Flash crowd in a file sharing system based on random encounters

I. Norros, B.J. Prabhu,and H. Reittu
VTT Technical Research Centre of Finland
P.O. Box 1000
Espoo, Finland
{ilkka.norros,ext-balakrishna.prabhu,hannu.reittu}@vtt.fi

ABSTRACT

BitTorrent revolutionized the technique of distributing a very large file to a very large number of recipients. The file is chopped into small chunks that the recipients can immediately upload further. In the original design, a "tracker" keeps certain centralized control over the chunk transfer process. This paper studies a BitTorrent-like "information diffusion" system that has a fully distributed and symmetric architecture. The peers join a Distributed Hash Table -based overlay network and contact each other randomly. This kind of designs have been implemented and analysed recently. A trackerless BitTorrent system has been introduced which can be regarded as one based on random encounters — the participating nodes contact each other at random and download missing chunks. On the analytical front, Massoulie and Vojnovic showed that a random encounter based system has surprisingly good performance without any chunk preference strategies, with the condition that each peer gets its first chunk from a sufficiently uniform distribution. In this paper, we focus on a scenario where this condition cannot be guaranteed, and show that a "rare chunk phenomenon" easily occurs, if both the encounters and the chunk selection are random. Classic urn models give some mathematical understanding of this phenomenon. We then discuss various techniques for alleviating the rare chunk problem and propose a simple distributed chunk selection policy that reduces the imbalance in the distribution of chunks within the network.

Categories and Subject Descriptors

C.4 [**Computer Systems Organization**]: Performance attributes; C.2.4 [**Distributed Systems**]: Distributed applications

General Terms

Algorithms, Performance

Inter-Perf'06, October 14, 2006, Pisa, Italy.

Keywords

Peer-to-peer, Information diffusion

1. INTRODUCTION

Peer-to-Peer (P2P) file sharing systems such as BitTorrent [1] have gained in popularity by making content distribution accessible to virtually every Internet user. A node which wants to distribute a file (also called a seed node) fragments the file into chunks and uploads these chunks to interested peers. The peers interested in downloading the file also help in distributing this file by uploading the chunks to other peers. Thus, nodes participating in such systems act as both clients and servers, thereby contributing to the service capacity of the system. A salient feature of such networks is the scalability of the service capacity with offered load [2], resulting in efficient utilization of network resources. Thus, even a seed node with a relatively small available upload bandwidth is able to distribute a file to a large number of nodes in a reasonable amount of time.

The original BitTorrent relies on a web-server called *tracker* to facilitate the exchange of chunks. The tracker maintains the IP addresses of the nodes currently downloading a particular file, and the chunks that each of these nodes has. A new node interested in downloading a file contacts the tracker, and obtains a list of potential servers (these can be seeds, who possess the whole file, but mostly they are peer downloaders). The first few chunks to be downloaded by a new node are selected at random. However, the latter chunks are downloaded using a *rarest first* chunk selection policy in order to maintain a roughly uniform distribution of the number of copies of each chunk in the system. Several simulation studies have shown that the *rarest first* policy outperforms other policies such as *random* selection [3], [4], and hence could be one of the main reasons for the good performance of BitTorrent.

One of the various topics of the Finnish research project PAN-NET (2004-2005) was to design an experimental prototype of a BitTorrent-counterpart with completely distributed architecture. This system is based on random encounters and uses Chord as the underlying structure where random contacts can be realised. The problems studied in this paper were largely prompted by the PAN-NET architecture, described in Section 2.1 below.

Massoulie and Vojnovic [5] had studied a quite similar system as an abstract model and obtained remarkable results by considering a limiting differential system in a linear scaling of the initial state of the system. In particular, they could conclude that a random encounter based file sharing system can be essentially insensitive to load balancing strategies like the "rarest first" principle implemented in BitTorrent.

This paper is motivated by the following question: what is the performance of this system in a *flash crowd* scenario, if the first chunk cannot be given by the seed? We argue that the initial state in a system without load balancing may not be as well behaved as previously assumed. This discrepancy in the initial state can lead to situations where a large proportion of the initial population is searching for one particular chunk, and is forced to download it from the seed node. The imbalance in the system arises due to random selection of the peers — a chunk with larger number of copies in the system will be sampled at a higher rate, and thus replicates much faster.

We assume that a seed node is interested in distributing a file using a BitTorrent like algorithm, and that a certain number of users enter the system simultaneously at time zero. Upon completing the file download, the node leaves the system promptly. We shall be interested in the time taken to empty the system, i.e., the time till every node from the initial population has a copy of the file. Intuitively, the faster an algorithm is able to distribute the file, the more efficient it is in utilizing the available bandwidth in the network. It will also be seen that a more balanced chunk distribution in the system leads to a faster replication of the file. A balanced chunk distribution also adds to the robustness of the file distribution system in case of a failure of the seed node.

The main contributions in this paper are the following

- Through simulations, we show that the *random* chunk selection policy can result in at least one chunk becoming rare in the system. Thus, a failure of the seed node can result in a large number of unfinished downloaders. We also relate the replication of chunks to the growth of the number of balls in a Pólya's urn model. Through this analogy, we show that the state of the system (i.e., number of copies of each chunk in the system) can have a large variance. Hence, the state of the system may not be as well behaved as was assumed to show insensitivity to load balancing strategies.

- We provide a simple and distributed load balancing algorithm which requires no state information, and yet outperforms the *random* selection policy. Using simulations we show that a large proportion of the initial population finishes downloading the file around the same time. Thus, the use of this particular strategy reduces the imbalance in the number of copies of the chunks in the system at any given time.

1.1 Related Work

Performance of file sharing systems has been studied using both simulations and mathematical models. The large number of parameters and complicated algorithms (e.g., chunk selection policies, upload and download bandwidths, etc.) make accurate mathematical models intractable. Hence, articles such as [5], [6] and [7] have modelled the system in limiting regimes. Our system model is mainly based on the model in [5]. As in [5], we are primarily interested in the evolution of the number of replicas of the chunks in the system. However, unlike their closed system model in which they study the leftovers in the system starting from certain initial conditions, we study the system starting from time zero. We model the initial growth in the number of replicas of the chunks in the system using Pólya's urn model. The models in [6] and [7] mainly deal with systems in which the entire file is considered as a single chunk, and the steady state performance measures (e.g., time to download the file) are then studied through fluid models. In this work, we do not consider the effect of upload/ download bandwidth and assume as in [5] that chunk are downloaded instantaneously. A recent article [8] gives the optimal number of rounds required to distribute M chunks to N nodes. They also give a centralized algorithm to optimally distribute the chunks. Unlike [8] which focuses on the optimal time to distribute a file, the present article studies the chunk population dynamics in the system.

