On Uncoordinated File Distribution with Non-altruistic Downloaders

Ilkka Norros[1], Balakrishna Prabhu[2], and Hannu Reittu[1]

[1] VTT Technical Research Centre of Finland, P.O. Box 1000, 02044 VTT, Finland
firstname.lastname@vtt.fi
[2] CWI, Postbus 94079, 1090 GB Amsterdam, The Netherlands
bjprabhu@gmail.com

Abstract. We consider a BitTorrent-like file sharing system, where the peers interested in downloading a large file join an overlay network. The seed node possessing the file stays in the system, whereas all other peers are non-altruistic in the sense that they leave the system as soon as they have downloaded the whole file. We consider a flash crowd scenario, where the peers join the overlay simultaneously. We show that the chunk selection algorithm is critical, propose an analytic approach to the process, and find that the encounters can be restricted to neighbours in a Chord overlay without losing much in performance.

1 Introduction

BitTorrent [1] revolutionized the technique of distributing a large file to a large number of recipients. The file is chopped into small chunks that the recipients can immediately upload further. 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. In the original BitTorrent, a "tracker" keeps certain centralized control over the chunk transfer process. This paper studies file distribution systems with fully distributed and symmetric architecture.

The Finnish research project PAN-NET (2004-2005) designed an experimental prototype of a BitTorrent-counterpart with completely distributed architecture. This system is based on random encounters, i.e., the peers contact each other randomly and transfer one (or several) chunk(s) if available. The PAN-NET prototype uses Chord [3] as the underlying structure where random contacts can be realised.

Massoulie and Vojnovic [4] studied a 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 principle to transfer always the rarest available chunk first, which is implemented in BitTorrent. This is interesting, because several simulation studies have shown that the *rarest first* policy outperforms other policies such as *random* selection [5], [6], and hence could be one of the main reasons for the good performance of BitTorrent. The above results are not contradictory, because the insensitivity conclusion of [4] presupposes that the system has already reached a sufficiently balanced state. In particular, they show that a sufficient condition for the preservation of a balanced chunk distribution is that the nodes join the overlay with a uniformly distributed first chunk in possession.

With this background, we focused in a recent paper [7] on the so called *flash crowd* scenario, where all nodes except the seed join the file distribution overlay simultaneously and empty-handed. This is an extreme case of the so-called flash-crowd scenario in which a large number of peers try to simultaneously acquire a copy of a popular software. It is also a model for distributing a new file in an existing network, say, within a company. We assumed that the nodes are *non-altruistic* in the sense that they stay in the system only as long as they have not downloaded all the chunks. This is an interesting scenario both as a pessimistic extreme and as being complementary to steady-state-oriented scenarios.

The main findings of [7] were: (i) The *random* chunk selection policy easily results in one chunk (usually exactly one!) becoming rare in the system. (ii) The replication of chunks resembles the growth of the number of balls in Pólya's urn model. This analogy helps understanding how the early history of the random copying process has a long-lasting influence on the chunk distribution. (iii) We provided an extremely simple distributed load balancing algorithm, the "Deterministic last K chunks policy", which outperforms the *random* selection policy without using any state information.

The present paper continues the work started with [7] mainly in two respects. First, we draw into consideration the principles of the optimal chunk distribution algorithms (thanking Michela Meo and Marco Mellia for raising this question), despite the fact that they require complete coordination and are as such unusable in a distributed control scenario. Using the optimal solution as a heuristic, we experimented by replacing unrestricted random encounters by chunk downloads from Chord topology neighbours only and found encouraging results. Second, we provide new analytical insight into the random process of the system's big bang -like evolution.

Related Work

Performance of file sharing systems has been studied both by simulations and through mathematical models. The large number of parameters and complicated algorithms make accurate mathematical models intractable. Hence, articles such as [4], [8] and [9] have modelled the system in limiting regimes. Fundamental performance limits [10,11] in this connection are discussed in Section 2. The models in [8] and [9] 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. Simulation studies such as [5] and [6] 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. Our observation regarding the existence of rare chunks in random chunk selection policy is consistent with that in [5].

