but may increase delays. Spatial clustering of peers may impact the service capacity of a P2P system in subtle ways, since serving a peer which is far away and may have low bandwidth, may subsequently help to quickly serve an entire neighborhood of interested peers. Recognizing some of these relationships new P2P applications attempt to use simple *credit based systems* so as to provide incentives for peers to stay online and 'open' their upload bandwidth to serve other peers. This is often done by keeping peers' credit history and based on their behavior give them different priority in transfers or access. Such mechanisms are geared at modifying user behavior and thus the offered load. As we will see in the sequel their impact on performance may be subtle. These complex relationships motivate the need for a systematic study of these factors and tradeoffs on a P2P system's transient as well as its stationary service capacity which is the starting point for our work.

B. Related Work

Most research on P2P systems so far has emphasized design, traffic measurement and workload analysis but not performance evaluation. Early work by [10][11][12] 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 P2P structure and dynamics, e.g., connectivity and peer/host behaviors. More recent research in the direction of evaluating P2P systems has focused on performance. Peer selection schemes were evaluated in [13], where measurements are used to optimize the selection of good peers and improve the overall system performance. A few researchers have used analytical models to study the performance of P2P networks. For example, [14] constructed a model for signaling messages in the Gnutella network and concluded that signaling might significantly compromise performance. The work in [15] is among the first to model 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.

C. Our Contributions

In this paper we study the performance characteristics of 2nd generation P2P applications, e.g., BitTorrent (BT), which implement various performance optimizations, via both trace measurement and analysis. We believe that policies aimed at improving system performance are crucial to building viable P2P applications. Our measurement work is also unique in that we consider performance as a function of time at the granularity of a single P2P transfer session. These measurements highlight the need for performance analysis focusing on both user experience in steady state and system performance in the transient regime.

With this in mind we study two measures for the *service capacity* of a P2P system. We model the transient service capacity of a P2P network by a branching process. Based on this model, we consider how to optimize service policies in order to maximize the service capacity growth rate. We are not aware of any previous work that has analyzed the transient capacity of a P2P

system. In addition to our transient model, we propose a simple steady state model, which captures the impact of peer departures or document deletions on the stationary service capacity of such systems. Our analytical results and measurements suggest how various mechanisms might be designed to make a P2P system suitable for handling very bursty and large volume demands and/or provide users good performance when the P2P network membership is dynamic and possibly heterogeneous.

D. Organization of this paper

The rest of the paper is organized as follows. We propose a transient model for P2P systems in Section II and consider how various policies would impact the speed at which the service capacity grows, i.e., catches up with demands. In Section III we propose a model and study the performance of a P2P system in steady state. We perform a detailed trace analysis of the BT P2P application in Section IV which supports in part the validity of our models for P2P networks in a more complex environment. Finally we conclude our work in Section V.

II. Transient Analysis of Service Capacity

The purpose of our transient analysis 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 introduced by a single peer, and may be subject to large bursts of requests far exceeding the available service capacity. Our goal is thus to ensure a document is disseminated as quickly as possible to interested peers until the system reaches a steady state where the service capacity is commensurate with demands.

A. Deterministic model

Let us first consider a simple model for file sharing in the transient regime. Suppose that $n-1$ users wish to acquire a document which is initially available at one peer. To make derivations simple let $n = 2^k$. Assume that each peer has a limited upload capacity but the network capacity is otherwise unconstrained. More specifically, each peer has an upload bandwidth

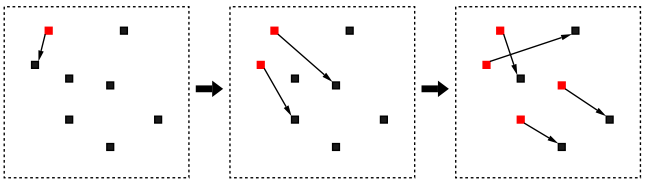


Fig. 2. File sharing in a P2P system.

of b bps, and can serve a document only once it has been fully downloaded. Suppose the document has size s bits. Thus to serve $n-1$ requests $(n-1)s$ 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 $n-1$ 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 *"service capacity"*, i.e., the number of peers available to serve the document. This deterministic model exhibits the great potential of a P2P framework to support large bursts of requests in a decentralized manner.

Under the proposed strategy the $n-1$ peers will be served by time $\tau\log_2(n+1) = \tau k$. During this transient regime the average delay $\bar{d}$ experienced by peers can be computed as follows. Let d_j denote the delay experienced by the jth peer and note that $2^{i-k}n$ peers complete service at time $(i+1)\tau$. Assume the peer who initially has the file experiences zero delay. Thus,

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

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 a linear scaling expected for a server-client model with a fixed set of servers.

