Estimating Self-Sustainability in Peer-to-Peer Swarming Systems

Daniel S. Menasché[•], Antonio A. A. Rocha[†],
Edmundo A. de Souza e Silva[†], Rosa M. Leão[†], Don Towsley[•], Arun Venkataramani[•]

[•] University of Massachusetts, Amherst, MA, USA
{sadoc, towsley, arun}@cs.umass.edu

[†] Federal University of Rio de Janeiro, Rio de Janeiro, RJ, Brazil
{arocha, edmundo, rosam}@land.ufrj.br

Abstract

Peer-to-peer swarming is one of the *de facto* solutions for distributed content dissemination in today's Internet. By leveraging resources provided by clients, swarming systems reduce the load on and costs to publishers. However, there is a limit to how much cost savings can be gained from swarming; for example, for unpopular content peers will always depend on the publisher in order to complete their downloads. In this paper, we investigate such a dependence of peers on a publisher. For this purpose, we propose a new metric, namely *swarm self-sustainability*. A swarm is referred to as self-sustaining if all its blocks are collectively held by peers; the self-sustainability of a swarm is the fraction of time in which the swarm is self-sustaining. We pose the following question: how does the self-sustainability of a swarm vary as a function of content popularity, the service capacity of the users, and the size of the file? We present a model to answer the posed question. We then propose efficient solution methods to compute self-sustainability. The accuracy of our estimates is validated against simulations. Finally, we also provide closed-form expressions for the fraction of time that a given number of blocks is collectively held by peers.

1 Introduction

Peer-to-peer swarming, such as used by BitTorrent [6], is a scalable and efficient way to publish content in today's Internet. Peer-to-peer swarming has been widely studied during the last decade, and its use by enterprises is steadily growing [2, 23, 30, 31]. By leveraging resources provided by clients, peer-to-peer swarming decreases costs to publishers, and provides scalability and system robustness. As demand for content increases, system capacity scales accordingly, as all clients collaborate with each other while downloading the desired content. As the demand for multimedia files and the size of these files increase, peer-to-peer swarming systems have become an important content dissemination solution for many content providers [2, 31].

However, there is a limit on how much savings can be gained from swarming techniques. For example, in the case of unpopular content, peers must rely on the publisher in order to complete their downloads. In this paper, we investigate such a dependence of peers on a publisher.

A swarm is a set of peers interested in the same content (file or bundle of files [15]) that exchange blocks of the files among themselves. We consider a scenario where each swarm includes one stable publisher that is always online and ready to serve content. The corresponding system is henceforth referred to as a hybrid peer-to-peer system, since peers can always rely on the publisher if they cannot find blocks of the files among themselves. If all blocks are available among the peers, the swarm is referred to as *self-sustaining*. Quantifying swarm self-sustainaibility, defined as the fraction of time during which the swarm is self-sustaining, is useful for provisioning purposes. The larger the swarm's self-sustainability, the lower the dependency of peers on the publisher, and the lower the bandwidth needed by the publisher to serve the peers.

The primary contribution of this paper is a model to study swarm self-sustainability. We use a two-layer model to quantify swarm self-sustainability as a function of the number of blocks in the file, the mean upload capacity of peers and the popularity of a file. The upper layer of our model captures how user dynamics evolve over time, while the lower layer captures the probability of a given number of blocks being available among the peers conditioned on a fixed upper layer population state. Our model is flexible enough to account for large or small numbers of blocks in the file, heterogeneous download times for different blocks, and peers residing in the system after completing their downloads. We derive closed-form expressions for the distribution of the number of blocks available among the peers and apply them to show that self-sustainability increases as a function of the number of blocks in the file. The derived expressions involve sums and subtractions of large numbers, and are amenable to numerical errors. Hence, we present an efficient algorithm to compute the swarm self-sustainability that avoids these problems. We then numerically investigate the minimum popularity needed to attain a given self-sustainability level. Finally, we validate the estimates made by the model against detailed simulations.

The remainder of this paper is organized as follows. After providing a brief background into swarming systems in §2, in §3 we propose our model. In §4 we present an efficient algorithm to solve the proposed model followed by analytical results in §5. In §6 we evaluate our model against experiments. In §7 we discuss some limitations and caveats of our model, §8 presents the related work and §9 concludes the paper.

2 Swarming Systems Primer

A swarm is a set of peers concurrently sharing content of common interest. A content might be a file or a bundle of files that are distributed together. The content is divided into blocks that peers upload to and download from each other. Since there is no interaction between peers across swarms, each swarm can be studied separately.

BitTorrent is one of the most popular applications that uses peer-to-peer swarming for content dissemination, and we will use it to illustrate how swarming works. Unlike a traditional server-based system, BitTorrent includes a *tracker* that promotes the interaction of participating peers. The identities of the trackers are announced to peers in *torrent* files, which can be found and downloaded through search engines such as Torrent Finder [1]. Peers periodically query the tracker to obtain a random subset of other peers in the swarm in order to exchange (upload and download) blocks with them. Peers also discover new neighbors from other peers, in addition to the tracker, when the Peer Exchange (PEX)

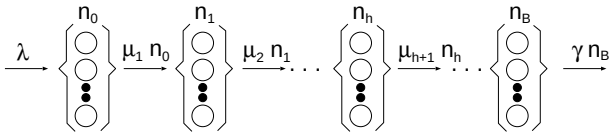

Figure 1: User dynamics. In stage h, there are n_h users, each user owning h blocks, $0 \leq h \leq B$.

extension is enabled.

There are two kinds of peers in the system: *seeds* and *leechers*. Seeds are peers that have completed the download and only upload blocks. Leechers are peers that have not completed their downloads and are actively downloading (and uploading) blocks of the file. Thus, leechers turn into seeds upon completing their downloads. Leechers adopt a tit-for-tat incentive strategy while downloading the file, i.e., leechers preferentially upload content to other leechers that reciprocate likewise, and "choke" or ignore leechers that do not reciprocate.

As many leechers may leave the system immediately after completing the download, content publishers often support a stable seed that we refer to as the *publisher*. A publisher is guaranteed to have all of the blocks constituting the file. In the rest of this paper we assume that each swarm includes one publisher.

BitTorrent peers adopt the rarest first policy to decide which blocks to download from their neighbors. According to the rarest first policy, a peer prioritizes the rarest blocks when selecting the ones to download next. We say that a peer is *interested* in another peer if the latter can provide blocks to the former. Since rarest first guarantees a high diversity of blocks in the system, any peer is almost always interested in any other peer, which in general yields high system performance [12].

Note, however, that in BitTorrent peers only have local information about the system. Hence, they can only implement a *local* rarest first policy. The intrinsic limit on the number of connections that a user can establish naturally provides each of them with only a myopic view of the system. Firewalls, NATs and other exogenous factors may also prevent users from establishing connections among themselves.

3 Model

In this section, we present our model to estimate swarm self-sustainability. The model is hierarchical in nature. The upper layer characterizes user dynamics, and the lower layer comprises a performance model used to quantify the distribution of blocks available among the peers, for a given population state. We present each of the layers, in §3.1 and §3.2, respectively, and then introduce the metric of interest in §3.3.

3.1 User Dynamics Model

A file consists of B blocks. Requests for a file arrive according to a Poisson process with rate λ. We further assume that the time required for a user to download its j^{th} block is a random variable with mean $1/\mu_j$, $1 \leq j \leq B$. After completing their downloads, peers remain in the system for mean time $1/\gamma$.

We model user dynamics with $(B+1)$ M/G/∞ queues in series. Each of the first B M/G/∞ queues models the download of a block, and capture the self-scaling property of BitTorrent swarms, i.e., each peer brings one unit of service capacity to the system. The last queue captures the residence time of seeds (see Figure 1).

The system population state is characterized by a $(B+1)$-tuple, $\mathbf{n} = (n_0, n_1, n_2, \ldots, n_B)$, where n_h represents the number of customers in queue h, i.e., the number of users that have downloaded h blocks of the file, $0 \leq h \leq B$. We denote by $\mathbf{N}$ the random variable characterizing the current state of the upper layer model and by $\mathbf{n}$ its realization. The number of peers in the system is referred to as n.

Peers arrive according to a Poisson process with rate λ to queue 0 and transit from queue h, also referred to as stage h, to queue $h+1$ $(0 \leq h \leq B-1)$ with rate μ_{h+1}, the download rate of the $(h+1)^{th}$ block downloaded by a peer. The mean residence time in queue B captures the mean time that peers remain in the system after completing their downloads, $1/\gamma$. Setting $\gamma = \infty$ models the case where all peers immediately leave the system after completing the download. Throughout this paper, unless otherwise stated, we assume that the mean download times of all blocks are the same, $1/\mu_j = 1/\mu$, $1 \leq j \leq B$. Nevertheless, all results are easily extended to the case where the mean time it takes for a user to download its j^{th} block is $1/\mu_j$.