Organisation of the Paper
We start in Section 2 by analysing an optimal solution to the file distribution problem and identify four interesting features of it. In Section 3, we present the models of uncoordinated file distribution strategies that will be studied in the sequel. Section 4 proposes a novel analytic scheme to describe the evolution of the "chunk universe" in the "big bang" of the flash crowd scenario. In Section 5, we present and discuss the results of several simulation setups. Finally, we draw the conclusions and propose items for further work in Section 6.

2 About the Optimal File Distribution Speed

Although our focus is on completely distributed algorithms, it is interesting to note how fast a coordinated procedure can distribute C chunks of same size to N peers with identical and finite upload capacities. The answer is $C + \log_2 N$, with single chunk copying time as the time unit. In the case $N = 2^L$, this has been known for long [11], but a proof for general N was given only quite recently by Mundinger *et al.* [10].

Remark 2.1. *We assume that chunks must be downloaded completely before they can be copied further to other peers. If the chunks were infinitely small, the distribution time would be independent of N.*

Although the algorithm for general N is quite complicated, it is based on some simple but important observations, formulated as lemmas in [10]. The most important one is that

(i) one should always transmit chunks at full speed, one at the time.

Interestingly, the classical BitTorrent uses typically four simultaneous downloads. Maybe this number is motivated by the heterogeneity of the peers' transmission speeds, but we don't know rigorous arguments for this choice. For small chunk size, other delays like queueing delays in the network start to affect along with the transfer delay. Then it may become favorable to split bandwidth to reduce these delays by parallel transmission.

When $N = 2^L$ with integer L, optimal file distribution algorithms need only the hypercube topology for chunk tranfer. Here is one of the variants. Let the node IDs be $(\iota_1, \dots, \iota_L)$, $\iota_i \in \{0, 1\}$. Call two nodes ℓ-pairs if their IDs differ only in the ℓth coordinate. Now, the "copying machine" runs as follows:

Algorithm 2.2 *([11], see also [12].) Assume $C > L$.*

```
for $\ell = 1, \dots, L, 1, \dots, L, 1 \dots$ (cyclically) do
  if node $(0, \dots, 0)$ has less than $C$ chunks
    then node $(0, \dots, 0)$ downloads next chunk from seed
    **else** *the $\ell$-pair of node $(0, \ldots, 0)$ downloads chunk $C$ from seed*
  *every other node downloads its $\ell$-pair's highest-numbered chunk*
```

For any C and L with $C \geq L + 1$, this "machine" distributes all chunks to all nodes in $C + L$ cycles. We note additional interesting features of this algorithm:

(ii) each node needs only $\log_2 N$ neighbours with whom to exchange chunks
(iii) most of the transfers are bi-directional, chunks moving along the hypercube edges in both directions
(iv) no nodes (except the seed) need to be altruistic. (For this we required $C > L$ — see what happens if $C = 1$ and the peers are non-altruistic!)

The reason for the sufficiency of the hypercube topology is that the number of copies of each chunk can at most be doubled in each period. It is easy to see that copying cyclically to hypercube neighbours, once to each, distributes a single chunk to all nodes. With appropriate organization, this process can run for $\log_2 N$ chunks in parallel and even so that all nodes except one finish simultaneously.

It is clear that only property (i) is a useful rule in most designs: the faster the chunks are transferred, the faster proceeds the copying process. A stochastic variant of property (ii) will be studied in Section 5 by simulations. Although this Chord-based solution does not reach optimal performance, the results are encouraging. The usefulness of property (iii) is questionable in an uncoordinated setup, but we have not experimented on it yet. Finally, property (iv) is obviously not true in general, on the contrary — the presence of altruistic peers is certainly advantageous in fully distributed schemes like ours. On the other hand, it is useful to know that if the copying process is sufficiently well organized, altruism is not necessary for fast file distribution.

3 Fully Distributed File Sharing Strategies in Overlays

The basic scenario of this paper is the following. One peer, called the seed, possesses a file, chopped into chunks, and stays in the system as long as it wants to distribute the file. The seed maintains also an overlay network, being initially the network's only node. Each peer interested in downloading the file joins the overlay and obtains the possibility to contact randomly selected peers of the overlay. (An algorithm for finding almost uniformly random peers was given in [7].) The peers then initiate independently such "random encounters" and, if the counterpart possesses chunks which the initiator does not have, a chunk is downloaded by the initiator.