Next let us consider multi-part downloads. Suppose the file is divided into m chunks with 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 should be 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 on 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 Fig.3(a), at time k slots, 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 Fig.3(b), 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 $\bar{d}^{(m)}$ 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 peers will receive chunk m during the last time slot $k+m$, the average delay experienced by peers can be computed as follows. Let $d_j^{(m)}$

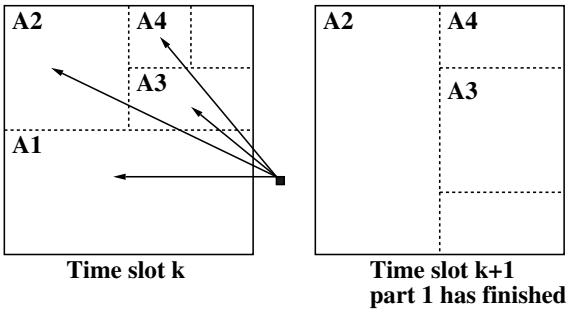


Fig. 3. Concurrent multi-part downloads among a set of n peers.

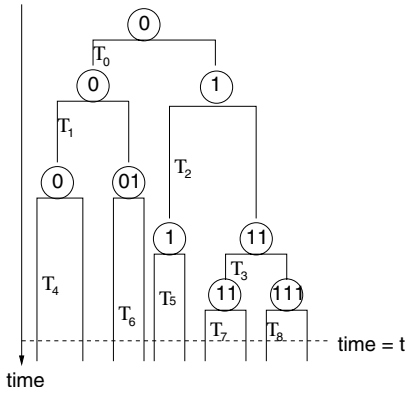


Fig. 4. Branching process model for file replication across as P2P system.

denote the delay for the jth peer in the multi-part download, then

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

Thus a large m, i.e., small chunk size, leads to a factor of m improvement in average delay for this transient regime. In practice, however, we 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. However our models are somewhat 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 bottle-necked, 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 as well as some other aspects of real P2P systems, e.g., speed at which peers leave the system.

B. Branching process model

In this section, 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 or peer behavior.

1) Basic branching process model: Let $N_d(t)$ denote the number of peers available to serve document d at time t. Note that the system's service capacity for d should be proportional to $N_d(t)$, see e.g., [15]. The proportionality constant might only depend on the heterogeneity of the upload/server capacity among peers assuming the network is not the bottleneck. We assume that initially there is but one copy of the document d in the network, i.e., $N_d(0) = 1$ with probability 1, and a large number of interested peers. Fig. 4 shows a typical evolution of the file sharing process assuming each peer serves one other peer 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. As shown in the figure 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 $F_T(t) = P(T \le t)$ and $E[T] = \tau = \frac{1}{\mu}$. This distribution captures 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 population, i.e., service capacity of our P2P model, can be found in [16] Section 10.4.22 and [17] Chapter IV Theorem.3A. The following is a restatement of the basic result for a branching process with i.i.d. family sizes with the same distribution as a random variable V.

Theorem II.1: In the super critical case where the mean family size per generation satisfies $E[V] = v > 1$ and F_T is non-lattice, the expected population of an age dependent branching process for large t is approximately given by

$$m(t) \sim \delta e^{\beta t}, \tag{1}$$

where $\beta > 0$ is such that $d\tilde{F}(x) = ve^{-\beta x}dF_T(x)$ is a probability distribution function, i.e., $\int_0^\infty ve^{-\beta x}f_T(x)dx = 1$ whose mean we denote by $\tilde{m}$ and where $\delta = \frac{v-1}{\tilde{m}\beta v}$.
Thus for the P2P model in Fig.4 the mean service capacity follows

$$E[N_d(t)] = \delta e^{\beta t}, \tag{2}$$

with β and δ as defined in Theorem.II.1 and where $v = 2$.

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 in the logarithm of n. The two parameters β and δ capture the growth characteristics of the service capacity, and depend on the distribution of the transfer times T. For example, if T is exponentially distributed, then $\beta = \mu$ and $\delta = 1$;

by contrast[1], as shown in the previous subsection if T is deterministic $\beta = \mu \ln 2$ and $\delta = 1$, leading to an exponential growth $2^{t/\tau}$.

Consider two branching processes with different generation time distributions, i.e., $T^{(1)}$ and $T^{(2)}$ such that $E[T^{(1)}] = E[T^{(2)}]$ and there is an increasing convex ordering (I.C.X.) on $T^{(1)}$ and $T^{(2)}$, i.e., $T^{(1)} \leq^{icx} T^{(2)}$[18]. In this case one can show that $\beta^{(1)} < \beta^{(2)}$. Indeed since β has to solve the integral equation in Theorem.II.1, in which the left hand side can be interpreted as an expectation of the convex function $e^{-\beta x}$, it follows that I.C.X. ordering ensures that given a β, $E[e^{-\beta T^{(1)}}] \leq E[e^{-\beta T^{(2)}}]$ and hence the relation $\beta^{(1)} < \beta^{(2)}$ has to be true to ensure both of them to solve the integral equation. Also note that with $T^{(1)} \leq^{icx} T^{(2)}$ and $E[T^{(1)}] = E[T^{(2)}]$, it then follows that $Var(T^{(1)}) \leq Var(T^{(2)})$. Thus we see that for a fixed mean, variability in generation times improves the growth exponent β, e.g., in the exponential case $\beta = \mu$ and in deterministic case $\beta = \mu \ln 2 < \mu$.