Let $\pi(n_0, \ldots, n_B)$ be the joint steady state population probability distribution, $\pi(n_0, \ldots, n_B) = P(\mathbf{N} = (n_0, \ldots, n_B))$, of finding n_h users in the h^{th} queue, $0 \leq h \leq B$, and let $\pi_h(n_h) = P(N_h = n_h)$, $h = 0, \ldots, B$, be the corresponding marginal probability. The steady state distribution of the queueing system has the following product form,

$$\pi(n_0, \ldots, n_{B-1}, n_B) = \prod_{h=0}^{B} \pi_h(n_h) = \frac{(\lambda/\gamma)^{n_B}}{n_B!} e^{-(\lambda/\gamma)} \prod_{h=0}^{B-1} [\frac{\rho^{n_h}}{n_h!} e^{-\rho}] \tag{1}$$

where $\rho = \lambda/\mu$ is the *load* of the system (refer to Table 1 for notation).

3.2 Performance Model For a Given Population State

We now describe the lower layer of the model. Given the current population state, $\mathbf{n} = (n_0, \ldots, n_B)$, our goal is to determine the distribution of the number of blocks available among the peers. We begin by stating our key modeling assumption.

Uniform and independent block allocation: In steady state, the set of blocks owned by a randomly selected user in stage h is chosen uniformly at random among the $\binom{B}{h}$ possibilities and independently among users.

A user u in stage h, $0 \leq h \leq B$, has a signature $s_{h,u} \in \{0,1\}^B$, defined as a B bit vector where the i^{th} bit is set to 1 if the user has block i and 0 otherwise. Each user in stage h owns h blocks and has one of $\binom{B}{h}$ possible signatures.

Under the uniform and independent block allocation, signatures are chosen uniformly at random and independently among users; the latter is clearly a strong assumption since in any peer-to-peer swarming system the signatures of users are correlated. Nevertheless, in §6 we show that the effect of such correlations on our metric of interest, swarm self-sustainability, is negligible in many interesting scenarios. Therefore, we proceed with our analysis under such an assump-

parameters	
λ	mean arrival rate of peers (peers/s)
$1/\mu$	mean time to download a block (s)
$\rho = \lambda/\mu$	mean load of the system (per stage)
B	number of blocks in file
$1/\gamma$	mean residence time of seeds
variables	
n_h	number of users that own h blocks
$\mathbf{n}$=$(n_0,\ldots,n_B)$	upper layer state
$\pi(\mathbf{n})$	steady state probability of state $\mathbf{n}$
$n = \sum_{i=0}^{B} n_i$	number of peers in the system
V	number of blocks available among the peers
metrics	
$p(v) = P(V = v)$	probability of v blocks being available among peers
A=$p(B)$	swarm self-sustainability

Table 1: Table of notation. Vectors are denoted by bold face symbols. Unless otherwise stated, $\gamma = \infty$ in which case $n_B = 0$. When referring to block availability, it is subsumed availability *among peers* (excluding publisher).

tion.

Let $S_{h,u}$ be the random variable denoting the signature of the u^{th} user in stage h, and $s_{h,u}$ its realization, $1 \leq u \leq n$. The sample space of the lower layer model, $\Omega_{\mathbf{n}}$, for a given state of the upper layer, $\mathbf{n}$, is the set of all $\{0,1\}^{Bn}$ bit vectors in which element $B(u-1)+i$ equals one if the u^{th} user has block i, and zero otherwise, $1 \leq u \leq n$, $1 \leq i \leq B$. An element in $\Omega_{\mathbf{n}}$ is the concatenation of n bit vectors of size B each. $\Omega_{\mathbf{n}}$ has cardinality $|\Omega_{\mathbf{n}}| = \prod_{h=0}^{B} \binom{B}{h}^{n_h}$. Then, under the uniform and independent block allocation,

$$P(S_{1,1}{=}s_{1,1},\ldots,S_{B,n_B}{=}s_{B,n_B}|\mathbf{N}=\mathbf{n}){=}\frac{1}{|\Omega_{\mathbf{n}}|} \tag{2}$$

In the next section, we relate the upper and lower layer models, showing how (1) and (2) yield the key metric of interest, namely, swarm self-sustainability.

3.3 Self-Sustainability

We now define the key metric of interest, swarm self-sustainability. Let V denote the steady state number of blocks available among the peers. Denote by $p(v)$ the steady state probability that v blocks are available among the peers,

$$p(v) = P(V = v) = \sum_{\mathbf{n}\in\mathbb{N}^{B+1}} P(V = v|\mathbf{N} = \mathbf{n})\pi(\mathbf{n}) \tag{3}$$

Definition 3.1. *The swarm self-sustainability, A, is the steady-state probability that the peers have the entire file,*

$$A = p(B) \tag{4}$$

Definition 3.1 together with equation (3) yield,

$$A = \sum_{\mathbf{n}\in\mathbb{N}^{B+1}} P(V = B|\mathbf{N} = \mathbf{n})\pi(\mathbf{n}) = \sum_{\mathbf{n}\in\mathbb{N}^{B+1}} P(V = B|\mathbf{N} = \mathbf{n})\frac{(\lambda/\gamma)^{n_B}}{n_B!}e^{-(\lambda/\gamma)}\prod_{h=0}^{B-1}[\frac{\rho^{n_h}}{n_h!}e^{-\rho}] \tag{5}$$

The second equality in (5) follows from (1). $P(V = B|\mathbf{N} = \mathbf{n})$ is obtained from (2) (see Appendix A).

If peers leave the system immediately after concluding their downloads, we refer to the swarm self-sustainability as A_∞. The swarm self-sustainability, A, when $\gamma < \infty$, is obtained from A_∞ as follows,

$$A = 1 - (1 - A_\infty)\exp(-\lambda/\gamma), \qquad \gamma < \infty \tag{6}$$

The above follows because a block is unavailable among the peers if it is unavailable among the leechers and there are no seeds in the system.

Note that A is expressed through (5) as an infinite sum. In what follows, we approximate A by its truncated version, $A^{(N)}$, considering only population states in which there are no more than N users in the system. The value of N is based on the desired error tolerance η, and is chosen as described in the end of §4.

$$A^{(N)} = \sum_{\mathbf{n}\in\mathbb{N}^B s.t. n\leq N} P(V = B|\mathbf{N} = \mathbf{n})\pi(\mathbf{n}) \tag{7}$$

A naïve use of (7) yields an inefficient algorithm to compute $A^{(N)}$ by exploring a number of states that grows exponentially with respect to the size of the file. This problem is addressed in the next section, where we provide an efficient algorithm to evaluate $A^{(N)}$.

In the rest of this paper, we refer to the truncated self-sustainability, $A^{(N)}$, simply as self-sustainability, the distinction between $A^{(N)}$ and A being clear from the context. In addition, since A is readily obtained from A_∞ using (6), henceforth we focus on the case $\gamma = \infty$ and make the dependence of $p(v)$ on γ explicit, denoting it by $p(v;\gamma)$, whenever $\gamma < \infty$.

4 An Efficient Solution Algorithm to Evaluate Self-Sustainability

In this section we present an efficient algorithm to compute the swarm self-sustainability in polynomial time. The key insight consists of aggregating the states in the upper layer of the model in such a way that the lower layer metrics are computed once per aggregate rather than once per state. The algorithm relies on three observations about our model, the first related to the performance model for a given population state (lower layer model) and the last two related to the user dynamics (upper layer model).

Let $\psi_h(k, v)$ be the probability that, in a system in which k blocks are initially available among peers, v blocks become available among the peers after an additional user contributes h blocks. Then, $\psi_h(k, v)$ is characterized by a

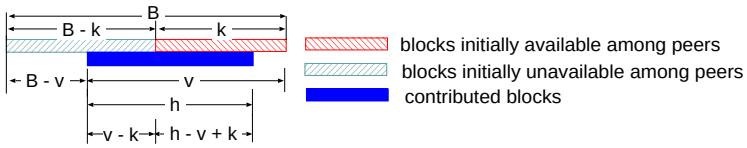

Figure 2: Recursion to compute probability of v blocks being unavailable among the peers. There are initially k blocks unavailable among the peers, and v after a user contributes h blocks.

hypergeometric distribution,

$$\psi_h(k,v) = \frac{\binom{k}{h-(v-k)}\binom{B-k}{v-k}}{\binom{B}{h}} \tag{8}$$

Equation (8) follows because there are $\binom{B-k}{v-k}$ ways in which $v-k$ blocks of the additional user do not overlap with the k previously available blocks, and there are $\binom{k}{h-(v-k)}$ ways in which the other $h-v+k$ blocks can overlap with previously available blocks (see Figure 2). A recursion to compute $\psi_h(k,v)$ is presented in Appendix B.

Our second observation regards the steady state probability that a randomly selected user is in stage h, denoted by $\sigma(h)$ $(0 \le h \le B-1)$. It can be shown that $\sigma(h) = 1/B$, $0 \le h \le B-1$. This is a consequence of the assumption that the download times of all blocks have the same mean, $1/\mu$. Note that, in general, if a user downloads its $(h+1)^{th}$ block at rate μ_{h+1}, $0 \le h \le B-1$, the model can be easily parameterized by setting $\sigma(h) = (1/\mu_{h+1})/(\sum_{i=1}^{B} 1/\mu_i)$.

Our third and last observation concerns the total number of users in the system. The total population is characterized by a Poisson random variable with mean $B\lambda/\mu$. This follows from the fact that the sum of B Poisson random variables, with mean λ/μ, is a Poisson random variable with mean $B\lambda/\mu$.