This scenario leaves many algorithms and their parameters to be specified:

(i) the overlay structure and its maintenance mechanism
(ii) the peer selection mechanism for encounters
(iii) the rule for selecting the chunk to be downloaded
(iv) the number of chunks tranferred in one encounter
(v) chunks can also be transferred in both directions within the same encounter

(vi) transfer speed
(vii) number of parallel down- and uploads involving one peer
(viii) distribution of the time a peer stays in the system after having downloaded the whole file.

We make comments on each item and explain the choices made in this paper.

About (i): The overlay structure used in our experimental prototype is Chord [3], which is a popular example overlay systems based on a Distributed Hash Table. In such networks, the nodes take random or hashed identifiers from a hash value space $I = \{0, 1, \ldots, 2^M\}$, and the routing and resource discovery mechanisms are based on these identifiers. In the case of Chord, the hash value space is interpreted as a ring (in mathematical modelling it could be represented as the unit circle S^1). For $x \in I$, let $\texttt{succ}(x)$ denote the node whose identifier is closest to x in clockwise direction. A node with identifier i maintains connections to nodes with identifiers $\texttt{succ}((i+2^k) \bmod 2^M)$, $k = 0, 1, \ldots, M-1$. These nodes are called the *fingers* of node i. With N nodes, a typical number of separate fingers of a node is $\log_2(N)$. The finger connections are the edges of the Chord graph.

About (ii): In our original setup, random peers were encountered uniformly over the whole population of $N+1$ peers. A novelty in this paper, motivated by the considerations of Section 2, is the restriction to encountering only the fingers of each node, and using cyclic instead of random selection among them.

About (iii): A good rule would be to choose the chunk whose copies are rarest in the whole population. This information is not available in a distributed algorithm. The strategies we compare in this paper are random choice and the "deterministic last chunk" policy, introduced in [7] (see Section 5.2).

About (iv), (v), (vi), (vii): We did not study variations of these parameter. On the contrary, we adopted in both analysis and simulations the abstract model of [4], where nodes make encounters according to independent Poisson processes with unit rate, and the download time is set to zero.

About (viii): Throughout this paper, we assume that the peers are non-altruistic, i.e., the last parameter is concentrated at zero. This is a natural pessimistic assumption. (We have studied the effect of altruism elsewhere and found it significant in the last phase of the process.)

4 Towards an Asymptotical Analytical Model

We wish to extend our previous results [7] on Pólya's urn models as simple models for the starting phase of the random chunk copying process. We take the case of two chunks, 0 and 1. In total there is a fixed number of $n+2$ nodes in the system. At the start, one random node has chunk 1 and another distinct random node has 0. The remaining n nodes do not have any chunks, they are the empty nodes. At each time step, a uniformly random ordered pair of nodes is selected and called a random encounter. The first node downloads a chunk that it does not have, if the second has it but does not have both chunks. The last condition means unaltruistic behavior. As soon as a node has both chunks it becomes uncooperative and does not share its file. For large n, this copying process is likely to start as a Pólya's urn: first there are two balls (0 and 1) in the urn, then, at each time step, one ball is selected uniformly randomly from the urn and the selected ball is returned to the urn along with its copy. At any moment of time the distribution of all possible outcomes is uniform. For a large time it stabilizes to some value that is still uniformly distributed. However, this model is a pure growth process. We would rather have a process that has a steady state, meaning that, eventually, no encounters can change the state of the system. This means, for instance, that one chunk has become extinct outside the nodes that have both chunks and all empty nodes have also disappeared. Another option is that both chunks have disappeared in a similar way, in which case some empty nodes could be present also.

Our idea is to assume n large and use first Pólya's model to cope with the starting phase of the process. Then, for considerable populations of chunks, a continuous model in the form of differential equations is probably plausible. Finally, we must fuse these approaches to describe the distribution of the final states as $n \to \infty$. So far, we have made only the first steps rigorously, but we have some conjectures how to complete this scenario. First, it can be shown that the probability $p_0(t)$ that after t steps from the start (we count only steps that result in the change of the state) there have been no encounters between two nodes possessing 0 and 1, respectively (in other words, at each step an empty node has encountered a node having 0 or 1):

$$p_0(t) = \frac{1}{2(n+1)} \frac{n!}{(n-t)!} \times \sum_{a \in \{0,1,\cdots,2^t-1\}} \{f_t(a)!(t-f_t(a))!$$
$$\times \prod_{s=2}^{t} \frac{1}{(s+1)(n-s+1) + 2(f_{s-1}(a)+1)(s-f_{s-1}(a))}\},$$

where a runs through all integer values from 0 to $2^t - 1$, and the functions f_s, $s = 1, 2, \cdots, t$ of a are defined as follows: write $a = \sum_{k=1}^{t} a_k 2^{t-1}$ with $a_k \in \{0, 1\}$, then $f_s(a) = \sum_{k=1}^{s} a_k$. Using this presentation we can show:

Proposition 4.1. *Assume that t, $n \to \infty$ in such a way that $t/n^{\frac{1}{3}} \to 0$. Then $p_0(t) \to 1$.*

This means that, asymptotically for large n, the system behaves at the beginning for long time like a Pólya urn, and only after a "macroscopic" time the encounters between non-empty chunk holders become likely. Another limiting case is obtained when the number of chunks is already very large. Then it is reasonable to shift to continuous modelling and continuous time t and assume densities of the chunk holders: $n_0 = |(\text{nodes with only chunk } 0)| \ /(n+2)$ and similarly n_1 and $n_\emptyset$ for chunk 1 holders and the empty nodes, correspondingly. They would evolve according to an autonomous system of differential equations:

$$\begin{aligned} n_0'(t) &= n_0(t)(n_\emptyset(t) - n_1(t)) \\ n_1'(t) &= n_1(t)(n_\emptyset(t) - n_0(t)) \\ n_\emptyset'(t) &= -n_\emptyset(t)(n_0(t) + n_1(t)), \end{aligned} \tag{1}$$

where $'$ denotes the time derivative.

Now, initial conditions are needed. Returning to the discrete system, at the moment when the number of chunks reaches the value n^α with $0 < \alpha < \frac{1}{3}$, the distribution of the number of chunk holders is uniform. The precise value of α is not principal. Denote $c = n^\alpha/n$. At this instance, we heuristically fuse the two approximations: take the initial conditions as $n_0(0) = \beta(c)c$, $n_1(0) = (1-\beta(c))c$, and $n_\emptyset(0) = 1 - c$, where $\beta(c)$ is a random variable taking values in the interval $[0, 1]$. We conjecture that, as $n \to \infty$, $\beta(c)$ converges to a uniformly distributed random variable $\beta(+0) \sim U[0, 1]$, as it should reflect the Pólya-like phase of evolution. The steady state is reached at time $t \approx \log n$, due to the exponential growth at the start. Numerical simulations indicate that the system of equations above really have such a limit for the steady state, see Figure 1 below. On the other hand, this limiting scheme seems to be a quite realistic approximation already for such a small number of nodes as 100. This approach could probably be extended to more realistic scenarios with more chunks and the seed. This would result in a different distribution for $\beta(+0)$.

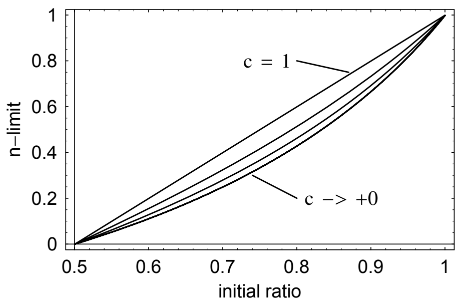

Fig. 1. Steady state solution for the larger component of the chunk holders' density as a function of initial conditions on the interval $(0, c]$. As $c \to +0$, this function seems to converge. The values were taken as $1, 0.5, 10^{-3}, 10^{-6}, 10^{-20}$ with seemingly monotonous convergence. The curves of two smallest values merge visually.

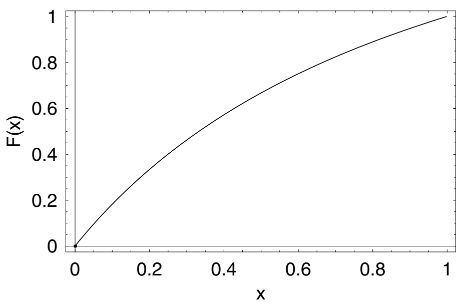

Fig. 2. Distribution function $F(x)$ for the steady state value of the successful (possessing both chunks) node density x. Solid line corresponds to the solution of the differential equations with $c = 10^{-20}$ and the points for a simulated system with $n = 98$ with experiments repeated 1000 times.