2) Modeling parallel uploads when $v > 2$: 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 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 $v-1$ times. 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 become available. Thus one might ask whether parallel uploading with $v > 2$ would lead to faster growth rates.

With the proposed re-scaling of the transfer times density, i.e., $\frac{1}{v-1} f_T(\frac{t}{v-1})$, according to Theorem.II.1 the growth rate β must satisfy

$$\int_0^\infty v e^{-\beta x} dF_T(\frac{x}{v-1}) \quad = \quad 1.$$

For the case where download times are exponentially distributed, when $v > 2$, one can show that $\beta = \mu$ and $\delta = 1$. Thus, in this case the expected service capacity $E[N_d(t)] = e^{\mu t} = e^{\frac{t}{\tau}}$, does not depend on v. For the case of deterministic download times, one need only modify the model in Section II-A by reducing the regeneration time to $\frac{\tau}{v-1}$ and increasing the number at each regeneration to v. This gives the exponential growth of $v^{\frac{t}{(v-1)\tau}}$, i.e., $\beta = \frac{\ln v}{v-1}\mu$ and $\delta = 1$. This indicates that the growth exponent β, decreases with v. The deterministic transfer time for a particular file is perhaps closer to practice to P2P systems, which see upload bandwidth constraints and limit the number of concurrent peers. This result suggests that the growth rate β might decrease, albeit moderately, if v is large. Moreover, considering the overheads associated with each transfer and non-linearities in performance degradation when $v > 2$, the actual performance with parallel uploading could be even worse. Thus it may make sense to limit the number of peers that any server will serve concurrently.

3) Uncooperative peers under a parallel uploading scenario: We have concluded that the growth rate in service capacity for a P2P network in the transient regime might be highest when a peer serves a limited number of other ones at a time. However, in practice peers that have completed a transfer may leave the system or delete the file. In this section we will show that when peers exhibit such uncooperative behavior parallel uploading may help achieve higher growth rates.

Consider once again our branching process model, where each peer serves $v-1$ others at a time. Upon completing their downloads, each peer independently determines whether it stays in the system and with probability $1-\zeta$ exits the system. Thus the family size is in fact a random variable with mean $v\zeta$. Under these dynamics our branching process may become 'extinct', i.e., eventually no peers are available to serve the document. In fact, standard results, see [17] Chapter IV, show the process will become extinct with probability 1 if $v\zeta < 1$. Hence if our goal is to maximize the growth rate and avoid extinction it is desirable to select a family size satisfying $v\zeta > 1$.

Assuming $v\zeta > 1$, let us consider the growth rates that would be achieved. Again based on Theorem.II.1, but incorporating uncooperative peers, and the transfer time re-scaling associated with $v-1$ parallel uploads the new growth rate β' and constant δ' satisfy

$$\int_0^\infty v\zeta F(\frac{x}{v-1}) \beta' e^{-\beta' x} dx = 1 \quad \text{and} \quad \delta' = \frac{1}{\tilde{m}'\beta'}[1 - \frac{1}{v\zeta}].$$

In this case it is no longer the case that the maximal growth rate is achieved when v is as small as possible. For example, as shown earlier assuming exponential peer to peer transfer times, $\beta = \mu$ and $\delta = 1$, i.e., the growth rates do not depend on v. Now considering the peer departure dynamics modeled by ζ, the new parameters for the growth process are given by

$$\beta' = (\zeta - \frac{1-\zeta}{v-1})\mu \text{ and } \delta' = 1 + \frac{1-\zeta}{v\zeta - 1}.$$

Note that the growth exponent β' increases, albeit slowly, in v. Fig. 5 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' still increases in v but not significantly. Thus when some fraction of peers exit the system upon completion of their download, allowing parallel uploads may help assure document availability and improve the overall exponential growth of the system's service capacity.

4) Role of multi-part downloads on transient capacity: Earlier, based on our idealized deterministic model, we showed that multi-part downloads will help further speed up growth and thus delays by a linear factor m. A similar benefit also exists for our branching process model.

Suppose a file is partitioned into m identical sized chunks, and assume that the number of requesting peers is large. As a simple model for service capacity growth under multi-part downloading

[1]Note that deterministic time does not satisfy the conditions of Theorem.II.1, but growth rates are easily computed

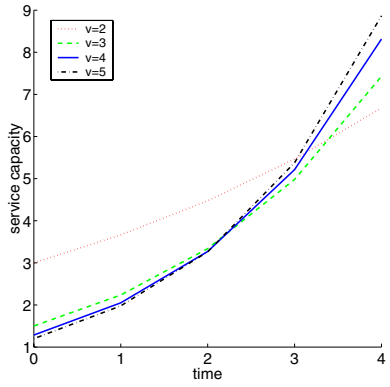


Fig. 5. Mean growth in service capacity in a system with uncooperative peers and parallel uploading: various choices of v are shown for $\zeta = 0.6$ and $\mu = 1$ fixed.