Denote by $p_n(v)$ the probability that v blocks are available among the peers conditioned on the presence of n users in the system,

$$p_n(v) = P(V = v \Big| |\mathbf{n}| = n) \tag{9}$$

It follows from the discussion in the previous paragraph that $p(v) = P(V=v) = \sum_{n=0}^{\infty} p_n(v) e^{-B\rho}(B\rho)^n/n!$. The truncated version of $p(v)$, $p^{(N)}(v)$, is

$$p^{(N)}(v) = \sum_{n=0}^{N} p_n(v) e^{-B\rho}(B\rho)^n/n! \tag{10}$$

It remains to show how to compute $p_n(v)$. This is accomplished by making use of our first two observations, as summarized by the following lemma.

Lemma 4.1. *The probability of v available blocks, conditioned on n users in the system, $p_n(v)$, satisfies the following recursion,*

$$p_n(v) = \begin{cases} \sum_{h=0}^{\min(v,B-1)} \sum_{k=v-h}^{v} p_{n-1}(k)\psi_h(k,v)/B, & n \ge 1 \\ 1, & n=0, v=0 \\ 0, & n=0, v \ne 0 \end{cases} \tag{11}$$

In Appendix C, we show that (11) correctly computes (9). Next, we further simplify recursion (11). Changing the

order of the summations and adapting their limits accordingly yields

$$p_n(v)=\sum_{k=0}^{v} p_{n-1}(k) \sum_{h=m-k}^{B-1} \psi_h(k,v)/B, \qquad n \geq 1 \tag{12}$$

The base cases are $p_0(v)$=1 if $v = 0$ and $p_0(v)$=0 if $m \neq 0$. In Appendix D we derive the following result,

$$\sum_{h=m-k}^{B-1} \psi_h(k,v) = \begin{cases} (B+1)/(B-k+1), & 0 \leq m \leq B-1 \\ k/(B-k+1), & v = B \end{cases} \tag{13}$$

Equation (13) is key to further simplifying (12). Replacing (13) into (12) yields, after algebraic manipulation,

Theorem 4.1. *The probability of v blocks being available among the peers, $p(v)$, equals*

$$p(v) = \sum_{n=0}^{\infty} p_n(v) e^{-B\rho} (B\rho)^n/n! \tag{14}$$

where $p_n(v)$ satisfies the following recursion, $0 \leq v < B$,

$$p_n(v)= \begin{cases} 1/B^n, & n \geq 1, v = 0 \\ p_n(v-1) + p_{n-1}(v)((B+1)/(B(B-v+1))), & n \geq 1, B > v > 0 \\ 1, & n = 0, v = 0 \\ 0, & n = 0, v \neq 0 \end{cases} \tag{15}$$

and $p_n(B) = 1-\sum_{v=0}^{B-1} p_n(v)$. The approximation $p^{(N)}(v)$ for (14) is obtained by truncating the infinite sum at N. $p^{(N)}(v)$ is computed in time $O(NB)$.

Theorem 4.1 yields an efficient algorithm to evaluate the swarm self-sustainability. The algorithm has complexity $O(NB)$, since $p_n(v)$ is computed for $0 \leq n \leq N$ and $0 \leq m \leq B$. Note also that once the elements $p_n(v)$ are computed for a fixed B, one can obtain the self-sustainability for different values of ρ in time $O(B)$.

Let $\varepsilon(N)$ be the truncation error, $\varepsilon(N) = p(B) - p^{(N)}(B)$. The maximum number of users in the system, N, can be chosen as a function of $\varepsilon(N)$,

$$\varepsilon(N) = \sum_{n=N+1}^{\infty} p(v) e^{-B\rho} (B\rho)^n/n! \leq \sum_{n=N+1}^{\infty} e^{-B\rho} (B\rho)^n/n! = 1 - \sum_{n=0}^{N} e^{-B\rho} (B\rho)^n/n! \tag{16}$$

If ρB is large ($\rho B > 1000$), the Poisson distribution is well approximated by a normal distribution. In this case, N can be chosen so that $1 - \Phi((N - B\rho)/\sqrt{B\rho}) \leq \eta$, where η is the desired error tolerance and $\Phi(\cdot)$ is the standard normal cdf.

Theorem 4.1 assumes $\mu_h = \mu$, $0 \leq h \leq B-1$. If that is not the case, self-sustainability can be computed in time $O(NB\log(B))$ using an alternative recursion. We refer the reader to [16] for details.

5 Model Analysis

We derive closed-form expressions for the probability that v blocks are available among the peers in the system and for the mean number of available blocks, in §5.1 and §5.2, respectively. The closed-form expressions are useful in order to gain insight on how different system parameters impact self-sustainability. In §5.2 we use the closed-form expressions to compute the minimum popularity to attain a given self-sustainability level. In §5.3 we show that the self-sustainability increases with the file size. However, the closed-form expressions may lead to numerical problems, if used to compute the self-sustainability for large files (e.g., $B > 500$), since they involve sums and subtractions of large numbers. This is why the recursion presented in §4 is useful.

In order to simplify the closed-form expressions, in the remainder of this section we assume that $\gamma = \mu$, i.e., peers, after completing their downloads, linger in the system as seeds for an interval with duration drawn from an exponential distribution with mean $1/\gamma$. Recall that when γ is finite, we make the dependence of $p_n(v)$ on γ explicit, and denote it by $p_n(v;\gamma)$.

5.1 Self-Sustainability Closed-Form Expression

Similar arguments to those in §4 (Theorem 4.1) yield, for $\gamma = \mu$ and $0 \le n \le N, 0 \le v \le B$,

$$p_n(v;\mu)=\begin{cases} 1/(B+1)^n, & n \ge 1, v = 0 \\ p_n(v-1;\mu) + p_{n-1}(v;\mu)(1/(B-v+1)), & n \ge 1, 0 < v \le B \\ 1, & n = 0, v = 0 \\ 0, & n = 0, v \ne 0 \end{cases} \tag{17}$$

This recursion can be solved (Appendix F), to obtain [1]

$$p_n(v;\mu)=\binom{B}{v}\sum_{l=0}^{v}\binom{v}{l}(-1)^l(B-v+l+1)^{-n}, \qquad 1 \le n, \qquad 0 \le v \le B \tag{18}$$

In particular, the probability that all blocks are available among the peers, conditioned on the number of users in the system, is $p_n(B;\mu) = \sum_{l=0}^{B}\binom{B}{l}(-1)^l(l+1)^{-n}$. Using this expression we derive the corresponding unconditional probability, namely, the swarm self-sustainability,

$$p(B;\mu) = \sum_{l=0}^{B}\binom{B}{l}(-1)^l e^{-(B+1)\rho l/(l+1)} = 1 - \sum_{l=1}^{B}\binom{B}{l}(-1)^{l+1} e^{-(B+1)\rho l/(l+1)} \tag{19}$$

We now interpret (19) using the inclusion/exclusion principle, which allows us to apply the Bonferroni inequalities in §5.2 and §5.3. The term $\exp(-(B+1)\rho l/(l+1))$ is the probability that l specific (*tagged*) blocks are unavailable among the peers(refer to [16] for the derivation). So, $\binom{B}{l}\exp(-(B+1)\rho l/(l+1))$ is the sum of the probabilities that *any* l blocks

[1] The expression corresponding to equation (18) for the case $\gamma = \infty$ is found in Appendix G.

are unavailable among the peers. Therefore, as a consequence of the inclusion/exclusion principle, the probability that *at least* one block is unavailable among the peers equals the rightmost summation in (19), and $p(B;\mu)$ is its complement.

In what follows, we use the above closed-form expression to analyze the mean number of blocks unavailable among the peers as well as the impact of the file size on the self-sustainability.

5.2 Minimum Load to Attain Self-Sustainability

We now provide a simple expression to estimate the minimum load necessary to attain high self-sustainability. The result relies on approximating the swarm self-sustainability using the mean number of available blocks among the peers. The mean number of available blocks among the peers, $E[V]$, is $E[V] = B - Bq$, where q is the probability that a tagged block is unavailable among the peers,

$$q = \exp(-\rho(B+1)/2) \tag{20}$$

The expression of q is readily obtained from (19). Note that the mean number of unavailable blocks, Bq, equals the first term ($l = 1$) in the rightmost summation in (19). This observation coupled with an application of the Bonferroni inequality [32] to (19) implies that $1 - p(B;\mu) \leq B - E[V]$. When $E[V] \approx B$, the upper bound $B - E[V]$ provides an approximation to the fraction of time that the swarm is not self-sustaining, $1 - p(B;\mu)$.

Next, we present a simple alternative derivation of (20). Let q_∞ be the probability that a tagged block is unavailable among leechers (excluding seeds). In order to compute q_∞, note that the mean time that a leecher holds a tagged block is $\sum_{l=0}^{B-1} l/(\mu B) = (B-1)/(2\mu)$ and the rate at which leechers acquire a tagged block is given by a Poisson process with mean rate λ. Therefore, $q_\infty = \exp(-\rho(B-1)/2)$. In general, a tagged block is unavailable among the peers if no leecher owns the block and there are no seeds in the system. The probabilities of these two events are $\exp(-\rho(B-1)/2)$ and $\exp(-\rho)$, respectively. Their product yields (20).