Simulation studies such as [3] and [4] allow to study the effect of various parameters (e.g., the chunk selection policy, the peer selection policy, and heterogenous upload and download bandwidths) on the performance of the system. Although the system that we simulate is not as comprehensive as the the systems in [3] and [4], and is based on simulating the model in [5], we note that our observation regarding the existence of rare chunks in random chunk selection policy is consistent with that in [3].

1.2 Organisation of the paper

In section 2 we first describe the PAN-NET file distribution system. We then give the system model (based on the PAN-NET system) and the various performance measures of interest. In section 3, we relate the replication of chunks with the growth of balls in Pólya's urn model, and show how the rare chunk problem occurs. In section 4 we propose distributed chunk selection policy and evaluate the its performance. Finally, in section 5 we give the conclusions.

2. SYSTEM DEFINITION

2.1 The PAN-NET file distribution system

The scenario of the PAN-NET system is the following. One peer, called the seed, has the complete file, chopped into chunks, and stays in the system as long as it wants to distribute the file. All peers, including the seed, run the same software. A peer interested in the file contacts the seed (or another known member of the distribution process), who gives it access to a Chord [9] network, initiated by the seed specially for this purpose. The new peer obtains the possibility to contact randomly selected peers, but remains invisible for searches (remains a "parasite") until it has downloaded at least one chunk.

The procedure of each random encounter between two peers A and B runs as follows. Peer A, the initiator, sends its chunk-bitmap (0 for missing, 1 for present chunks) and an optional list of specially wanted chunks. A transfer rate (interpreted as upper bound) can be specified as well. Peer B answers with its own bitmap and, optionally, its list of "recommended" chunks and its opinion on the maximal transfer rate. Peer A then sends its decision on the chunk to be downloaded, and B starts to upload that chunk. The scheme was made this complicated to allow experimentation with various decision algorithms.

After a successful download, peer A can either repeat the above procedure to get the next chunk, or select another random peer. The PAN-NET client can run several independent downloading processes in parallel.

Selecting a random peer

Finding a random (in the sense of *uniform* distribution) peer from a network is a non-trivial problem that has not been considered much in literature (see, however, [10]). In Chord, for example, it is not sufficient to issue `find_successor(`x`)` with a random argument x, because the nodes are not equidistant on the Chord ring. We propose the following algorithm, which is based on the assumption that all peers perform random peer *look-ups* uniformly. (This is not strictly true even in the PAN-NET scenario, since a peer's access rate affects

the frequency of its look-ups). Each peer, say A, maintains a storage for one address, say r_A. Initially it stores its own address. Now, if B wants to find a random peer, it issues `find_successor(`x`)` with random x and gets the address of, say, C. Peer B then contacts C and receives the content of r_C as the desired "roughly random peer address", whereas C stores the address of B in r_C to wait for the next request.

2.2 The abstract system considered in this paper

Consider a node (henceforth, called the seed node) on the Internet which wants to distribute a file. The seed node fragments this file into C chunks. At time zero, N nodes, who wish to download this file, enter the system. This setting is motivated by the *flash crowd* phenomenon wherein a large number of nodes get interested in a particular file within a relatively short period of time and try to download it. We further assume:

(i) some mechanism for contacting a random node is implemented;

(ii) the seed node is present throughout the lifetime of the system as (just) one of the peers;

(iii) a node with no chunks to upload is "invisible", i.e., no nodes (except the seed node) can contact it; it becomes a member of randomly selectable nodes once it has downloaded one chunk;

(iv) every node in the system maintains a timer which expires after exponentially distributed time intervals; on the expiry of its timer, the node initiates a random encounter;

(v) within each encounter, the initiator downloads at most one chunk from its counterpart;

(vi) *chunk selection policy:* we consider several algorithms for selecting the chunk that is transferred in an encounter;

(vii) the download time is zero;

(viii) *exit policy:* the nodes (other than seed) leave the system as soon as they obtain all the chunks.

The modelling assumptions (iv) and (vii) are adopted from [5]. The scenario of [5] differs from ours in two respects:

(i) their seed gives each node one chunk (or more) when it enters the system, but remains invisible for further encounters;

(ii) in each subsequent encounter, the chunk is randomly selected among those are missing from the initiator.

The exit policy assumption (viii) is a worst case scenario for the seed node.

Upload policy of the seed node: We assume that the seed node uploads the chunks in a round robin manner. This policy leads to more a uniform chunk distribution in the system as compared to the purely random chunk uploading policy.

2.3 Performance measures

In order to distribute C chunks to N nodes, there need to be at least $C \cdot N$ encounters. This would be the case if every node knew where to download its missing chunks from, i.e., if every node possessed the global state information. In our system, the nodes have knowledge of only their own state.

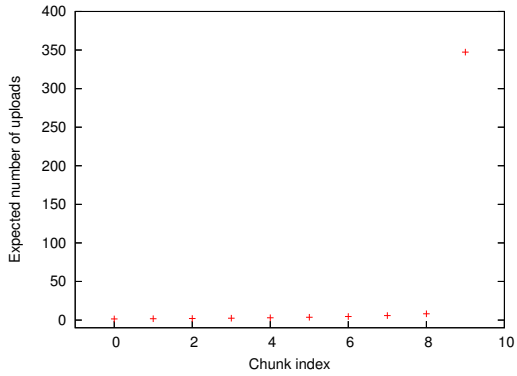

Figure 1: Expected number uploads of chunks by the seed node. *Random chunk* **policy.** $N = 1000$. $C = 10$.

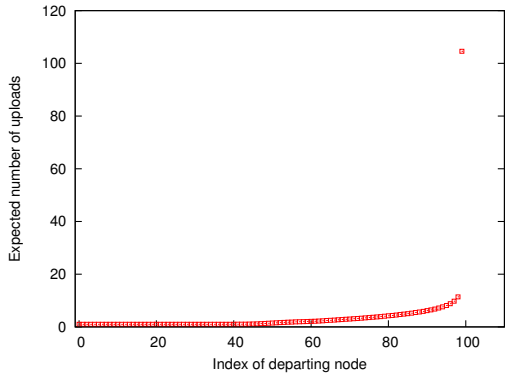

Figure 2: Expected number uploads of chunks by the seed node. *Random chunk policy.* $N = 1000$. $C = 100$.

Hence, there will be encounters which will not result in a downloading of a chunk. The first performance measure we study is the expected number of encounters until the last node finishes downloading the file and leaves the system. This measure gives us the *expected time to empty the system* and quantifies, in some sense, the inefficiency due to the lack of global information.