suppose that after a finite time each chunk has a distinct source peer and subsequently the m chunks are duplicated over m independent branching trees. We can modify our original branching process model to account for chunk size by re-scaling the distribution for downloads by a factor of $\frac{1}{m}$. Thus the growth in service capacity for a given chunk $N_d^{(m)}(t)$ can be related to our original model $N_d(t)$, as follows

$$E[N_d^{(m)}(t)] = E[N_d(mt)] = \delta e^{m\beta t},$$

i.e., growth rate increases from β to βm. Given a burst of demands q, the time to complete q jobs is roughly $\frac{1}{\beta m}\ln(\frac{qm}{\delta})$. Thus, we again observe that using a multi-part download scheme reduces delay roughly by a factor of $\frac{1}{m}$. Note that the above multi-part model is quite optimistic since it assumes peers will not be serving multiple chunks at the same time. Such concurrency would slow down file sharing.

5) Discussions: Optimizing P2P systems to deal with flash-crowds: When a P2P system is subject to a flash crowd it is desirable that its service capacity grows as quickly as possible, i.e., β be high. Thus our models suggest that P2P systems enabling multi-part downloads will achieve significantly better performance. As will be observed in the trace analysis in Section IV, steady state performance also improves although sub-linearly in the number of chunks m, perhaps due to signaling/transfer overheads associated with realizing multi-part schemes.

The benefit of allowing parallel uploads is not clear unless peers tend to be uncooperative. Our models suggest that in a structured application like a media grid, in which peers are always available, parallel uploads may not improve the overall performance. However in file sharing applications where peers can freely leave the system, allowing parallel uploads may enable higher accessibility and better growth rates in the service capacity during transients.

User behavior is hard to control yet may be influenced by implementing service policies. For example many P2P applications use a credit system to reward peers that contribute well to the overall system. Such a policy might increase a peer's download volume/rate based on the peer's measured upload volume/rate. This may have two positive impacts on system performance. First, it may give better delivery rates to peers that are likely to continue serving others and thus increase the overall transient growth rates in service capacity. Second, it may encourage peers to increase their upload bandwidth and participate more aggressively in sharing. This obviously encourages peers to be more friendly and cooperative.

Interestingly, in the transient regime bursty demands help the system handle the traffic better, since, particularly under a multi-part P2P design, the system will be able to leverage the service capacity of the peers while they are downloading the document leading to fast growth in the service capacity. In this sense, a significant amount of a P2P system's service capacity, in both the transient but also the stationary regime may be derived from leveraging the service capacity of peers which are concurrently downloading the same document, before they leave the system.

III. Steady State Analysis Of Average Delays

Next we consider a steady state analysis for P2P service capacity based on a Markov chain model. 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.

A. Markov chain model

We shall consider all peers in a P2P system which are interested in, or serving, a particular document and assume that there will always be at least one peer serving the document. Suppose new requests follow a Poisson process with rate λ. The system's state is pair $(x,y) \in \mathsf{N} \times \mathsf{N}^+$, where x denotes the number of peer requests currently in progress or queued and y denotes the number of peers that have finished downloading and still 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 total service capacity in the system is thus proportional to the effective number of servers in the system, we denote it by $\mu(\eta x + y)$, where μ denotes the service rate for a request at a peer which can serve the document in full. Each time a peer completes downloading the document it becomes a server in the system, but each such peer may leave the system at rate γ. Thus, in this model the service time for a request at a single peer and the time until a peer having completed a download leaves the system are independent 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 $\mathsf{N} \times \mathsf{N}^+$ given by :