We now use the results above to provide a simple expression to estimate the minimum load, $\rho^\star$, necessary to attain a given self-sustainability level, $p^\star$, when $\gamma = \infty$. It follows from the discussion in the previous paragraph that if $\gamma = \infty$ the probability that a tagged block is unavailable among the peers is $q_\infty = \exp(-\rho(B-1)/2)$. For values of q_∞ close to 0 ($q_\infty \leq 0.01$), $p(B) \approx 1 + E[V] - B = 1 - Bq_\infty$, as indicated in the beginning of this section. This approximation, in turn, can be used to select the load $\rho^\star$ to attain self-sustainability level $p^\star$,

$$\rho^\star \approx \left[2\log\left(B/(1-p^\star)\right)\right]/(B-1), \qquad \gamma = \infty \tag{21}$$

We further study (21) in §6.3, where we compare the results obtained with this approximation against those obtained with recursion (15), that provides exact results.

5.3 The Impact of File Size on Self-Sustainability

Increasing the file size, B, increases the mean download time of peers, B/μ. In this section, we show that such an increase in the residence time of peers yields larger swarm self-sustainability. Theorem 5.1 states the result for $\gamma = \mu$, $B \geq 4$ and $\rho \geq 1.6$ and in §6 we provide evidence that it also holds when $\gamma = \infty$ and for small values of ρ.

Theorem 5.1 (File Size Impact)**.** *If $B \geq 4$, $\rho \geq 1.6$ and $\gamma = \mu$, self-sustainability increases with the file size, B.*

Proof. We denote the swarm self-sustainability for a given value of B and ρ as $\hat{p}(B, \rho)$. The Bonferroni inequalities [32], which generalize the inclusion/exclusion principle, applied to (19), yield upper and lower bounds on $\hat{p}(B, \rho)$ and $\hat{p}(B+1, \rho)$,

$$\hat{p}(B,\rho) \leq 1 - Be^{-(B+1)\rho/2} + (B(B-1)/2)e^{-(B+1)\rho 2/3}; \qquad 1-(B+1)e^{-(B+2)\rho/2} \leq \hat{p}(B+1,\rho) \tag{22}$$

It is easy to show that if $\rho \geq 1.6$ and $B \geq 4$ then $1 - Be^{-(B+1)\rho/2} + (B(B-1)/2)e^{-(B+1)\rho 2/3} \leq 1-(B+1)e^{-(B+2)\rho/2}$, from which the result follows. □

The key insight of Theorem 5.1 can be easily explained in terms of the busy periods of the proposed model. The busy period is defined as an uninterrupted interval during which the swarm is self-sustaining. As the file size increases, the number of blocks that need to be maintained increases linearly but the busy period of the system increases exponentially [15]. Indeed, as the file size increases the availability gain compensates the overhead to maintain a larger number of blocks, and the self-sustainability increases.

6 Evaluation

In this section we report (*a*) a validation of the proposed model, against detailed simulations, showing that despite the simplifying assumptions considered in our model, it captures how self-sustainability depends on different system parameters and (*b*) results on the minimum popularity to attain a given self-sustainability level.

6.1 Experimental Setup

Our simulation experiments were conducted using the Tangram II modeling environment [8]. Tangram II is an event-driven, object oriented modeling tool. The three main objects in our simulations are the tracker, the peer and the seed. Their implementations are based on the official BitTorrent protocol description [6, 12].

6.1.1 Simulator and Protocol Descriptions

When a peer P joins the system, it receives a random list of fifty other peers from the tracker, which constitutes its *peer set*. Throughout the simulation, as peers leave the system the size of the peer set of P may dwindle to less than twenty. Once the peer set size is less than twenty, P requests additional neighbors from the tracker. The set of peers to whom P offers content blocks is a subset of its peer set, referred to as the *active peer set*.

BitTorrent proceeds in rounds of ten seconds. By the end of each round, peer P runs the tit-for-tat incentive mechanism. According to this mechanism, P reciprocates contents with those neighbors that contributed in the previous round. P selects r of those peers that contributed in the last round to add to its active peer set ($r \leq 4$). In the next round, the active peer set of P will consist of the r aforementioned peers plus $5 - r$ additional peers selected uniformly at random out of its peer set. This random selection of peers is referred to as *optimistic unchoke*, performed to allow peers to get bootstrapped as well as to let them learn about new neighbors. Finally, peers select blocks to download using the rarest first algorithm, except for the first four blocks, which are chosen uniformly. Each block of the file is divided into sixteen sub-blocks. After selecting the blocks to download, peers can get different sub-blocks (of the same block) from different neighbors concurrently. A block can be uploaded after all its sub-blocks are downloaded, ensembled and checked using the block hash key.

In our experiments we observe that self-sustainability decreases with the size of the active peer sets. That is because, if the active peer sets are large, each peer splits its bandwidth across many other peers and blocks take longer to get replicated in the network. As mentioned above, we select an active peer set of size five, which is adopted by many BitTorrent implementations.

In our experiments the seed behaves as a standard peer, except that it (i) initially owns all blocks and (ii) is altruistic, hence does not execute the tit-for-tat algorithm. We did not implement the mechanism used by peers to download their last block, also known as *end game mode* [6]. This is inconsequential, though, since the end game mode does not significantly affect the steady state behavior of the system (see §6.2.1 and [4]).

Every time a peer enters the system, receives a block or leaves, we record the event in our logs, the current timestamp, peer id, and signature (see §3.2).[2]

6.1.2 Experimental Parameters

The configuration of our experiments consists of torrents that publish a file of size S divided into B blocks of size s, $s = 256$KB, a typical block size in BitTorrent. The number of blocks in the file, B, takes values 16, 50, 100 and 200, which corresponds to files of size 4MB, 12MB, 25MB and 51MB, respectively. Note that if a swarm is constituted of multiple files which can be downloaded separately, we are interested in analyzing the self-sustainability of each individual file. A file of size 51MB already yields self-sustainability larger than 0.9 for $\lambda > .05$ peers/min (see Figure 4). Simulations to analyze such a steep increase in self-sustainability quickly yield prohibitively high running times and significant variability in the metrics of interest across runs. For this reason, we focused on quantitatively validating our model for files with up to 200 blocks, but also use the model to analyze files of size greater than 200 blocks.

The uplink capacity of each peer is 39KBps, which corresponds to $\mu = 39/256 = 0.15$ blocks/s, a typical effective capacity for BitTorrent peers [22, Figure 1]. The publisher maximum upload capacity is the same as that of a peer. A publisher that contributes the same capacity as an ordinary peer might correspond to either a domestic user or a commercial publisher that supports a large catalog of titles and only provides enough capacity for each swarm so as to

[2]Our simulator as well as the traces generated for this study will be made available in public domain.

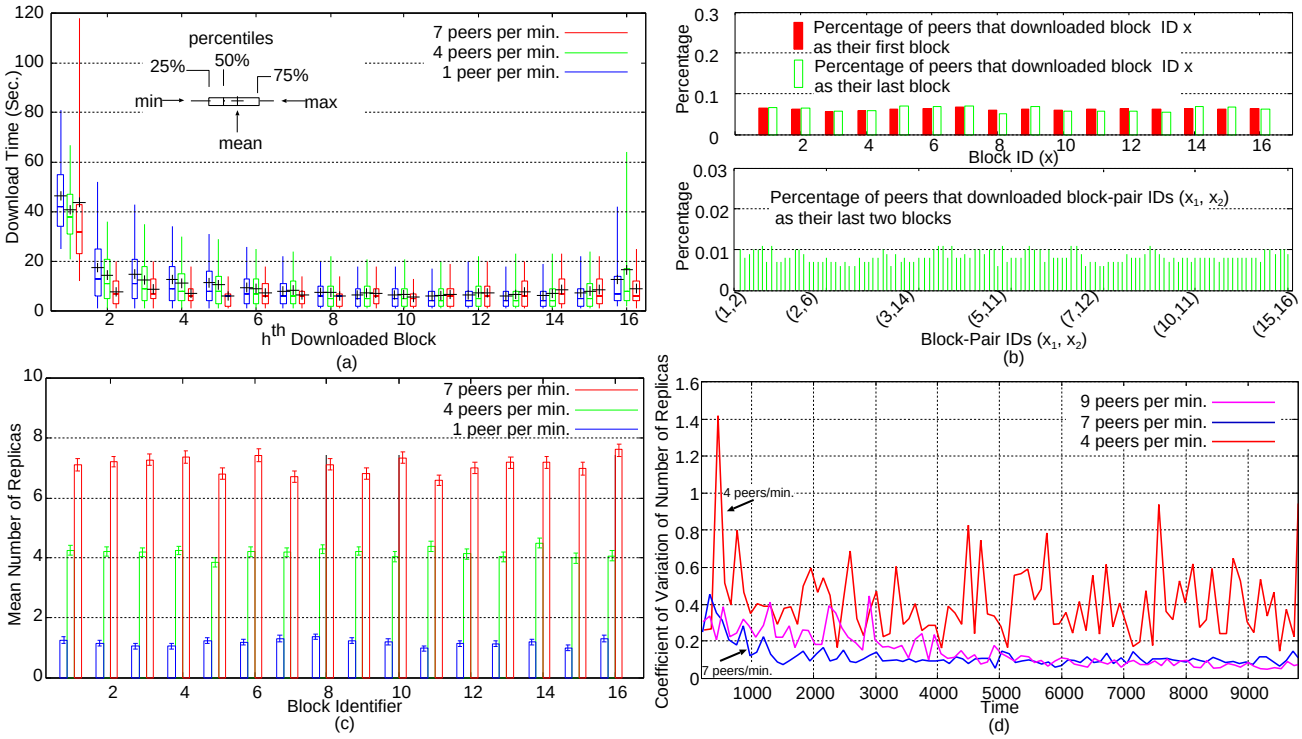

Figure 3: (a) Download times of h^{th} block downloaded by a peer. The boxplots show the four quartiles and crosses indicate means. (b) Top: light (resp., shaded) rectangles are percentage of peers that download block x as their first (resp., last) block. Bottom: percentage of peers that download block-pair (x_1, x_2) as their last two blocks. (c) Mean number of replicas of each block with 95% confidence intervals. (d) Coefficient of variation of number of replicas of blocks versus time.

allow peers to complete their downloads at rate μ. The peer arrival rate is varied according to the experimental goals between $(.25, .5, 1, 2, \ldots, 9)$ peers/min, as described next.