The second performance measure we study is the *expected number of uploads* of each chunk by the seed node. A high variance in the number of uploads (i.e., some chunks are uploaded very few times compared to others) would indicate some sort of imbalance in the chunk distribution in the system. If some chunks are uploaded a significantly higher number of times by the seed node than others, then we could conclude that some chunks are rare in the system — see Section 3.

3. THE RARE CHUNK PHENOMENON

3.1 An empirical look at the phenomenon

Having defined the system and its performance measures, we begin by simulating the system for a particular set of parameters. Let $N = 1000$ be the number of nodes that enter the system at time zero. In figures 1 and 2, we plot the expected number of uploads of a chunk by the seed node in ascending order of the number of uploads for $C = 10$ and $C = 100$, respectively.

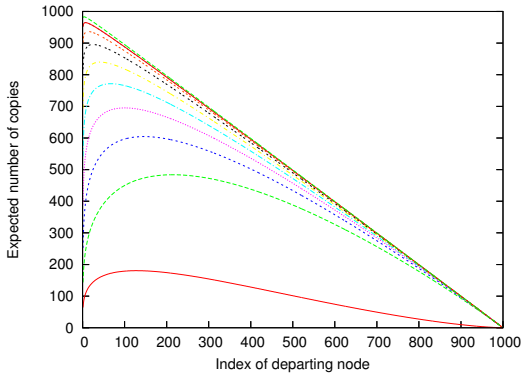

Figure 3: Expected number of copies of the chunks as seen by departing nodes. *Random chunk* **policy.** $N = 1000$. $C = 10$

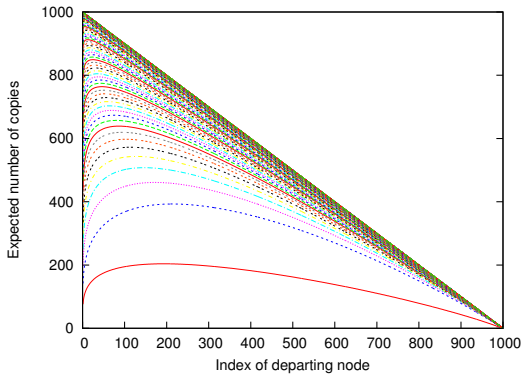

Figure 4: Expected number of copies of the chunks as seen by departing nodes. *Random chunk* **policy.** $N = 1000$. $C = 100$.

From these plots, we can observe that there exists one chunk which the seed node has to upload a disproportionate number of times compared to the other chunks. This suggests that there exists an imbalance in the number of replicas of the chunks in the system. This phenomenon becomes critical towards the end of the system lifetime when the nodes would need to download this rare chunk from the seed node. For $C = 10$ and $C = 100$, in figures 3 and 4, we plot the expected (ordered) number of replicas of chunks in the system as seen by the nth departing node. We observe that the number of replicas of the chunks which are distributed first tend to grow faster than those which are enter the system later. Moreover, we see that, after a while, the very last chunk becomes more and more rare, whereas the others tend to good balance with each other. A similar observation was also made in [3]. In order to gain a better understanding of the reasons behind this imbalance, we relate our system to Pólya's urn models.

3.2 Mathematical discussion: Pólya's urn models

Here we analyse some simplified models that clarify our point. Assume that the total size of the flash crowd is N and that each node, including the seed, have download and upload capacity c chunks per time unit. Each node that has downloaded at least one chunk is called *visible*. At time 0, the seed alone is visible. Assume that the whole flash crowd is connected such that each node is able to request downloads from a randomly selected visible node.

As long as no node leaves the system, the number V_t of visible nodes grows roughly according to the logistic model

$$\mathrm{d}V_t = c\left(1 - \frac{V_{t-}}{N}\right) V_{t-}\,\mathrm{d}t,$$

interpreted in the sense of point process dynamics, i.e., the right hand side presents the stochastic intensity of a point process [11]. Thus, at the beginning the overwhelming majority of uploads are directed to so far invisible nodes, and the visible nodes have typically a single chunk in possession.

A crude model for the start of chunk replication in a big flash crowd event is thus that each visible node that is not the seed has exactly one chunk. It turns out that it is important to understand these very first moments, since the proportions of chunks numbers fluctuates remarkably only for a short while after the start. This is a consequence of the analogy with Pólya's classic urn models.

The formation of this "initial population" can be studied as a discrete vector process $(M_n(1), \ldots, M_n(C))$, where $M_n(i)$ is the number of non-seed nodes whose only chunk is chunk i, at the time when the n'th upload has just happened. Denote the n'th uploaded chunk by I_n and

$$\mathcal{F}_n = \sigma\left\{M_k(i) :\ k \le n,\ i \in \{0, \ldots, C-1\}\right\}.$$

Assuming that the seed selects each uploadable chunk randomly, we have that

$$\begin{aligned}\mathbb{E}\left[M_n(i) \,|\, \mathcal{F}_{n-1}\right] &= M_{n-1}(i) + \mathbb{P}[I_n = i \,|\, \mathcal{F}_{n-1}] \\ &= M_{n-1}(i) + \frac{1}{Cn} + \frac{M_{n-1}}{n} \\ &= \frac{n+1}{n} M_{n-1}(i) + \frac{1}{Cn}.\end{aligned}$$

It follows that for each i, the sequence

$$\frac{M_n(i) + 1/C}{n+1}$$

is an $(\mathcal{F}_n)$-martingale. Bounded martingales converge with probability one, and at each phase the conditional expectation is that the chunk distribution, where the seed's chunks are "devalued" as $1/C$ each, stays at its present proportions. Thus, the early, random history of the copying process has a lasting effect.

Pólya's simplest urn model starts with two balls, one black and one white. At each step, a ball is picked at random, and a new ball is added that has the same color as the just picked one. With $C = 2$, the only difference with our setting is the seed. In the above Pólya model, the proportion of black balls is a martingale whose limit value is uniformly distributed over $[0, 1]$. To show how sensitive this models are to slight differences in early history, we note that when the seed is added, the uniform distribution gets replaced by the Arcsin law:

PROPOSITION 3.1. *Let* $C = 2$.

$$\begin{aligned}\mathbb{P}(M_n(1) = m_1) &= \frac{1}{\pi} \frac{\Gamma(m_1 + 1/2)}{\Gamma(m_1 + 1)} \frac{\Gamma(n - m_1 + 1/2)}{\Gamma(n - m_1 + 1)} \\ &\sim \frac{1}{\pi} \frac{1}{\sqrt{m_1(n - m_1)}}.\end{aligned}$$