5 Simulation Results

In our simulations, we always assume that N nodes arrive simultaneously to the system at time zero. The file to be broadcast is available as a set of C chunks at the seed node. In order to download these chunks, each node initiates encounters with other nodes in the system, and downloads missing chunks. Having downloaded the whole file, the node leaves the system. Each node initiates encounters as a Poisson process with rate one, and the download time of a chunk is zero. The main metric of interest in this model is the average time to broadcast the file to N nodes.

We shall simulate the above system for two different topologies. In the *fully-connected* topology, a node can contact any other node present in the system. The second topology is the Chord topology which was described in Section 3.

5.1 The Rare Chunk Phenomenon

We shall first assume that the network graph is fully connected, i.e., a node can initiate an encounter with any other node present in the system.

In Figure 3, we plot the expected number of uploads of a chunk by the seed node in ascending order of the number of uploads for $C = 100$.

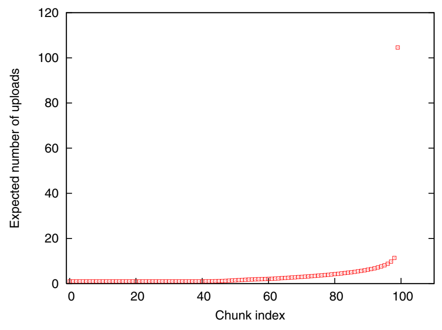

Fig. 3. Expected number uploads of chunks by the seed node. *Random chunk policy.* $N = 1000$. $C = 100$.

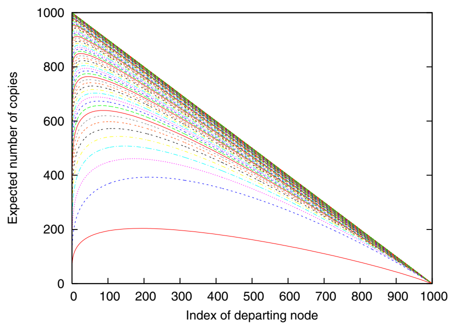

Fig. 4. Expected number of copies of the chunks as seen by departing nodes. *Random chunk* policy. $N = 1000$. $C = 100$.

From this plot, 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. In Figure 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 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 made in [5].

5.2 Deterministic Last K Chunks Policy

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 the last missing chunk of each node, say, to be uniformly random and independent of that of all other nodes. This would make the seed node less critical to the downloading process.

A decentralized way 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 was proposed in [7]:

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.*

The order in which these K chunks will be downloaded is arbitrary and is not decided beforehand.

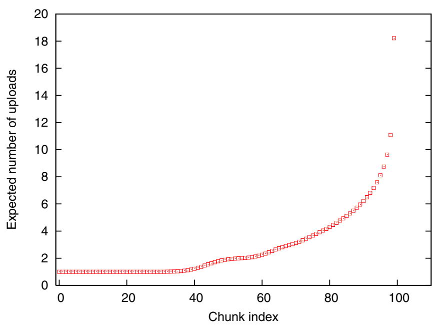

Fig. 5. Expected number uploads of chunks by the seed node. *Deterministic last chunk* policy. $N = 1000$. $C = 100$. $K = 1$.

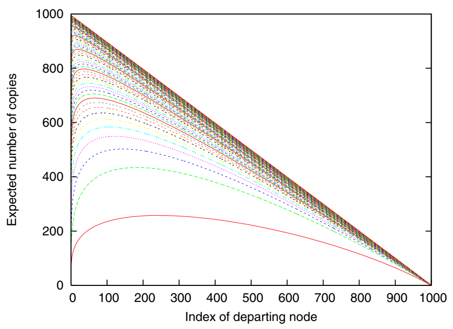

Fig. 6. Expected number of copies of the chunks as seen by departing nodes. *Deterministic last chunk* policy. $N = 1000$. $C = 100$. $K = 1$.