6.2 Model Validation

Our analytical model makes a number of simplifying assumptions as described in §3. In what follows, our goal is to show that even with those simplifying assumptions, discussed in §6.2.1, our model still captures swarm self-sustainability in a realistic setting, as shown in §6.2.2.

6.2.1 Validating Model Assumptions

Our aim in this section is to validate (a) that the mean download times of blocks are roughly the same (an exception being the first block downloaded by the peers), i.e., $\mu_h = \mu$, $2 \leq h \leq B$ (see §3.1) and (b) that the signatures of the users are uniformly distributed (see §3.2).

Figure 3(a) shows the mean download time of the h^{th} block downloaded by a peer. The boxplots and lines show the distribution quartiles and minimum and maximum values. Crosses indicate means download times of blocks, which are approximately the same, except for the first and last blocks. In particular, even though our simulator ignores the *end game mode* [4], in general peers do not experience difficulty finding a neighbor from whom to download their last block.

The median of the last block is roughly the same as the one observed for the other blocks, and the mean is only slightly larger. The first block requested by a peer, however, takes longer to be downloaded. This happens because a peer can only download its first block after being optimistically unchoked (see §6.1.1). Although this affects the time spent by peers in stage zero of our model and as a consequence the total download time, it is inconsequential to our self-sustainability estimates (the time that peers remain in stage zero of the upper layer model has no influence in our results). As BitTorrent peers downloading their first block cannot upload content, they also don't contribute to self-sustainability (see §6.1.1).

Our second goal is to study the users' signatures distribution (§3.2). For this purpose, we validated that the two first and two last blocks downloaded by a user are indeed uniform and then studied one of the consequences of the uniform and independence assumption, namely, that the number of replicas of each block in the system is well balanced.

Figure 3(b) (top) shows, for each block, the fraction of peers that downloaded that block as their first block (light bars). The figure was generated from independent samples: every 500 seconds, one user owning one block was selected at random, and the identity of its block was recorded (the same procedure was repeated for users owning all but one block of the file (shaded bars)). Similarly, Figure 3(b) (bottom) shows, for each $B(B-1)$ block-pairs, the fraction of peers that downloaded that pair as their first two blocks. Figure 3(b) indicates that the first and last blocks downloaded by users, as well as the first block-pair downloaded, are approximately uniformly distributed (the same procedure was repeated for users owning all but two blocks, with similar results).

Figure 3(c) shows the mean number of replicas of each block for $\lambda = 1$ peer/min, 4 peers/min and 7 peers/min. The mean number of replicas is around $\rho(B-1)B/2$, which corresponds to a well balanced system (see §5.2). Figure 3(d) corroborates this claim by showing the coefficient of variation of the number of replicas of blocks as a function of time. Let $r_{i,t}$ be the number of replicas of block i at time t. The mean number of replicas of blocks at time t is $\mu_t = \sum_{i=1}^{B} r_{i,t}/B$ and the coefficient of variation is $c_t = \sqrt{(\sum_{i=1}^{B}(r_{i,t}-\mu_t)^2/B)}/\mu_t$. Figure 3(d) indicates that throughout the simulation, the coefficient of variation is most of the time smaller than 0.8, which means that the number of replicas of blocks has low variance. In a system where users' signatures are uniform and independent we would observe similar behavior.

6.2.2 Validating Model Estimate of Self-Sustainability

To study how content popularity impacts self-sustainability, we simulated BitTorrent in the reference setting described in §6.1.2, varying the arrival rate of peers, λ, from 1 peer/minute up to 8 peers/minute, in increments of 1 peer/minute, and keeping all other parameters fixed. Equivalently, this corresponds to an increase in the load, $\rho = \lambda s/\mu$, from 0.1 to 0.8. Each simulation lasted for 10,000s. Twenty one independent simulations were executed for each value of λ, and used to compute the 95% confidence intervals. The same experiment is repeated for $B = 16$, 50, 100 and 200.

Figure 4 shows the self-sustainability, A, as a function of the content popularity, λ, for $B = 16, 50, 100$ and 200. For unpopular contents, $\lambda = 1$ peer/min, and small files, $B = 16$, the swarm self-sustainability is around 0.1 and the publisher needs to frequently provide blocks that are unavailable among the peers. As the popularity of the files increases, swarm self-sustainability increases and content is available even in the absence of the publisher. For $\lambda = 8$ peers/min, the fraction

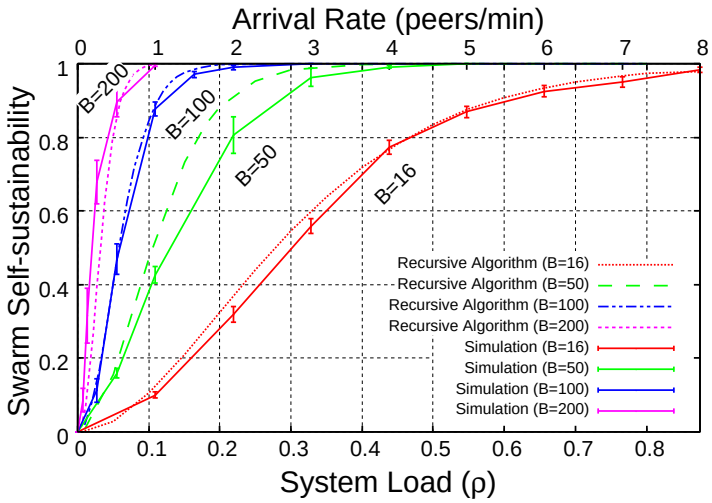

Figure 4: Model validation. Swarms-self sustainability as a function of the system load. Results obtained with recursive algorithm and simulations (with 95% confidence intervals) are plotted with dotted and solid lines, respectively.

of time at which the publisher needs to provide blocks to peers is close to zero.

Figure 4 indicates that the results obtained with our model predict the simulation well. Even assuming that the mean download times of all blocks are the same ($\mu_h = \mu$, for $1 \leq h \leq B$) in the model, it was able to capture the trend of self-sustainability observed in our simulations. We repeated the simulations for $B = 16$, with heterogeneous peer upload capacities, the upload rate distribution taken from the measured data used to generate Figure 1 in [22] (normalized to a mean upload rate of 39KBps) and our results did not qualitatively change (details in [16]).

Consider now the impact of file size on swarm self-sustainability. Figure 4 shows that for a fixed content popularity, as the file size increases, self-sustainability increases. This is in accordance to Theorem 5.1, and reflects the fact that, as the file size is increased, peers stay longer in the system and the coverage, defined as the mean number of users in the system, increases. The higher the coverage, the higher the self-sustainability of the swarm. In fact, as file size increases, the number of blocks that needs to be maintained by the publisher increases linearly but the availability gain increases exponentially [15].

6.3 Popularity to Attain High Self-Sustainability

We now address the following question: what is the minimum file popularity (or load) needed to attain a given self-sustainability? Answering this question is useful not only for publisher dimensioning but also for other strategic decisions such as how to distribute and bundle files across multiple swarms [15].