PROOF. The conditional probability that chunk 1 is copied at time $n+1$ given that there have been m_1 such encounters so far is

$$\frac{1}{2} \cdot \frac{1}{n+1} + \frac{m_1}{n+1} = \frac{2m_1 + 1}{2(n+1)},$$

where the first term at left corresponds to encountering the seed and obtaining 1 from it and the next corresponds to encountering a node holding chunk 1. This probability has the form of a Pólya urn model, in the sense that the probability of encountering m_1 times 1 in $n \geq m_1$ encounters does not depend on which order 1 and 0 is encountered. Therefore, the probability of such an event is the probability of encountering first chunk 1 m_1 times in row and then $m_0 \equiv n - m_1$ times chunk 0, multiplied by number of ways m_1 elements can be selected out of n. The last factor is the binomial coefficient $C_n^{m_1}$ and the first one is

$$\begin{aligned}
&\frac{(2\cdot 0+1)}{2\cdot 1}\frac{(2\cdot 1+1)}{2\cdot 2}\cdots\frac{(2(m_1-1)+1)}{2m_1}\cdot\\
&\frac{(2\cdot 0+1)}{2(m_1+1)}\frac{(2\cdot 1+1)}{2(m_1+2)}\cdots\frac{(2(m_0-1)+1)}{2(m_1+m_0)}\\
&=\frac{1\cdot 3\cdots(2m_1-1)\cdot 1\cdot 3\cdots(2m_0-1)}{2\cdot 4\cdots(2n)}\\
&=\frac{(2m_1-1)!!(2m_0-1)!!}{(2n)!!}.
\end{aligned}$$

Using the formulas $(2k-1)!! = \frac{2^k\Gamma(k+1/2)}{\sqrt{\pi}}$ and $(2k)!! = 2^k k!$, where Γ, is Euler's Γ-function, we find that the probability of uploading chunk 1 m_1 times out of n encounters is

$$\begin{aligned}
&C_n^{m_1}\frac{(2m_1-1)!!(2n-2m_1-1)!!}{(2n)!!}\\
=\;&\frac{n!2^{m_1+n-m_1}\Gamma(m_1+1/2)\Gamma(n-m_1+1/2)}{m_1!(n-m_1)!\pi 2^n n!}\\
=\;&\frac{1}{\pi}\frac{\Gamma(m_1+1/2)\Gamma(n-m_1+1/2)}{\Gamma(m_1+1)\Gamma(n-m_1+1)},
\end{aligned}$$

which is the claimed result. The asymptotic comes from Stirling's formula. □

REMARK 3.2. *The importance of the early history can be quantified in terms of information. Consider the simple Pólya model starting with two balls, and denote by X_n the proportion of 1-balls after step n. We noted that, $X_n \to X_\infty \sim U[0,1]$ as $n \to \infty$. However, the outcome of random variable X_∞ is "determined" by the few first encounters with good precision. The distribution of $M_n = M_n(1)$ has entropy $H(M_n) = \log(n+1)$ and tends to infinity as $n \to \infty$. Fix some integer $0 < f < \infty$ and consider the mutual information*

$$I(M_n; M_f) \equiv H(M_n) - H(M_n \mid M_f),$$

where $n > f$ and the conditional entropy is defined as

$$H(M_n \mid M_f) = -\sum_{m_f}\mathbb{P}(M_f = m_f)\cdot$$

$$\sum_{m_n}\mathbb{P}(M_n = m_n \mid M_f = m_f)\log\mathbb{P}(M_n = m_n \mid M_f = m_f),$$

and the summations are over all possible outcomes of M_n and M_f, respectively. The limit $I(M_n; M_f)$ as $n \to \infty$ is finite and tells the amount of information about X_∞ contained in observing the first f encounters. Since the outcome of M_∞ is a real number between 0 and 1, x bits of information fixes roughly x binary digits of the outcome. Notably these digits are the most significant, first x digits. This can be verified by dividing the interval $(0,1)$ into equal subsequent intervals of finite length $= 1/f$ and looking on events of X_∞ belonging in these segments, denoted as X_∞^f. Obviously $H(M_\infty^f) = \log f$, since M_∞^f has f outcomes that have equal probabilities. Now our claim, so far verified only by simulations, is that $I(X_\infty^f; M_f) \approx H(X_\infty^f)$. This result relates our simplified models to more realistic scenarios, since the crucial start of chunk propagation from the seed, would be similar in both cases, provided the number of peers is large. This result indicates a qualitative scenario: the last missing two chunks exist in very different in size populations and finally in the end game mode there is a large number of peers missing the same chunk.

Let us continue with the case $C = 2$, and assume that no new nodes join any more, and, neglecting the seed in this phase, the population now consists of M_i nodes possessing chunk i, $i = 0, 1$. Assume that each node makes random encounters with unit frequency. Encounters with the same chunk are useless, so a good balance is clearly beneficial. The dynamics of the "end game" are roughly governed by the deterministic system

$$\begin{aligned}
x' &= -\frac{y}{x+y}x\\
y' &= -\frac{x}{x+y}x,
\end{aligned}$$

where x and y are the current sizes of the populations of 0-nodes and 1-nodes, respectively. When the variables are changed to $m = x + y$, $r = x/(x+y)$, it turns out that m'/m can be eliminated from the system, yielding for r the equation

$$r' = (2r-1)r(1-r).$$

The qualitative behavior of $r(t)$ is obvious: $1/2$ is an unstable equilibrium, outside which $r(t)$ converges to 0 or 1. We also have explicit solution

$$r(t) = \frac{1}{2}\left(1 \pm \sqrt{1 - \frac{4}{4 + c_0 e^t}}\right), \quad \text{where } c_0 = \frac{(2r(0)-1)^2}{r(0)(1-r(0))}.$$

The total population $m(t)$ is then obtained as

$$\begin{aligned}
m(t) &= m(0)\exp\left(-2\int_0^t r(s)(1-r(s))\,\mathrm{d}s\right)\\
&= m(0)\left(\frac{4e^{-t}+c_0}{4+c_0}\right)^{1/2}.
\end{aligned}$$

Thus, with $r(0) = 1/2$ the population vanishes exponentially as $m(t) = m(0)\exp(-t/2)$, whereas a deviation from equilibrium results in slower decay. In fact, this mode is qualitatively unavoidable because the equilibrium in unstable. Moreover, there remains a homogeneous left-over population of size $m(0)|2r(0)-1|$. In the discrete reality, a left-over homogeneous population has to download the remaining chunk from the seed. (The mathematical analysis above is in fact a special case of Theorem 5 in [5].)