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

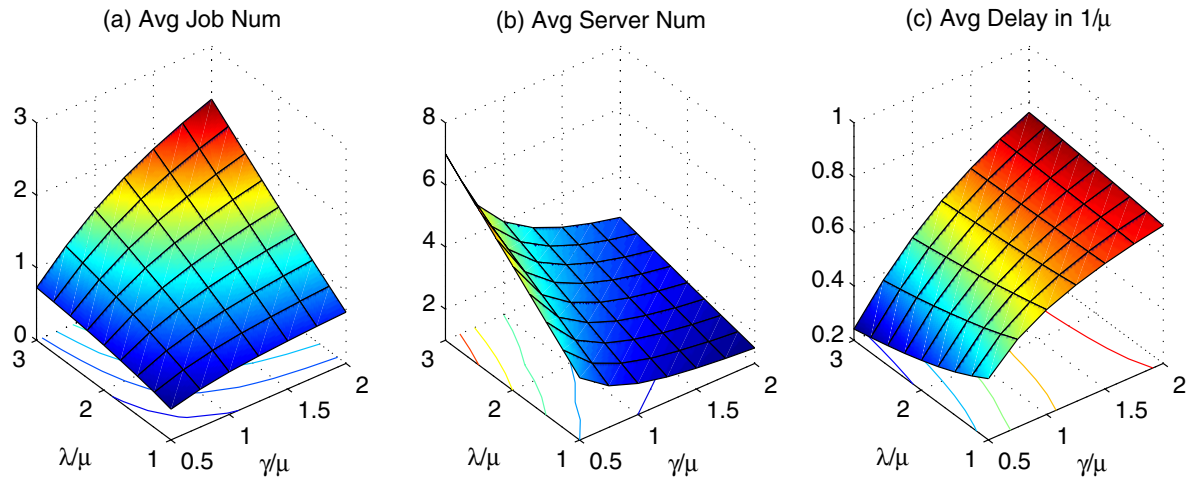


Fig. 6. Performance in steady state, with $\eta = 0.5$

We numerically computed the stationary distribution for this Markov chain by truncating the state space appropriately, and calculated the mean number of jobs, servers, and delay for this system. We do this for a range of parameters; specifically we let $\mu = 4.0$, $\eta = 0.5$ and varied the values of λ and γ from 4.0 to 12.0 and 2.0 to 8.0, respectively. Our performance metrics actually depend only on the ratios $\frac{\lambda}{\mu}$ the offered load, and $\frac{\gamma}{\mu}$ the rate at which peers exit the system, as long as delays are measured in the units of holding times μ^{-1}. Thus we rescaled our results to provide better insight. As shown in Fig.6 the mean number of queued jobs increases sub-linearly in the offered load; the mean number of servers in the system is roughly linear in the offered load; and the mean delay decreases in the offered load as long as the exit rate is less than 1.75 and increases otherwise. This last observation is perhaps the most interesting. It suggest that in a P2P system where nodes exit slowly the average delay seen by peers might improve in the offered load. The intuition here, is that service capacity (number of servers plus a fraction of downloaders) is increasing linearly in the offered load. Note even if peers exit the system at a higher rate, e.g., for our example $\frac{\gamma}{\mu} > 1.75$, the average performance seen by each peer degrades slowly to a constant as the offered load increases.

Note that in the limiting regime where $\gamma \to \infty$, i.e., nodes immediately exit the system, one can show the average delay per peer increases slowly with the offered load converging to roughly $\frac{1}{\eta\mu}$, as might be expected from a $M/G/\infty$ queue, with mean service times $\frac{1}{\eta\mu}$. In summary, as was the case in transient regime considered in Section II by providing incentives for peers stay and share documents, i.e, decrease the parameter γ/μ, the average delay can be reduced, and in some scenarios even made to scale favorably in the offered load.

B. Estimating effective throughputs realized by peers

In the above model we optimistically assumed that the full service capacity of the y peers in the system who have completed downloading a file is available to the x peers for which downloads are in progress – we refer to these as *seeds* and *downloaders* respectively. Further we modeled the capacity that can be leveraged from one of the x downloaders as a fraction η of that made available by a seed. Thus assuming the state of the system is (x,y) then the total aggregate throughput exchanged in the system would be

$$\text{agg. throughput} = (\eta\mu s)\cdot x + (\mu s)\cdot y \approx u_{dl}\cdot x + u_{seed}\cdot y \quad (3)$$

where s denotes the size of the file under consideration and μ is the mean service time to upload the entire file from a seed. In a practical system due to the mechanisms and heterogeneity involved for searching and exchanging documents, or parts thereof, each downloader and seed can be thought of as realizing an *effective* upload throughput which we will denote by u_{dl} and u_{seed} respectively. Thus the parameter η can roughly be thought of as the ratio between the effective throughput of a downloader versus that of a seed. Ideally a system with a high η is one which is effective at leveraging the capacity of concurrent downloaders, and thus likely to be excellent at tracking bursty offered loads.

In the sequel we will make use of this simple linear model to attempt to estimate the effective throughputs u_{dl} and u_{seed} in downloading activities associated with P2P systems sharing documents of different sizes. We will assume that if the offered load high enough these parameters are fairly insensitive to offered load. Indeed, as shown in Fig.6 the marginal change of system performance is fairly small when the offered load is high. However, u_{dl} and u_{seed} are likely to depend on the file size s. Indeed we would expect that the ability of a seed and downloader to contribute will depend on various aspects. For example if a file is large, downloading will take longer, and thus information about partial availability of parts associated with a larger file will have time to propagate through the system and be leveraged by other downloaders. Thus one might expect the overheads to disperse and process information about the availability of parts of a file may be high for small files, and preclude an effective utilization of all available resources. In SectionIV, we use trace measurement to study the effective upload contributed by peers in detail to find that our empirical results support these observations.

IV. Trace Measurements and Traffic Characterization

In this section we will analyze several traces obtained on the BitTorrent(BT) P2P application. Our purpose is to study system performance and validate, in part, some of the results and observations in Sections II and III.

A. BitTorrent P2P application and measurement setup

We sampled system data on BT P2P over a period of several days. Information on this open-source project can be found at *http://bitconjurer.org/BitTorrent/*. 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 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 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.I. 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; 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. The system simultaneously tracks about 150–200 files that have been recently inserted. Thus a trace permits one to evaluate how the system capacity for an individual file evolves over time.