Recall that the load is defined as $\rho = \lambda/\mu$. Figure 5 shows the minimum load, $\rho^\star$, necessary to achieve high self-sustainability, $A^\star$ ($A^\star = 0.9, 0.999$), for file sizes varying between 2MB and 256MB ($B = 8, \ldots, 1,000$). The solid curves are obtained using the recursions contained in Theorem 4.1, eq. (15). The dotted curves are obtained using the approximation (21). Figure 5(a) suggests that when the goal is to find the minimum popularity to attain a high self-sustainability level, equation (21) can be used to approximate $\rho^\star$. However, if the goal is to compute self-sustainability under different loads and for different files sizes, as illustrated by Figure 4, the recursion provided by Theorem 4.1 needs

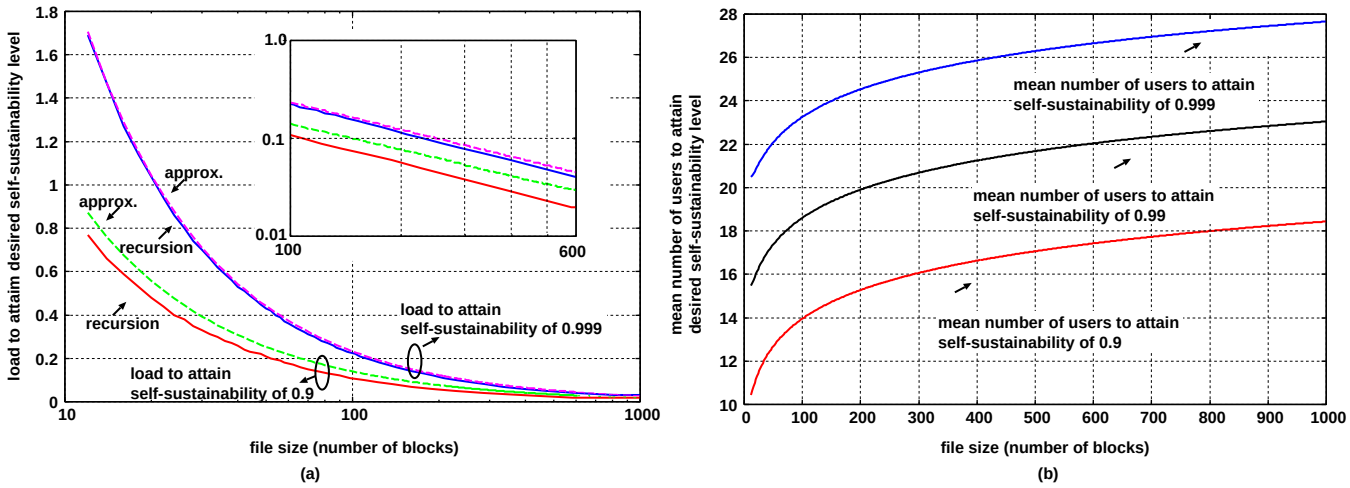

Figure 5: Larger files yield increased availability. (a) x-axis, file size. y-axis, necessary load ($\rho^\star = \lambda/\mu$) to attain self-sustainability greater than 0.9 (red) and 0.999 (blue). The results obtained using the approximation (21) are also shown. (b) the mean number of users in the system, $B\rho^\star$, to attain a desired self-sustainability level.

to be used.

Figure 5(a) indicates that the popularity, $\rho^\star$, needed to attain a high degree of self-sustainability increases as the file size, B, decreases. In particular, the zoom in the figure shows that $\log \rho^\star$ is linear in $\log B$. The comments made at the end of §6.2 to explain Figure 4 also apply here. Peers take longer to download larger files, which increases block availability. Nevertheless, the benefits of leveraging peer-to-peer swarming can be noted even for small files. Figure 4(a) shows that for a file of 4MB ($B = 16$), an arrival rate of 8 peers/min (which corresponds to a load of 0.8) already yields a very high self-sustainability.

More insights on how the file size impacts self-sustainability are obtained from Figure 4(b). Figure 4(b) plots the mean number of users in the system, also referred to as the mean coverage [15], necessary to attain a high level of self-sustainability. The curves in Figure 4(b) correspond to the the respective ones in Figure 4(a) multiplied by B. The main insight shown in this example is that the coverage necessary to attain a given self-sustainability level slowly increases as a function of B. As the file size increases, a slightly larger population suffices to attain a given self-sustainability level. For instance, for a file of size 10 a coverage of 20 is necessary to attain self-sustainability of 0.999, whereas a coverage of 29 suffices to achieve the same self-sustainability if the file has 1,000 blocks.

7 Limitations and Caveats

Next, we discuss the simplifications adopted to yield a tractable model.

Uniformity and independence assumptions: In the performance model presented in §3.2 we assume that the signatures of users are drawn uniformly and independently at random. In particular, we do not account for the correlations among users' signatures. Although such correlations are present in practice, our simulations have indicated that the independence assumption is appropriate in order to capture the self-sustainability of swarms.

In the user dynamic model presented in §3.1 we make the following assumptions: (i) peers arrive according to a

Poisson process with rate λ (steady state assumption), (*ii*) the download times of all blocks have the same mean, $1/\mu$ (smooth download assumption) and (*iii*) users leave the system immediately after completing their downloads, $\gamma = \infty$ (self-regarding users assumption). We discuss each of them in turn.

Steady state assumption: It has been shown in [15, Section 4.3.4] that a vast number of long-lived swarms have relatively stable mean arrival rates over periods of months. Our model can be used to predict the self-sustainability of such swarms.

Smooth download assumption: Under this assumption, the capacity of the system scales perfectly with the number of users. Our simulations indicate that the mean download time of the first and last blocks are slightly larger than the others. Although our model has flexibility to capture such asymmetries (see observation two in §4), we show that their implications in the estimates of self-sustainability are not significant (see §6.2.1).

Self-regarding users assumption: Our model has flexibility to account for users that stay in the network after completing their downloads. However, in todays' BitTorrent users have no incentive to stay in the system after obtaining the files of their interest. Therefore, we focus on the worst case scenario in which users, not having incentives to linger in the system after completing their downloads, depart immediately.

Finally, in our simulations we consider a publisher that is always online and that behaves like a typical peer.

Typical peer-like publisher: Our simulations indicate that if the publisher has the same capacity as typical peers, the smooth download assumption holds and the swarm self-sustainability estimates of our model are accurate. Coping with intermittent publishers and devising dynamic bandwidth allocation strategies according to which the smooth download assumption holds is non trivial, and is subject of future work.

8 Related Work and Discussion

Modeling of Peer-to-Peer Swarming Systems

The literature on availability [19], performance [24] and incentive issues [22] in BitTorrent-like swarming systems is vast. Nevertheless, to our knowledge this paper presents the first analytical model for publisher dependency estimation. We are unaware of related analytical work that analyzed swarm self-sustainability taking into account the fact that files are divided into multiple blocks, a very fundamental characteristic of these systems.

For large populations, Massoulié and Vojnovic [14] used a coupon collector model to show that asymptotically the distribution of blocks across the population is well balanced, and does not critically depend on the block selection algorithm used by the peers. Qiu and Srikant [24] and Fan et al. [9] also considered the large population regime, and used fluid approximations and differential equations to model the system assuming that the efficiency is always high. In this paper we are particularly interested in the small population regime. For small populations, Markov Chain (MC) models have been proposed by Veciana et al. [35], providing insights on the performance of the system but not dealing with the problem of availability of blocks among peers. Reittu et al [25], using a different model, studied the dissemination of a two-block file in a closed network accounting for the availability of the blocks. In this paper, we consider an open network and propose

a model which can be used to estimate self-sustainability of files of arbitrary size.

In this work we studied the implications of the content popularity on the self-sustainability of swarms. In a real time setting, Leskela et al. [13] pointed out a phase transition in the stability of the peer-to-peer system as a function of the content popularity. In contrast, our system is always stable. Norros et al. [20] imply a phase transition of the mean broadcast times as a function of the departure rate of seeds. In this paper we are concerned with self-sustainability.

Hybrid Peer-to-Peer Swarming Systems

The literature on the use of peer-to-peer swarming systems for enterprise content delivery is rapidly growing [7, 21, 26]. The methodology usually consists on defining an optimization problem to be solved by the publishers and then showing how different system parameters affect the optimal strategy for bandwidth allocation. The approach we take in this paper is different. We are interested in the *minimum* fraction of time that the publisher must be active so as to guarantee that all blocks are always available.

Ioannidis et al. [10] study how fast does the bandwidth available at the server has to grow as the number of users increases. For this purpose they consider two query propagation mechanisms, the random walk and the expanding ring. Here, on the other hand, we assume that peers can always find the blocks they need in case they are available. While [10] focus on the control plane and in asymptotic analysis, here we focus on the data plane and account also for small files.

Menasche et al. [15], Altman et al. [34] and Susitaival et al. [29] propose models for content availability in BitTorrent without accounting for the fact that files are divided into blocks. Our model differs from [15, 29, 34] in that we (*a*) consider an hybrid peer-to-peer system, in which a publisher is always available and (*b*) account for the fact that the file is divided into blocks. Finally, explicit scheduling of blocks exchanges to minimize peer download times was studied by Mundinger et al. [18]. In this paper we assume that peers exchange blocks using only local information, as in BitTorrent.

Balls and Bins

The derivation of some of our results fit into the balls and bins framework. Each user is allocated a set of blocks (balls) each of which must correspond to a different identifier (bin). In the context of balls and bins, a set of balls each of which must be allocated in a different bin is referred to as complex [11]. Previous work on the allocation of complexes into bins appears in Saidbek et al. [17], Kolchin et al. [11, Chapter VII], and references therein. In particular, the definition of $\psi_h(i, m)$ in this paper was inspired by [5, Figure 1].

In a peer-to-peer setting, balls and bins were used by Simatos et al. [28] to study the duration of the regime during which the system is saturated because capacity is smaller than demand. The scenario studied in this paper differs from [28] in several aspects. For instance, [28] considers a finite population of peers.

9 Conclusion

Peer-to-peer swarming systems are a powerful tool for content delivery, as reflected by the immense popularity of BitTorrent and the vast literature on the topic. However, most works in this area have focused on the dissemination of popular content,

for which peer-to-peer systems are naturally suitable. In this work, we investigate the dependency of peers on a publisher that leverages peer-to-peer techniques for the dissemination of both popular and unpopular content. In particular, the latter deserve special attention, since unpopular content can represent a significant fraction of demand and revenue [3]. We believe that devising strategies for disseminating large catalogs of files leveraging peer-to-peer techniques is an important and interesting research area, and we see our model as a first attempt to shed light into the intrinsic advantages and limitations of peer-to-peer swarming systems for the dissemination of such catalogs.