The above discussion suggests that entirely randomised chunk selection results in an unpredictable disbalance in chunk propagation. However, in the case of only two chunks, consider another randomised policy: every time step a chunk holder is encountered randomly and uniformly, if 1 (resp. 0) is encountered, the new node gets chunk 0 (resp. 1) is added. Is such a policy implementable? In principle, yes: an invisible node encounters a random visible one, but, instead of downloading that node's chunk, proceeds random encounters until it finds a node having the complementary chunk. This process results in a well balanced chunk propagation, with both chunk populations growing with the same pace. We have the following result:

PROPOSITION 3.3. *Let $C = 2$. With the inverted chunk selection described above, denote by $M_n = M_n(1)$ the amount of nodes possessing chunk 1 after the n'th upload. Then, $M_n/n \to 1/2$ almost everywhere and in $\mathcal{L}_1$-norm.*

PROOF. We use the following martingale argument of Mucci, [12]. Write

$$\begin{aligned}\mathbb{E}[M_n \mid \mathcal{F}_{n-1}] &= M_{n-1} + \mathbb{P}[I_n = 1 \mid \mathcal{F}_{n-1}] \\ &= M_{n-1} + \left(1 - \frac{M_{n-1} + 1/2}{n}\right) \\ &= \frac{n-1}{n} M_{n-1} + \frac{n - 1/2}{n}.\end{aligned}$$

Now, $Y_n = M_n/n$ is bounded and satisfies $\mathbb{E}[Y_n \mid \mathcal{F}_{n-1}] - Y_{n-1} \to 0$ a.s. as $n \to \infty$. This means that Y_n is a so called martingale in the limit and uniformly integrable. According to Theorem 2 in [12], there exists a random variable $M_\infty \in \mathcal{L}_1$ such that $M_n \to M_\infty$ almost everywhere and in $\mathcal{L}_1$-norm.

Take any $\eta \in (0, 1/2)$ and assume for the contrary that $\mathbb{P}(Y_\infty < \eta) > 0$. Then there exists n_0 such that

$$\mathbb{P}(Y_n < (\eta + 1/2)/2 \ \forall n \geq n_0) > 0.$$

Denote by ν the stopping time $\nu = \inf\{n \geq n_0 : \ Y_n > 1/2\}$. Now

$$\begin{aligned}\mathbb{E}Y_{n\wedge\nu} &= \mathbb{E}\sum_{k=1}^{n\wedge\nu} \mathbb{E}[Y_k - Y_{k-1} \mid \mathcal{F}_{n-1}] + \mathbb{E}Y_0 \\ &= \mathbb{E}\sum_{k=1}^{n\wedge\nu} \frac{2k-1}{k^2}(1/2 - Y_{k-1}) + \mathbb{E}Y_0.\end{aligned}$$

The starting expression is bounded, but the last is converges to infinity, and we obtain a contradiction. The case $\{Y_\infty > 1/2\}$ is symmetric. □

4. DISTRIBUTED CHUNK SELECTION POLICIES

We can summarize the observations from the previous sections as follows.

(i) The imbalance in the number of replicas of chunks in the system can be traced to the first few moments which determine which chunks replicate faster than the others. It would be desirable to have a more balanced number of replicas of chunks at the start of the system as this would lead to a more predicatable number of replicas of chunks in the system at later instants.

(ii) It would be very desirable to have high diversity in last missing chunk, say, for each node it is independent of all other nodes and is uniformly random. This would make the seed node less critical to the downloading process.

Can this be achieved in some decentralized and probabilistic way? We now describe a chunk selection policy which attempts to address the above mentioned concerns, i.e., to ensure a better balance in the number of replicas of chunks during the first few moments and to ensure a high diversity in last missing chunk.

Deterministic last K chunks policy. *Every node selects $K < C$ chunk indices at random. These chunks are downloaded only after the remaining $C - K$ chunks have been downloaded.*

Each node selects only the indices of the K chunks it will download towards the end. The order in which these K chunks will be downloaded is arbitrary and is not decided beforehand.

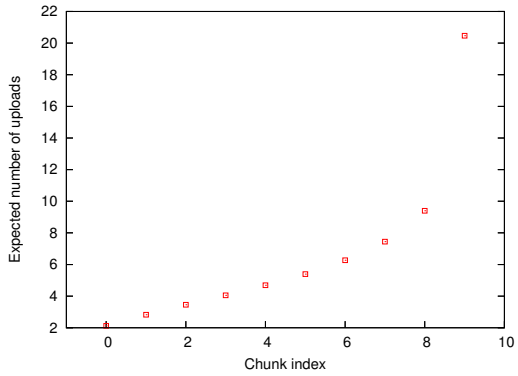

Figure 5: Expected number uploads of chunks by the seed node. *Deterministic last chunk* **policy.** $N = 1000$. $C = 10$. $K = 1$.

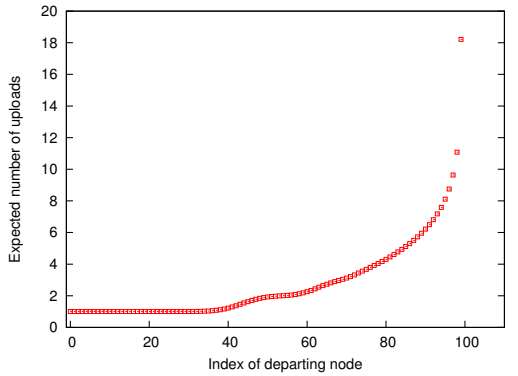

Figure 6: Expected number uploads of chunks by the seed node. *Deterministic last chunk* **policy.** $N = 1000$. $C = 100$. $K = 1$.

REMARK 4.1. *This policy can also be thought of as deterministic first $C - K$ chunks policy.*

For $K = 1, 2$, we compare the performance measures of this policy with those of the random selection policy. The initial number of nodes in the system is same as before (i.e., $N = 1000$). Figures 5, 6, 7, and 8 show the number of expected (ordered) number of uploads by the seed node for different values of C and K. On comparing these with figures 1 and 2 we see that the maximum uploads by the seed node has reduced compared with the random selection policy. Compared with $K = 1$, the policy with $K = 2$ reduces further the load on the seed node. By rejecting some chunks that may be on offer in the initial stages of downloading, the policy delays the departure of the first few nodes from the system. Thus, there are more uploaders in the system are present in the system for longer duration, thereby reducing the load on the seed.

The expected departure instants (in number of encounters) for the nodes is shown in figures 9 and 10. Using this policy, the first few nodes leave the system later compared to the random selection policy. However, this initial delay proves beneficial at later instants as more nodes are able to exit the system earlier than in the the random selection policy. Compared with $K = 1$, although the policy with $K = 2$ decreases the expected number of uploads by the seed node,

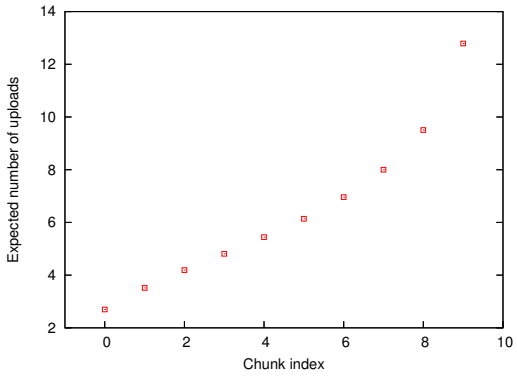

Figure 7: Expected number uploads of chunks by the seed node. *Deterministic last two chunks* **policy.** $N = 1000$. $C = 10$. $K = 2$.