B. Methodology

1) Service capacity: Since BT uses multi-part downloads service capacity must be carefully defined. We estimate the service capacity as

$$\text{effective \# of servers} = \frac{\text{“total storage space shared”}}{\text{size}},$$

i.e., the *effective number* of replicas available in the system, including partial downloads. Since peers may exit or delete a file upon completing a download, we can estimate service capacity based on the following formula

$$\frac{\text{TX vol} - (\text{\# finished} - \text{\# seeds} + 1) \times \text{size}}{\text{size}}.$$

2) Throughput and delay of each peer: We estimate the average instantaneous throughput seen by each peer as follows

$$\text{avg throughput per peer} = \frac{\text{throughput}}{\text{\# downloaders}},$$

and we define the KByte transmission delay for each peer as

$$\frac{1}{\text{avg throughput per peer}} = \frac{\text{\# downloaders}}{\text{throughput}}.$$

This is roughly the current average (across peers) delay to transfer 1KByte of data, which is but a rough estimate for average transfer delays seen by peers.

3) Estimating effective throughputs for seeds and downloaders: As discussed in Section III-B we let u_{dl} and u_{seed} denote the effective upload throughputs realized by typical downloaders and seeds respectively. Based on the measurements we were able to directly collect estimates for the peer upload throughputs. To circumvent problem we sampled the file sharing dynamics associated with particular files to obtained multiple triples comprising the # seeds, # downloaders, and (aggregate) throughput exchanged by peers over time. We then did a least squares fit to this data to the linear model presented earlier, see Eq.(3). This essentially corresponds to performing a two-dimensional regression to obtain the sensitivity of throughput to number of seeds and downloaders correspond to u_{dl} and u_{seed}.

This study was conducted for sets of peers sharing popular files, i.e., typically involving more than 100 concurrent downloaders and associated with file sizes ranged from 200MB to 2GB. The objective was to study the relative values of u_{dl} and u_{seed} and their sensitivities to file size.

C. Is there an exponential growth in the transient service capacity and average throughput per peer?

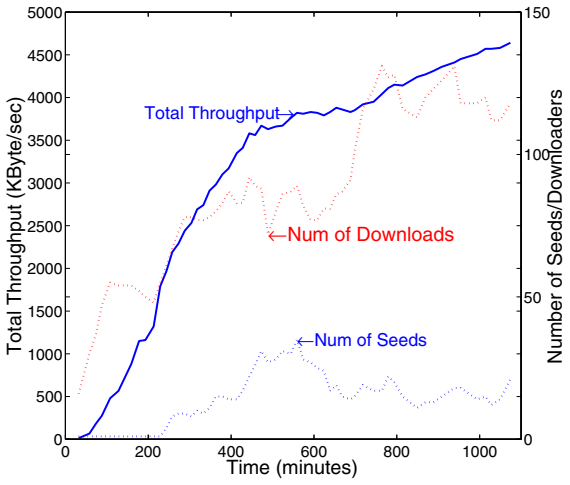


Fig. 7. Total throughput, number of seeds, and downloaders associated with a file in the BT system.

Fig.7, shows the total throughput, 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 how fast increases in service capacity in the initial transient mode are enabled by multi-part downloading, i.e., downloading

insert time	file name	size	#seeds	#downloaders	#finished	TX vol	throughput	life
6.24.2003.10:45	F1 challenge	678MB	2	8	104	75.05GB	265.31KB/s	3 10:43

TABLE I

FORMAT OF BT TRACE FILE

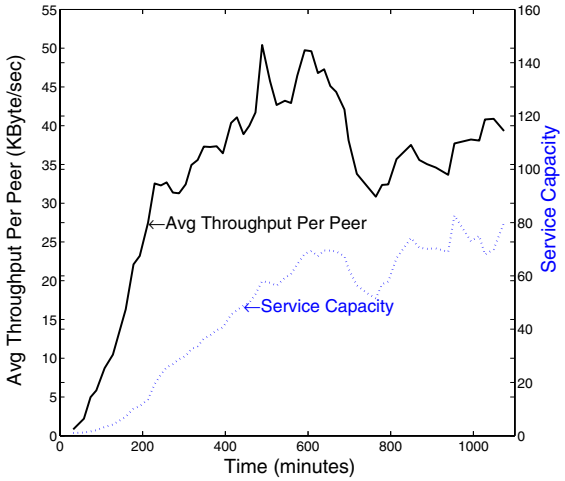


Fig. 8. Average throughput per peer and service capacity for a file in BT system.

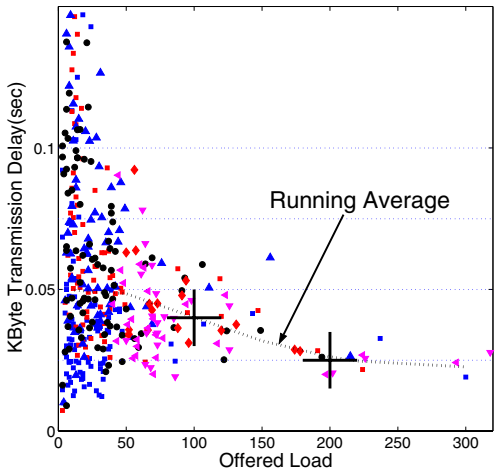


Fig. 9. The KByte transmission delay versus offered load(estimated by summing over *download* + *seed*) of files in BT system.