References

[1] Torrent Finder. `http://torrent-finder.com/`.

[2] Amazon. Using BitTorrent with Amazon S3. http://aws.amazon.com/.

[3] Anderson, C. *The Long Tail: Why the Future of Business is Selling Less of More*. Hyperion, 2006.

[4] Bharambe, A., Herley, C., and Padmanabhan, V. Analyzing and improving BitTorrent network performance mechanisms. In *Infocom* (2006).

[5] Burger, A. P., and van Vuuren, J. H. Balanced minimum covers of a finite set. *Discrete Mathematics 307* (2007), 2853–2860.

[6] Cohen, B. Incentives build robustness in BitTorrent. In *P2PECON* (2003).

[7] Das, S., Tewari, S., and Kleinrock, L. The case for servers in a peer-to-peer world. In *ICC* (2006).

[8] de Souza e Silva, E., Leao, R. M. M., and Figueiredo, D. R. An integrated modeling environment for computer systems and networks. *Performance Evaluation Review 36*, 4 (2009), 64–69.

[9] Fan, B., Chiu, D.-M., and Lui, J. Stochastic differential equation approach to model peer to peer systems. In *ICC* (2006).

[10] Ioannidis, S., and Marbach, P. On the design of hybrid peer-to-peer systems. In *SIGMETRICS* (2008).

[11] Kolchin, V., Sevastyanov, B., and Chistyakov, V. *Random Allocations*. V. H. Winston and Sons, 1978.

[12] Legout, A., Liogkas, N., and Kohler, E. Rarest first and choke algorithms are enough. In *IMC* (2006).

[13] Leskela, L., Robert, P., and Simatos, F. Stability properties of linear file sharing networks. *arXiv.org* (2009).

[14] Massoulie, L., and Vojnovic, M. Coupon replication systems. In *SIGMETRICS* (2006).

[15] Menasche, D., Rocha, A., , Li, B., Towsley, D., and Venkataramani, A. Content availability and bundling in swarming systems. In *CONEXT* (2009).

[16] Menasche, D. S., Rocha, A. A., de Souza e Silva, E., Leao, R. M., Towsley, D., and Venkataramani, A. Estimating self-sustainability in peer-to-peer swarming systems. *UMass Technical Report* (2010).

[17] Mirakhmedov, S. S., and Mirakhmedov, S. M. On asymptotic expansion in the random allocation of particles by sets. *Journal of Theoretical Probability* (2009).

[18] Mundinger, J., Weber, R., and Weiss, G. Optimal scheduling of peer-to-peer file dissemination. *Journal of Scheduling* (2008).

[19] Neglia, G., Reina, G., Zhang, H., Towsley, D., and Arun Venkataramani, J. D. Availability in Bittorrent systems. In *IEEE Infocom* (2007).

[20] Norros, I., Prabhu, B. J., and Reittu, H. On uncoordinated file distribution with non-altruistic downloaders. In *ITC-20* (2007).

[21] Peterson, R. S., and Sirer, E. G. Antfarm: efficient content distribution with managed swarms. In *NSDI* (2009).

[22] Piatek, M., Isdal, T., Anderson, T., Krishnamurthy, A., and Venkataramani, A. Do incentives build robustness in Bittorrent? In *4th USENIX Symposium on Networked Systems Design & Implementation* (2007).

[23] Pollack, P. Warner Bros and P2P. http://arstechnica.com/news.ars/post/20060130-6080.html.

[24] Qiu, D., and Srikant, R. Modeling and performance analysis of Bittorrent-like peer to peer networks. In *SIGCOMM* (2004).

[25] Reittu, H., and Norros, I. Toward modeling of a single file broadcasting in a closed network. In *IEEE Workshop on Spatial Stochastic Models in Wireless Networks* (2007).

[26] Rimac, I., Elwalid, A., and Borst, S. On server dimensioning for hybrid peer-to-peer content distribution networks. In *P2P'08* (2008).

[27] Schneider, C. Symbolic summation assists combinatorics. *Sem.Lothar.Combin. 56* (2007), 1–36.

[28] Simatos, F., Robert, P., and Guillemin, F. Analysis of a queueing system for modeling a file sharing principle. In *SIGMETRICS* (2008).

[29] Susitaival, R., Aalto, S., and Virtamo, J. *Lecture Notes in Computer Science*. Springer, 2006, ch. Analyzing the dynamics and resource usage of P2P file sharing by a spatio-temporal model.

[30] Torrent Freak. Twitter uses BitTorrent for server deployment. http://torrentfreak.com/twitter-uses-bittorrent-for-server-deployment-100210/.

[31] Ubuntu. Download Ubuntu using BitTorrent. http://torrent.ubuntu.com:6969/.

[32] Wikipedia. Boole's inequality. http://en.wikipedia.org/wiki/Boole%27s_inequality.

[33] Wilf, H. *Generatingfunctionology*. Acad. Press, 1994.

[34] Wong, S., Altman, E., and Ibrahim, M. P2P networks: interplay between legislation and information technology. In *INRIA 6889* (2009).

[35] Yang, X., and De Veciana, G. Service capacity of peer to peer networks. In *INFOCOM* (2004).

[36] Zeilberger, D. The method of creative telescoping. *Journal of Symbolic Computation archive 11* (1991), 195–204.

A An Expression of $P(V = B|\mathbf{N} = \mathbf{n})$

We now show how to obtain $P(V = B|\mathbf{N} = \mathbf{n})$ from (2). Recall that, given an upper layer state $\mathbf{n}$, Ω is the lower layer sample space. Denote by $\mathbf{S}$ the random variable that represents the lower layer state, and by $\mathbf{s}$ its realization. Then

$$P(V = B|\mathbf{N} = \mathbf{n}) = \sum_{\mathbf{s}:V=B,\mathbf{s}\in\Omega_{\mathbf{n}}} P(\mathbf{S} = \mathbf{s}|\mathbf{N} = \mathbf{n}) \stackrel{(2)}{=} \sum_{\mathbf{s}:V=B,\mathbf{s}\in\Omega_{\mathbf{n}}} 1/|\Omega| \tag{23}$$

Therefore, the problem of computing $P(V = B|\mathbf{N} = \mathbf{n})$ is reduced to that of counting the states in which all blocks are available among the peers. Using the inclusion/exclusion principle, $P(V = B|\mathbf{N} = \mathbf{n}) = |\Omega_{\mathbf{n}}|^{-1} \sum_{i=0}^{B-1} (-1)^i \binom{B}{i} \prod_{j=1}^{B-1} \binom{B-i}{j}^{n_j}$. In general, $P(V = v|\mathbf{N} = \mathbf{n})$ is also obtained using the inclusion/exclusion principle [33, Section 4.2].

B Recursion to Compute $\psi_h(k, v)$

Consider the scenario where k blocks are available among the peers and an additional user contributes h blocks. $\psi_h(k, v)$ is the probability that v blocks are available among the peers after accounting for the blocks contributed by the additional user. $\psi_h(k, v)$ can be recursively computed,

$$\psi_h(k, v) = \psi_{h-1}(k, v-1)\frac{B-v-h}{B-h+1} + \psi_{h-1}(k, v)\frac{v}{B-h+1}, \qquad 0 \le k, v \le B, 0 < h \le B \tag{24}$$

The base cases are $\psi_0(k, v) = 0$ if $v \neq k$ and $\psi_0(k, v) = 1$ if $v = k$.

For presentation convenience, we consider an arbitrary ordering of the h blocks contributed by the additional user. After contributing the first $h-1$ blocks, there are two cases to consider, (i) v blocks are available among the peers [and the h^{th} block overlaps with a previously available block, an event which happens with probability $v/(B-h+1)$] or (ii) $v-1$ blocks are available [and the h^{th} block does not overlap with previously available blocks, an event which happens with probability $(B-v-h)/(B-h+1)$]. The above recursion is convenient to avoid numerical problems since it only involves additions and multiplications of probabilities.

C Proof of Lemma 4.1

Proof. We now show that (11) and (9) are equivalent. To this purpose, we consider an extended representation of the upper layer states. In such representation, the upper layer state, $\mathbf{n}'$, is defined as follows.

Consider the users in the system ordered uniformly at random. The number of blocks owned by the i^{th} user is denoted by n'_i. The upper layer state is represented by vector $\mathbf{n}'$. The dimension of $\mathbf{n}'$ is $1 \times n$, where n is the number of users in the system.

Note that $\mathbf{n}$ is inferred from $\mathbf{n}'$. Let $n_l(\mathbf{n}')$ be the number of users that have l blocks when the state is $\mathbf{n}'$. The steady state probability of state $\mathbf{n}'$ is $\pi(\mathbf{n}')$, $\pi(\mathbf{n}') = \prod_{l=0}^{B-1}[\frac{\rho^{n_l(\mathbf{n}')}}{n_l(\mathbf{n}')!}e^{-\rho}]/n!$; any permutation of the users in the system is equally likely. The steady state probability of state $\mathbf{n}'$, conditioned on the event that there are n users in the system, is

$$\frac{\pi(\mathbf{n}')}{\sum_{\forall \mathbf{n}':|\mathbf{n}'|=n}\pi(\mathbf{n}')}=\frac{\prod_{l=0}^{B-1}[\frac{\rho^{n_l(\mathbf{n}')}}{n_l(\mathbf{n}')!}e^{-\rho}]/n!}{e^{-B\rho}(B\rho)^n/n!} = \frac{1}{B^n\prod_{l=0}^{B-1}n_l(\mathbf{n}')!} \tag{25}$$

The first equality follows from the fact that a convolution of B Poisson processes with rate ρ is a Poisson process with rate $B\rho$.