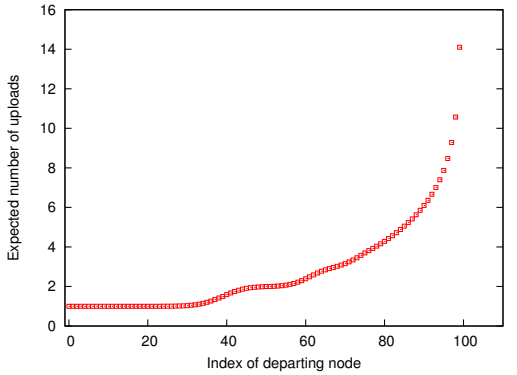

Figure 8: Expected number uploads of chunks by the seed node. *Deterministic last two chunks* **policy.** $N = 1000$. $C = 100$. $K = 2$.

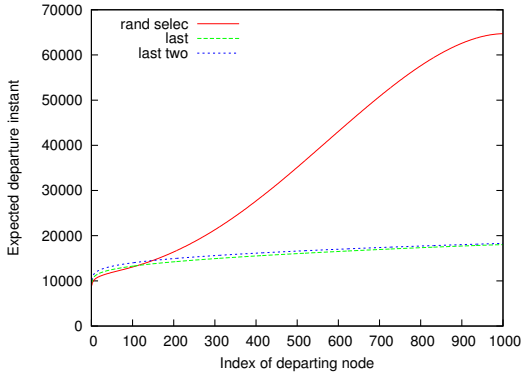

Figure 9: Expected departing instant of the ith departing node. $N = 1000$. $C = 10$.

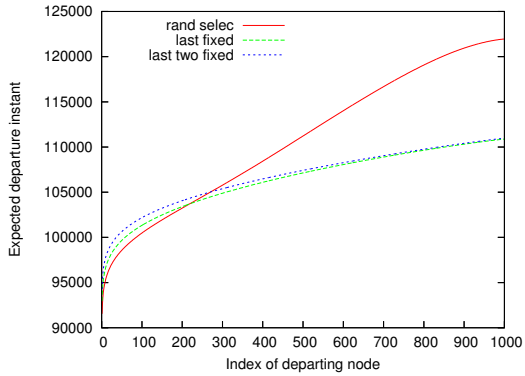

Figure 10: Expected departing instant of the ith departing node. $N = 1000$. $C = 100$.

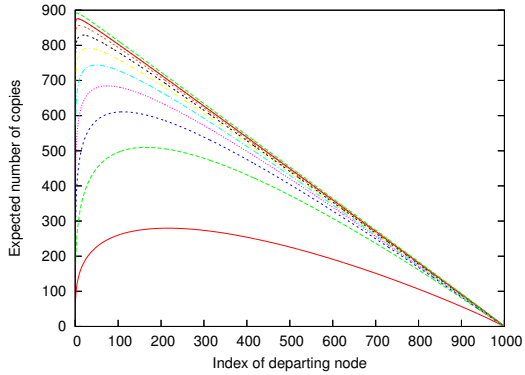

Figure 11: Expected number of copies of the chunks as seen by departing nodes. *Deterministic last chunk* **policy.** $N = 1000$. $C = 10$. $K = 1$.

it has the drawback of increasing the expected exit times of the nodes from the system as is shown. Since the last two chunks are now fixed, a node will be forced to reject them even if they have been offered for downloading. This increases the number of unproductive encounters, and adds to the time to exit from the system. There appears to be a trade-off between the load on the seed node and the time to empty the system.

For $C = 10$, figures 11 and 12 show the expected (ordered) number of copies of chunks as seen by the ith departing node. The total variation in the expected number of copies has reduced compared with the random selection policy. Less variation in the number of copies of different chunks would mean that more nodes will be able to finish downloading a file if the seed node were to leave the system for some reason. The corresponding values for $C = 100$ are shown in figures 13 and 14.

In order to get an approximate idea of what a desirable policy would be like, we now simulate two policies which require more resources (in terms of exchange of information or the uploading bandwidth of the server) compared with the policies that we just described. The first policy we consider is the *rarest first* policy. In the simulations, we assume that each node has global information of the number of copies of each chunk in the system. This is a best-case scenario for this policy. In practice, nodes may not have accurate

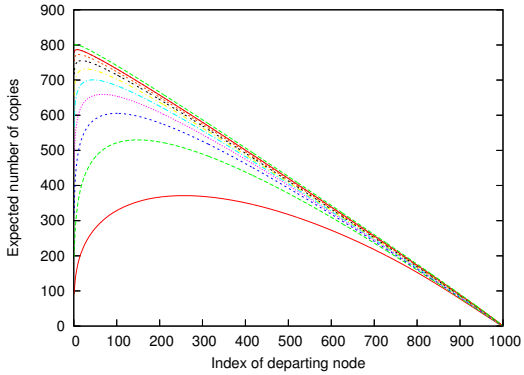

Figure 12: Expected number of copies of the chunks as seen by departing nodes. *Deterministic last two chunks* **policy.** $N = 1000$. $C = 10$. $K = 2$.

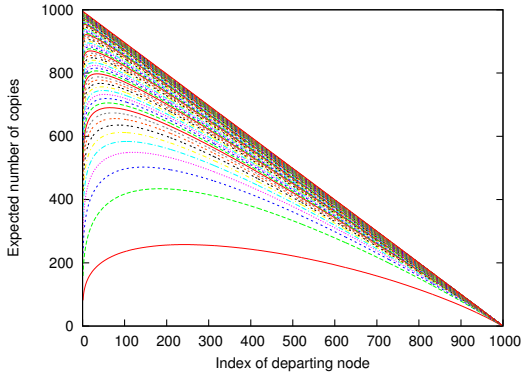

Figure 13: Expected number of copies of the chunks as seen by departing nodes. *Deterministic last chunk* **policy.** $N = 1000$. $C = 100$. $K = 1$.