For $K = 1$, we compare the performance measures of this policy with those of the random selection policy. The initial number of nodes in the system is the same as before (i.e., $N = 1000$). Figure 5 and 6 show the number of expected (ordered) number of uploads by the seed node, and the expected number of copies of each chunk at departure instants, respectively, for $C = 100$. On comparing these with

Figures 3 and 4 we see that the maximum uploads by the seed node has reduced compared with the random selection policy. The better performance of this policy can be ascribed to two reasons. Firstly, 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 present in the system for longer duration, thereby reducing the load on the seed. Secondly, in the *random* chunk policy, the chunks which entered the system initially would spread quickly to all nodes. However in the *deterministic last* K chunks policy a proportion of nodes would reject these chunks thereby reducing the imbalance between the rapidly multiplying chunks and rare chunks. However, one cannot expect to improve the performance by increasing K a lot. As K increases, the number of unsuccessful encounters, i.e., encounters in which chunks are rejected, would also increase, and this would increase the broadcast time again.

5.3 Expected Broadcast Time

We now compare the time to broadcast all the C chunks to the N nodes using the *random* and *deterministic last* K policies. We are interested in the scaling of the broadcast time with the number of nodes. In order to do so we simulate the system for $N = 2^j, j = 1, ..., 9$. For the Chord topology, we shall always assume that the number of addresses available, 2^M, is twice the number of initial nodes in the system. In these simulations we considered an augmented Chord topology where each node was given two additional, randomly distributed fingers. The finger tables are updated instantaneously as soon as a node leaves the system. Each node initiates encounters by selecting one finger either at random or, as another variant, cyclically. Since the Chord topology is a random graph (it is random because the $N + 1$ different addresses are chosen randomly), the results for the Chord graph are obtained by averaging over different topologies.

We recall that in this model each node is initiating an encounter at rate one. Hence, the expected time to broadcast would give the average number of encounters initiated by the last departing node. In Figures 7 and 8, we plot the expected time to broadcast the file in both the Chord and fully-connected topologies and for the *random* and *deterministic last* chunk policy. A surprising and counter-intuitive observation from these figures is that the expected time to broadcast on Chord topology is smaller than on the fully connected one for the *random* chunk selection policy. The result is surprising because one would expect restricted connectivity to increase the expected time to broadcast. However, the restricted connectivity slows down the spread of the intially released chunks as they are unable to spread to all the network quickly. This reduces the imbalance in the chunk distribution, and also retains the nodes for a longer duration in the system, thereby reducing the expected broadcast time. We note that the larger the number of nodes in the system, the faster the system evolves and the greater is the number of uploaders in the system. The effect of the Chord topology is quite similar to the *deterministic last* K chunks policy in that there would be some set of nodes where the initially released chunks would not reach quickly enough, thereby reducing the chunk imbalance. The trade-off between slowing

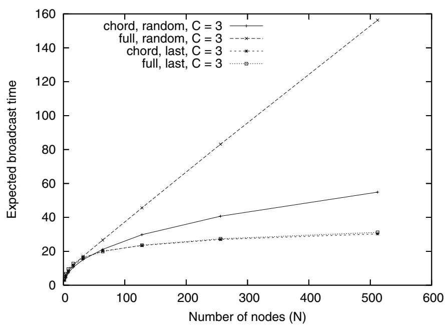

Fig. 7. Expected broadcast time for different values of N. $K = 1$. $C = 3$.

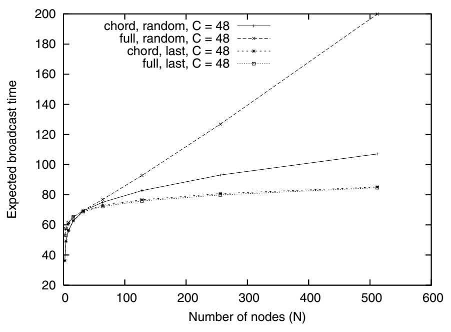

Fig. 8. Expected broadcast time for different values of N. $K = 1$. $C = 48$.

down the chunk replication (increasing the broadcast time) and retaining the nodes for a longer duration is apparent when we compare the expected broadcast time for small and large values of N. For smaller values of N, the *deterministic last K* chunk policy performs worse than the *random* chunk policy because the number of "unsuccessful encounters" may outweigh the benefits of retaining the nodes for a longer duration.

In Figures 9 and 10, we compare the performance of cyclical finger selection in the Chord topology to that of random finger selection. We observe that cyclical finger selection yields slightly better performance than random finger selection.