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 mentioned in Section I. 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 due to the signaling overheads among larger number of peers scaling poorly.

Fig.8 shows the average throughput per peer and service capacity of the system as a function of time. The exponential growth of throughput per peer is remarkable during the first 300 minutes. Thereafter the individual peer's throughputs track the service capacity fairly well. The drop at time 600 minutes seems to be associated with a sharp drop in the number of seeds with a concurrent large increase in the number of downloaders.

D. How does the offered load impact average throughput performance per peer?

The data shown in Fig.9 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 throughput per peer should be representative of their popularity/offered loads. For each file, we plot the KByte transmission delay, i.e., inverse of the average throughput per peer (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 in the offered load. For example, as marked in Fig.9, when number of peers is 200, the average throughput per peer is roughly 40KBytes/sec, i.e., the delay to transmit 1Kbyte is 0.025sec and when peer number is 100 the average throughput per peer 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, possibly due to TCP flow control. Comparing this performance characteristics with those for the model shown in

Fig.6(c), we noted that the average delay performance gets better in the offered load but the marginal gains decrease.

E. Impact of file size on the per seed and downloader's effective throughput?

Let us first consider the dependence of the effective upload u_{seed} and u_{dl} on the file size. Fig.10(a) shows estimated parameters for files of various sizes. It should be clear that both u_{seed} and u_{dl} improve with the increasing file size, which, as expected, is due to the improved capability of peers to leveraging service capacity via multi-part downloading if the file being exchanged is large. Also, as shown in Fig.10(b), the average per peer download throughput also improves with the increasing file size. Indeed since downloaders can share the service capacity leverage from both downloaders and seeds, it is natural to observe that the average download throughput per peer, which is roughly u_{dl} plus a fraction of u_{seed}, mostly lies above u_{dl}. We note however, that this data is intended to only show rough trends, and the reliability of the collected data would not warrant a conclusive claim on the precise trend, i.e., linear or sub-linear. In particular the 95% confidence intervals on the point estimates shown in Fig.10(a) range from ± 32 KByte/sec to ± 227 KByte/sec. A detailed investigation of our data sets revealed the presence of outliers that we believe are due to resetting of BitTorrent trackers. Furthermore the simple linear model we are using to capture the system's performance may be a crude approximation. Indeed it is possible that offered loads to seeds versus downloaders depend on the relative quantities of these, i.e., u_{seed} and u_{dl} might depend on the ratio of y/x and thus change the model. For this reason the data shown in the figure is associated with files and samples having roughly the same orders of magnitude for x, y.

F. Discussion

To summarize we have observed two different stages of the dynamics of P2P systems. First an initial transient flash crowd phase wherein the overall service capacity is low but quickly catches up with the demands leveraging peers that are downloading a multi-part file to serve others. This is clearly a regime where the system's service capacity is limited yet can quickly benefit from the crowd to bootstrap its ability to serve peers. In a second regime, a steady state, the overall service capacity increases slowly and fluctuates along with demands with an average throughput per peer remaining fairly stable. The steady state performance seen by peers improves in the popularity of the file, i.e., number of participating peers. However we noted an example where subsequent burst of demands does not lead to a dramatic exponential growth in the aggregate throughput, in fact it looks more like a linear response. As explained earlier, we conjecture that this may be due to either a lack scalability, perhaps signaling overheads or be the result of a credit system geared at favoring some peers, versus increasing the rate of growth. We leave this conjecture to further research, but suggest that in steady state, when the number of peers is already large, a credit system may be biased against peers who just enter and penalize the ability of a P2P system to catch up exponentially further upsurges in demands. Finally we observed that multi-part downloading allows significant improvement in the performance as the file size increase.

V. Conclusion

In this paper, we have modeled the service capacity of a P2P system in two regimes. One is the transient phase in which the system tries to catch up bursty demands (flash crowd). Both our analytical model and trace measurements exhibit the exponential growth of service capacity during the transient phase. In the second regime, the steady state, we show that the service capacity of a P2P system will scale with and track the offered loads, so individual user's performance will not degrade significantly. Moreover, both our analysis and empirical data suggest that at higher offered loads and with cooperative users the system performance might improve. These characteristics are particularly desirable in systems designed to handle exceedingly bursty demands.

In addition, we studied various techniques that might help improve P2P performance. Multi-part combined with parallel uploading when properly optimized will generally improve system performance, particularly when peers exit the system at a high rate. A credit system might help provide peers incentives to share and thus improve performance. Yet even a simple credit system based on 'short term' history of peer's upload download volume may limit the system's capability to deal with flash crowds above and beyond an established steady state.