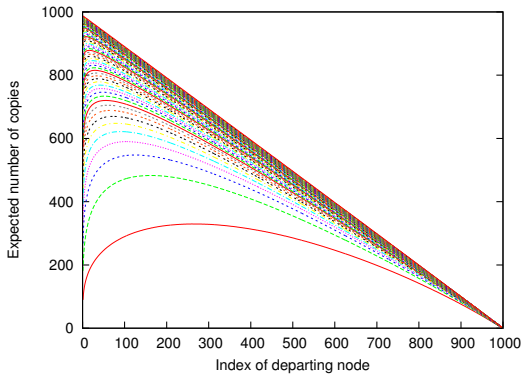

Figure 14: Expected number of copies of the chunks as seen by departing nodes. *Deterministic last two chunks* **policy.** $N = 1000$. $C = 100$. $K = 2$.

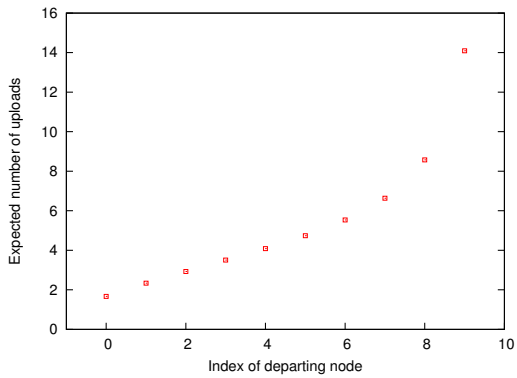

Figure 15: Expected number uploads of chunks by the seed node. *Rarest first* **policy.** $N = 1000$. $C = 10$.

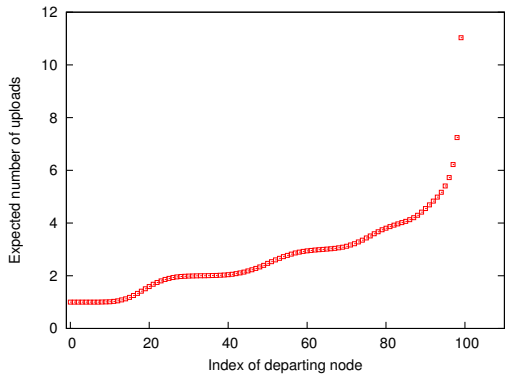

Figure 16: Expected number uploads of chunks by the seed node. *Rarest first* **policy.** $N = 1000$. $C = 100$.

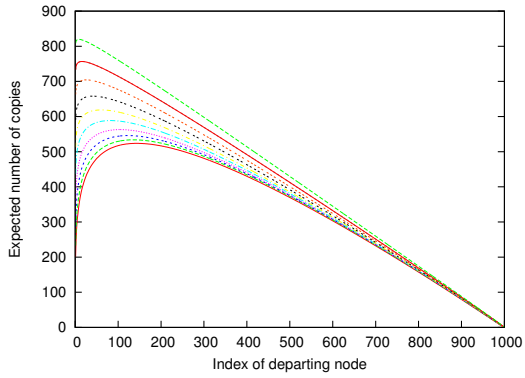

Figure 17: Expected number of copies of the chunks as seen by departing nodes. *Rarest first* **policy.** $N = 1000$. $C = 10$

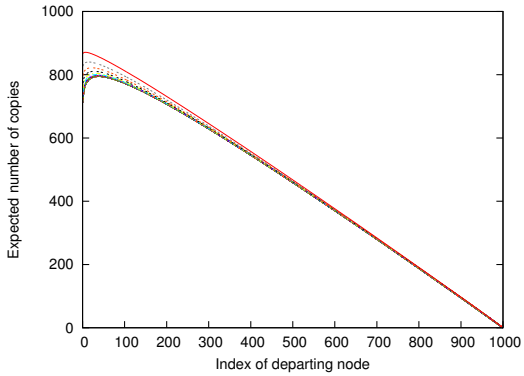

Figure 18: Expected number of copies of the chunks as seen by departing nodes. *Rarest first* **policy.** $N = 1000$. $C = 100$.

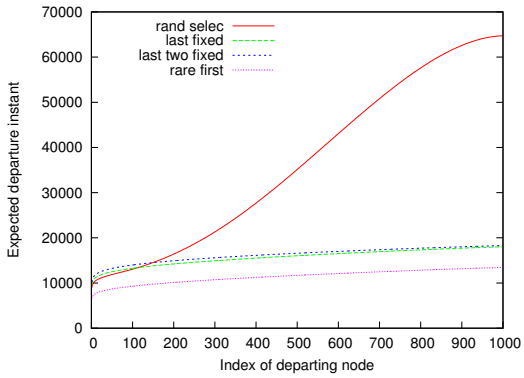

Figure 19: Expected departing instant of the ith departing node. $N = 1000$. $C = 10$.

information of the rarity of chunks, and the performance measures would depend on the algorithm used for estimating the rarity. In figures 15 and 16 we show the expected (ordered) number of uploads by the seed, and in figures 17 and 18 we plot the expected (ordered) number of copies of chunks as seen by the ith departing node. For $C = 100$, the number of copies seen of each chunk as seen by a departing nodes has very small total variation as compared to that in the *deterministic last K chunks* policy suggesting that all the chunks are very well distributed throughout the lifetime of the system. In figures 19 and 20 we compare the expected departure instants of nodes in various policies. From these figures we can infer that the *rarest first* policy indeed performs better the *random selection* and the *deterministic last K chunks* policy. The second policy is inspired by the one used in [5]. In this policy, we assume that each node has a random chunk at the start of the system. From the analysis in [5], this policy can be expected to perform quite well. In figures 21 and 22, we plot the expected (ordered) number of upload of each chunk by the seed node. The performance for $C = 10$ is better than the rarest first policy. However, for $C = 100$ the two policies have comparable performance. In figures 23 and 24, we plot the we plot the expected (ordered) number of copies of chunks as seen by the ith departing node. For $C = 10$, the maximum total variation in the expected number of copies seen by departing nodes is lesser compared to the *deterministic last K*

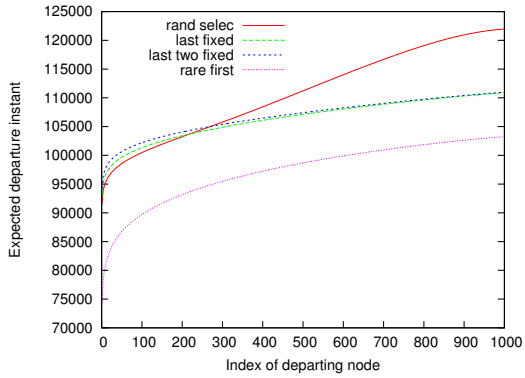

Figure 20: Expected departing instant of the ith departing node. $N = 1000$. $C = 100$.