The key idea of the proof consists of partitioning the state space into sets $G_h(n)$, $0 \le h \le B-1$, $0 \le n$. Set $G_h(n)$ contains states in which there are n users in the system and $n'_n = h$. $G_h(n)$ is defined as $G_h(n) = \{\mathbf{n}' : n'_n = h \text{ and } 0 \le n'_j \le B-1, 1 \le j < n\}$. The set containing all states in which there are n users in the system, $G(n)$, is

$$G(n) = \cup_{h=0}^{B-1}G_h(n) = \{\mathbf{n}' : 0 \le n'_j \le B-1, 1 \le j \le n\} \tag{26}$$

Note that $G_h(n)$ is obtained from $G(n-1)$ by adding to each element of $G(n-1)$ a user that owns h blocks,

$$G_h(n) = \{(\mathbf{n}', 0) + h\mathbf{1}_n : \mathbf{n}' \in G(n-1)\} \tag{27}$$

$\mathbf{1}_n$ denotes a $1 \times n$ vector in which all elements equal zero, except element n, which equals one. Let $p(v|\mathbf{n}')$ be the probability of v blocks being available among the peers when the upper layer state is $\mathbf{n}'$. If $\mathbf{n}' \in G_h(n)$ $(n > 0)$,

$$p(v|\mathbf{n}') = \sum_{k=0}^{B} p(k|\mathbf{n}' - h\mathbf{1}_n)\psi_h(k, v), \qquad \mathbf{n}' \in G_h(n) \tag{28}$$

According to definition (9),

$$p_n(v)=\left[\sum_{\mathbf{n}'\in G(n)} p(v|\mathbf{n}')\pi(\mathbf{n}')\right] \Big/ \left[\sum_{\mathbf{n}'\in G(n)}\pi(\mathbf{n}')\right] = \sum_{\mathbf{n}'\in G(n)} p(v|\mathbf{n}')\frac{1}{B^n\prod_{l=0}^{B-1}n_l(\mathbf{n}')!} \tag{29}$$

where the first and second equalities follow from (26) and (25), respectively. Hence,

$$p_n(v) = \sum_{h=0}^{B-1}\sum_{\mathbf{n}'\in G_h(n)} p(v|\mathbf{n}')\frac{1}{B^n\prod_{l=0}^{B-1}n_l(\mathbf{n}')!}=\sum_{h=0}^{B-1}\sum_{\mathbf{n}'\in G_h(n)}\sum_{k=0}^{B} p(k|\mathbf{n}' - h\mathbf{1}_n)\psi_h(k, v)\frac{1}{B^n\prod_{l=0}^{B-1}n_l(\mathbf{n}')!} \tag{30}$$

$$=\sum_{h=0}^{B-1}\sum_{k=0}^{B}\underbrace{\sum_{\mathbf{m}'\in G(n-1)} p(k|\mathbf{m}')\frac{1}{B^{n-1}\prod_{l=0}^{B}n'_l!}}_{p_{n-1}(k)}\psi_h(k, v)\frac{1}{B}=\sum_{h=0}^{B-1}\sum_{k=0}^{B}p_{n-1}(k)\psi_h(k, v)\frac{1}{B} \tag{31}$$

where the first, second and third equalities follow from (26), (28) and (27), respectively. Note that the summands in (31) in which $h > v$ or $k \in [0, v-h-1] \cup [v+1, B]$ are equal to zero, since in these cases $\psi_h(k, v) = 0$. Therefore, (31) yields (11). □

D Probabilistic Derivation of (13)

Next, we provide a probabilistic derivation of (13). An algebraic proof can be obtained using the Sigma package [27], applying a paradigm called creative telescoping [36].

Proof. Henceforth, we consider the case $v < B$ (the case $v = B$ follows similarly). The probabilistic interpretation for (13) follows from the connection between $\psi_h(k, v)$ and the hypergeometric distribution. Suppose we have an urn containing B balls, $B - k$ of which are white (unavailable blocks) and k are black (available blocks). $\psi_h(k, v)$ is the probability of selecting without replacement h balls, of which $v - k \leq h$ are white.

Consider now another experiment, namely selecting without replacement all balls from the urn. Let J_w be the round in which the w^{th} white ball is selected, $0 < J_1 < J_2 < \ldots < J_{B-k} < B + 1$. Let W_b be the number of black balls selected between the b^{th} and $b + 1^{th}$ white, plus one. Equivalently, W_b is the number of elements in the set $S = \{n \in \{0, \ldots, B\} :$ exactly b white balls are selected before ball $n\}$. Then $W_0 = J_1$, $W_1 = J_2 - J_1$, ..., $W_{k-1} = J_k - J_{k-1}$, and $W_{B-k} = B + 1 - J_i$. Clearly, $\sum_{b=0}^{B-k} W_b = B + 1$. By symmetry, $E[W_b] = (B + 1)/(B - k + 1)$ $(0 \leq b \leq B - k)$.

Now let $v - k$ be given, $v - k \in \{0, 1, \ldots, v\}$. Let $\mathbf{1}_h$ be the indicator equal to 1 if exactly $v - k$ white balls are selected among the first h balls. Note that $E[\mathbf{1}_h] = \psi_h(k, v)$. Also, from the definition of W_{v-k}, we have $W_{v-k} = \sum_{h=0}^{B} \mathbf{1}_h$. Therefore, $E[W_{v-k}] = \sum_{h=0}^{B} E[\mathbf{1}_h] = \sum_{h=0}^{B} \psi_h(k, v) = (B + 1)/(B - k + 1)$. □

E Proof of Theorem 4.1

Proof. Substituting (13) into (12) yields,

$$p_n(v) = \begin{cases} 1/B^n, & n \geq 1, v = 0 \quad (i) \\ \sum_{k=0}^{v} p_{n-1}(k)(B + 1)/(B(B - k + 1)), & n \geq 1, B > v > 0 \quad (ii) \end{cases} \tag{32}$$

$p_n(B) = 1 - \sum_{l=0}^{B-1} p_n(l), n \geq 1$. The base cases are $p_0(0) = 1$ and $p_0(v) = 0$ if $v \neq 0$. In case (ii), $p_n(v) = p_{n-1}(v)(B + 1)/(B(B - v + 1)) + \sum_{k=0}^{v-1} p_{n-1}(k)(B + 1)/(B(B - k + 1)) = p_{n-1}(v)(B + 1)/(B(B - v + 1)) + p_n(v - 1)$ which yields (15).

□

F Derivation of (18)

Proof. We now show that (18) follows from (17). The case $v = B$ follows trivially. For $v < B$, we show that (18) satisfies (17), i.e., $p_n(v) - p_{n-1}(v)(1/(B - v + 1)) = p_n(v - 1)$,

$$p_n(v) - p_{n-1}(v)(1/(B - v + 1)) = \binom{B}{v} \sum_{l=0}^{v} \binom{v}{l} (-1)^l (B - v + l + 1)^{-n} - \frac{\binom{B}{v} \sum_{l=0}^{v} \binom{v}{l} (-1)^l (B - v + l + 1)^{-n+1}}{B - v + 1} \tag{33}$$

$$= \binom{B}{v} \sum_{l=0}^{v} \binom{v}{l} (-1)^{l+1} (B - v + l + 1)^{-n} \frac{l}{B - v + 1} = \binom{B}{v - 1} \sum_{l=0}^{v-1} \binom{v - 1}{l} (-1)^l (B - v + l + 2)^{-n} = p_n(v - 1) \tag{34}$$

□

G Expression of $p_n(v)$ When $\gamma = \infty$

If $\gamma = \infty$, the closed-form expression for $p_n(v)$ is $p_n(v) = \binom{B}{v} \left(\frac{1+B}{B}\right)^n \sum_{l=0}^{v} \binom{v}{l} (-1)^l (B - v + l + 1)^{-n}$, $n \geq 1, B > v \geq 0$, and $p_n(B) = 1 - \sum_{v=0}^{B-1} p_n(v)$.

Proof. We now show that $p_n(v)$ follows from (15). The cases $v = B$ and $v = 0$ follow trivially. For $0 < v < B$, we show that $p_n(v) - p_{n-1}(v)((B + 1)/(B(B - v + 1))) = p_n(v - 1)$,

$$p_n(v) - p_{n-1}(v)((B + 1)/(B(B - v + 1))) = \binom{B}{v} \left(\frac{1 + B}{B}\right)^n \left[\sum_{l=0}^{v} \binom{v}{l} (-1)^l \left((B - v + l + 1)^{-n} - \frac{(B - v + l + 1)^{-n+1}}{B - v + 1}\right)\right]$$

$$= \binom{B}{v - 1} \left(\frac{1 + B}{B}\right)^n \sum_{l=0}^{v-1} \binom{v - 1}{l} (-1)^l (B - v + l + 2)^{-n} = p_n(v - 1)$$

□