References

[1] "Report: Napster users lose that sharing feeling," in *CNN news, URL:* `http://www.cnn.com/2001/TECH/internet/06/28/napster.usage/`.

[2] "Peer-to-peer file sharing: The effects of file sharing on a service provider's network," in *Industry White Paper, Sandvine Incorporated*, 2002.

[3] "Top applications (bytes) for subinterface: Sd-nap traffic," in *CAIDA workload analysis of SD-NAP data, URL:* `http://www.caida.org/analysis/workload/byapplication/sdnap/index.xml`, 2002.

[4] Jaeyeon Jung, Balachander Krishnamurthy, and Michael Rabinovich, "Flash crowds and denial of service attacks: Characterization and implications for cdns and web sites," in *Proceedings of the International World Wide Web Conference (WWW'02)*, May, 2002.

[5] Dan Rubenstein and Sambit Sahu, "An analysis of a simple p2p protocol for flash crowd document retrieval," in *Technical report EE011109-1, Columbia University*, November 2001.

[6] Angelos Stavrou, Dan Rubenstein, and Sambit Sahu, "A lightweight, robust p2p system to handle flash crowds," in *Proceedings of IEEE ICNP 2002, Paris, France*, November, 2002.

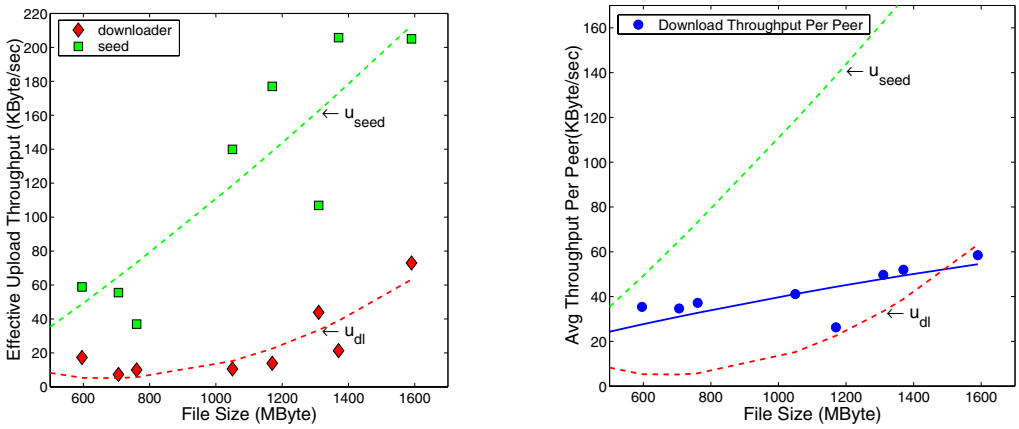
[7] Tyron Stading, Petros Maniatis, and Mary Baker, "Peer-to-peer caching schemes to address flash crowds," in *Proceedings of IPTPS'02, Cambridge, MA*, March 2002.

[8] Sean Rhea, Chris Wells, and Patrick Eaton et al, "Maintenance-free global data storage," in *IEEE Internet Computing, pages 40–49*, September-October 2001.

[9] Sven Graupner, Winfried Kalfa, and Carsten Reimann, "Modeling and simulation of media-on-demand services - evaluating a digital media grid architecture," in *HP Laboratories technical report, HPL-2002-192*, 2002.

[10] Matei Ripeanu, Ian Foster, and Adriana Iamnitchi, "Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design," *IEEE Internet Computing*, vol. 6, no. 1, pp. 50–57, Jan.-Feb.2002.

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



(a) (b)

Fig. 10. On the left, the estimated effective upload u_{seed} and u_{dl} for files of different sizes. The dotted lines are obtained from quadratic fits to the data. On the right, the average per peer download throughput versus the file size. Note as expected it lies roughly between u_{seed} and u_{dl}, where the solid line is a quadratic fit for the average per peer download throughput.

[12] Stefan Saroiu, P. Krishna Gummadi, and Steven D. Gribble, "A measurement study of peer-to-peer file sharing systems," in *Proceedings of Multimedia Computing and Networking, San Jose*, January 2002.

[13] T.S. Eugene Ng, Yang hua Chu, Sanjay G. Rao, Kunwadee Sripanidkulchai, and Hui Zhang, "Measurement-based optimization techniques for bandwidth-demanding peer-to-peer systems," in *proceedings of IEEE INFOCOM03, San Francisco*, April 2003.

[14] Jordan Ritter, "Why gnutella can't scale. no, really," in *URL http://www.tch.org/gnutella.html*, 2001.

[15] Zihui Ge, Daniel R. Figueiredo, Sharad Jaiswal, Jim Kurose, and Don Towsley, "Modeling peer-peer file sharing systems," in *proceedings of IEEE INFOCOM03, San Francisco*, April 2003.

[16] G.R. Grimmett and D.R. Stirzaker, *Probability and Random Processes, 2nd Edition*, 1995.

[17] K.B.Athreya and P.E.Ney, *Branching Processes*, 1972.

[18] S.M. Ross, *Stochastic Processes*, 1983.