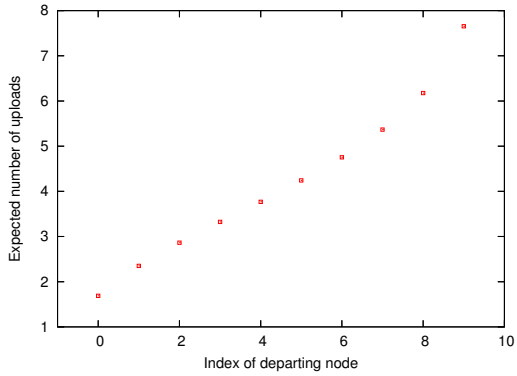

Figure 21: Expected number uploads of chunks by the seed node. *Initial piece from server* **policy.** $N = 1000$. $C = 10$.

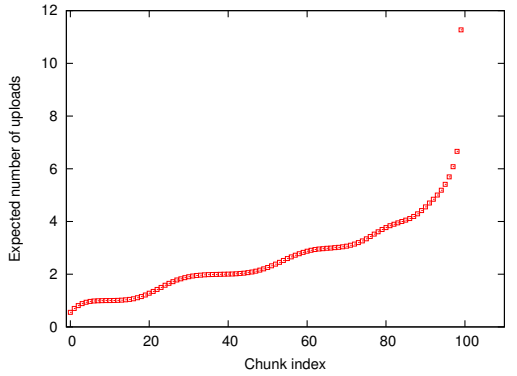

Figure 22: Expected number uploads of chunks by the seed node. *Initial piece from server* **policy.** $N = 1000$. $C = 100$.

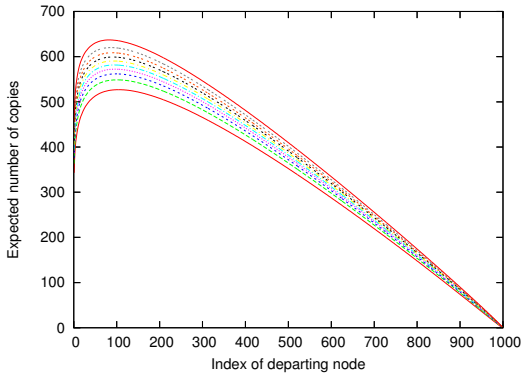

Figure 23: Expected number of copies of the chunks as seen by departing nodes. *Initial piece from server* **policy.** $N = 1000$**.** $C = 10$

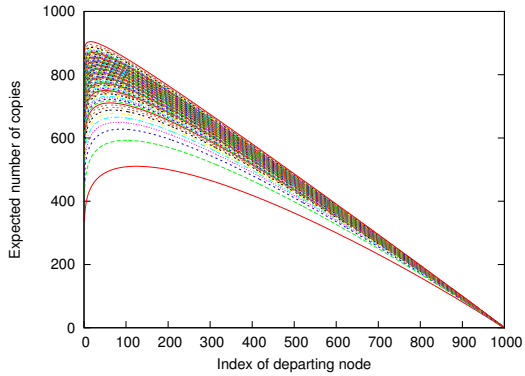

Figure 24: Expected number of copies of the chunks as seen by departing nodes. *Initial piece from server* **policy.** $N = 1000$**.** $C = 100$**.**

chunks policy and the *rarest first* policy. This implies that the chunks are quite evenly distributed in the system at all time instants and that many nodes can be expected to finish downloading the file even if the seed node leaves the system towards the end. However, this kind of policy may be impractical in a flash crowd scenario as all the peers will query the seed simultaneously and overload it.

5. CONCLUSIONS

In this paper, we studied the behaviour of *random chunk* selection policy for a random encounter based file distribution system during flash crowds. The *random chunk* policy leads to at least one chunk becoming rare in the system. These rare chunks need to be downloaded from the seed node thereby making the presence of the seed node critical to the completion of file downloads by peers. An insight into the origins of the rare pieces was gained by an analogy with Pólya urn models. Learning from these insights, we proposed the *deterministic last K chunks* policy to alleviate the rare chunk phenomenon. This is a completely *distributed* chunk selection algorithm which uses the same information as the *random chunk* selection policy. Unlike the *rarest first* policy, the proposed algorithm does not require global information on the number of copies of each chunk in the system. Through simulations we observed that the proposed policy performs well for up to a hundred chunks. The policy reduces the load on the seed node by reducing the rarity of the rare chunk, and it also increases the efficiency of the system by reducing the overall time to empty the seed.

Acknowledgements. The fully distributed design considered in this paper was developed in the Finnish project PAN-NET, whose other participants were Prof. Jorma Virtamo's group at Helsinki University of Technology and the companies Nokia and ROMmon. Experimental client software, written by Kari Keinänen at VTT, is available at `http://opensource.tte.erve.vtt.fi/pannet`. The research of B.J. Prabhu is supported by a fellowship granted by the European Research Consortium for Informatics and Mathematics (ERCIM).

6. REFERENCES

[1] "BitTorrent." [Online]. Available: http://www.bittorrent.com

[2] X. Yang and G. de Veciana, "Service Capacity in Peer-to-Peer Networks," in *Proc. IEEE INFOCOM*, Hong Kong, 2004.

[3] P. Felber and E. Biersack, *Self-star Properties in Complex Information Systems: Conceptual and Practical Foundations*, ser. Lecture Notes in Computer Science. Springer Berlin, 2005, vol. 3460, ch. Cooperative Content Distribution: Scalability through Self-Organization.

[4] A. R. Bharambe, C. Herley, and V. N. Padmanabhan, "Analyzing and Improving a BitTorrent Network's Performance Mechanisms," in *Proc. IEEE INFOCOM*, Barcelona, 2006.

[5] L. Massoulie and M. Vojnovic, "Coupon Replication Systems," in *Proc. ACM SIGMETRICS*, Banff, Canada, 2005.

[6] D. Qiu and R. Srikant, "Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks," in *Proc. ACM Sigcomm*, Portland, OR, 2004.

[7] F. Clévenot-Perronnin, P. Nain, and K. W. Ross, "Multiclass P2P Networks: Static Resource Allocation for Service Differentiation and Bandwidth Diversity," *Performance Evaluation*, vol. 62, no. 1-4, pp. 32–49, October 2005.

[8] J. Mundinger, R. Weber, and G. Weiss, "Analysis of peer-to-peer file dissemination," *To appear in Performance Evaluation Review, Special Issue on MAMA 2006.*

[9] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 17–32, February 2003.

[10] V. Vishnumurthy and P. Francis, "On Heterogeneous Overlay Construction and Random Node Selection in P2P and Overlay Networks," in *Proc. of IEEE INFOCOM*, Barcelona, 2006.

[11] P. Bremaud, *Point Processes and Queues:Martingale Dynamics.* Springer Verlag, 1981.

[12] A. Mucci, "Limits for martingale-like sequences," *Pacific Journal of Mathematics*, vol. 48, no. 1, 1973.