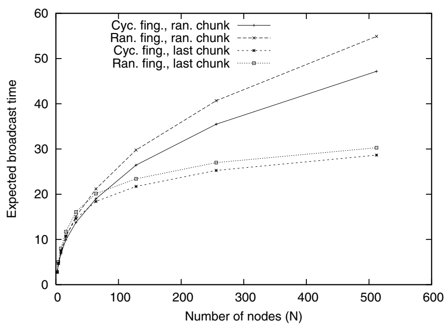

Fig. 9. Comparing cyclical and random finger selection. Expected broadcast time for different values of N. $K = 1$. $C = 3$.

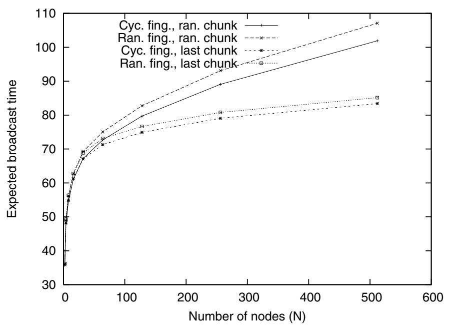

Fig. 10. Comparing cyclical and random finger selection. Expected broadcast time for different values of N. $K = 1$. $C = 48$.

6 Conclusions

We have shown that even in a flash crowd scenario, chunk-wise distribution of large files can be based on random encounters within an overlay. However, attention must be paid to the avoidance of the rare chunk phenomenon by applying some other chunk selection algorithm than random choice. The extremely simple deterministic last chunk policy was shown to improve the situation already.

To analyse the flash crowd scenario mathematically, we sketched an approach where the early evolution of the system is modelled by Pólya's urn model, and after the stabilization of the chunk distribution the evolution is described by a system of differential equations.

As a novel idea, suggested by the hypercube-based optimal solution to the chunk distribution problem, we experimented by replacing random encounters by encounters restricted to the neighbour nodes in the Chord topology. The results were promising and prompt for further study of this kind of systems.

Acknowledgement. Part of this work was done in the EU FP6 projects Euro-NGI/REDLARF and project 034413 NET-ReFOUND.

References

1. Cohen, B.: BitTorrent specification (2006) `http://www.bittorrent.org`
2. Yang, X., de Veciana, G.: Service Capacity in Peer-to-Peer Networks. In: Proc. IEEE INFOCOM, Hong Kong (2004)
3. Stoica, I., Morris, R., Liben-Nowell, D., Karger, D.R., Kaashoek, M.F., Dabek, F., Balakrishnan, H.: Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications. IEEE/ACM Transactions on Networking 11(1), 17–32 (2003)
4. Massoulie, L., Vojnovic, M.: Coupon Replication Systems. In: Proc. ACM SIGMETRICS, Banff, Canada (2005)
5. Felber, P., Biersack, E.: Cooperative Content Distribution: Scalability through Self-Organization. In: Babaoğlu, Ö., Jelasity, M., Montresor, A., Fetzer, C., Leonardi, S., van Moorsel, A.P.A., van Steen, M. (eds.) Self-star Properties in Complex Information Systems. LNCS, vol. 3460, Springer, Heidelberg (2005)
6. Bharambe, A.R., Herley, C., Padmanabhan, V.N.: Analyzing and Improving a BitTorrent Network's Performance Mechanisms. In: Proc. IEEE INFOCOM, Barcelona (2006)
7. Norros, I., Prabhu, B., Reittu, H.: Flash crowd in a file sharing system based on random encounters. In: Inter-Perf, Pisa, Italy (2006) `http://www.inter-perf.org`
8. Qiu, D., Srikant, R.: Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks. In: Proc. ACM Sigcomm, Portland, OR (2004)
9. Clévenot-Perronnin, F., Nain, P., Ross, K.W.: Multiclass P2P Networks: Static Resource Allocation for Service Differentiation and Bandwidth Diversity. Performance Evaluation 62(1-4), 32–49 (2005)
10. Mundinger, J., Weber, R., Weiss, G.: Analysis of peer-to-peer file dissemination. To appear in Performance Evaluation Review, Special Issue on MAMA 2006 (2006)
11. Kwon, C.H., Chwa, K.Y.: Multiple message broadcasting in communication networks. Networks 26, 253–261 (1995)
12. Ho, C.: Optimal broadcasting on SIMD hypercubes without indirect addressing capability. J. Parallel Distrib. Comput. 13, 246–255 (1991)