Cooperative Content Distribution: Scalability through Self-Organization

Pascal Felber[1][*] and Ernst W. Biersack[2]

[1] University of Neuchâtel, Switzerland
`pascal.felber@unine.ch`
[2] Institut EURECOM, France
`erbi@eurecom.fr`

Abstract. Peer-to-peer networks have often been touted as the ultimate solution to scalability. Although cooperative techniques have been initially used almost exclusively for content lookup and sharing, one of the most promising application of the peer-to-peer paradigm is to capitalize the bandwidth of client peers to quickly distribute large content and withstand flash-crowds (i.e., a sudden increase in popularity of some online content). Cooperative content distribution is based on the premise that the capacity of a network is as high as the sum of the resources of its nodes: the more peers in the network, the higher its aggregate bandwidth, and the better it can scale and serve new peers. Such networks can thus spontaneously adapt to the demand by taking advantage of available resources. In this paper, we evaluate the use of peer-to-peer networks for content distribution under various system assumptions, such as peer arrival rates, bandwidth capacities, cooperation strategies, or peer lifetimes. We argue that the *self-scaling* and *self-organizing* properties of cooperative networks pave the way for cost-effective, yet highly efficient and robust content distribution.

1 Introduction

Peer-to-peer systems, in which peer computers form a cooperative network and share their resources (storage, CPU, bandwidth), have attracted a lot of interest lately. After the apparition of the first truly successful peer-to-peer systems [1, 2], and the significant amount of research conducted in Academia and in the Industry, most researchers now agree that peer-to-peer systems are more than just a fashion phenomenon. They offer great potential for building cooperative networks that are self-organizing, efficient, scalable, and reliable.

Research in peer-to-peer networks has so far mainly focused on content storage and lookup, but fewer efforts have been spent on its actual distribution. By capitalizing the *bandwidth* of peer nodes, cooperative architectures offer great potential for addressing some of the most challenging issue of today's Internet: the cost-effective distribution of bandwidth-intensive content to thousands of simultaneous users both Internet-wide and in private networks, and the resilience

---

[*] This work was performed while the author was at Institut EURECOM.

to "flash crowds"—a huge and sudden surge of request traffic that usually leads to the collapse of the affected server, as happened to the Web sites of major media companies during the events of 9/11.

Cooperative content distribution networks are inherently *self-scalable*, in that the bandwidth capacity of the system increases as more peers arrive: each new peer requests service from, but also provides service to, the other peers. The network can thus spontaneously adapt to the demand by taking advantage of the resources provided by every peer.

As an example of the self-scaling properties of cooperative content distribution, consider the situation where a server must replicate a critical file to a large number of clients, e.g., an antivirus update, to all $100,000$ machines of a large company. Given a file size of 4 MB, and a server (client) bandwidth capacity of 100 Mb/s (10 Mb/s) with 90% link utilization, a classical client/server distribution protocol would distribute the file by iteratively serving groups of 10 simultaneous clients in $u = \frac{32 \text{ Mb}}{9 \text{ Mb/s}} = 3.55$ seconds. Updating $100,000$ clients would thus necessitate $\frac{100,000}{10}u$, i.e., almost 10 hours.

In contrast, cooperative distribution leverages the bandwidth of the nodes that have already obtained the file, thus dynamically increasing the service capacity of the system as the file propagates to the clients. As each client that has already received the file can serve another client while the server updates 10 new clients, we can compute the number of clients updated at time t as $n(t) = 2n(t-u) + 10 = 2^{\lfloor t/u \rfloor}10 - 10$. Updating $100,000$ clients would thus necessitate less than 1 minute, as can be observed in Figure 1. The exponential increase of peer-to-peer distribution provides a sharp contrast with the linear progression of traditional client/server distribution, and illustrates the self-scaling property of cooperative networks.

We have studied in [3] the scalability of cooperative distribution architectures, where each peer has equal upload and download rates of b, and there are no failures. We have shown that it takes $1 + \lfloor log_k N \rfloor \cdot \frac{k}{C}$ rounds to serve N peers organized in k spanning trees, where a round is the time needed to download the complete file at rate b and c is the number of chunks the file is split into. This result indicate that the number of peers that complete the download grows exponentially in time and in the number of chunks (large numbers of chunks allow all peers to busy most of the time). Obviously, such static and homogeneous scenarios are rare in real-world system, where the peers typically have different (often asymmetric) bandwidth, can join and leave anytime, and have only a limited view of the complete system insufficient for global optimizations.

In this paper, we discuss and evaluate the use of peer-to-peer networks for content distribution under various *realistic* system assumptions, such as peer arrival rates, bandwidth capacities, cooperation strategies, or peer lifetimes. We argue that a key property for the good scalability of content distribution architecture is their ability to *self-organize* by letting each peer select dynamically which other peers to cooperate with over time. This study exhibits the trade offs encountered when deploying a content distribution network and emphasizes that

the choice of a specific strategy strongly depends on the considered optimization criteria.

2   Cooperative Content Distribution

In order to maximize the participation of each of the peers in the network, large content is typically split into many blocks (or "chunks") that are directly exchanged between the peers—a technique also known as "swarming." The large number and small size of the chunks are key to quickly create enough diversity in the network for each of the peers to be useful to some other peers.

Cooperative networks are usually build incrementally, with joining peers dynamically connecting to existing peers to eventually create complex mesh topologies. In practice, a peer usually knows only a subset of other peers, and actively trades with an even smaller subset. In addition to the actual structure of the mesh (i.e., which and how many neighbors each peers has), two factors are crucial to the global effectiveness of the content distribution process:

– *Peer selection strategy:* which among our neighboring peers will we actively trade with, i.e., serve or request chunks from?
– *Chunk selection strategy:* which chunks will we preferably serve to, or request from, other peers?

The popular BitTorrent [4] tool, which we have studied extensively in [5], empirically selects the peers that offer the best upload and download rates to trade with ("tit-for-tat" strategy). When a new peers joins the system, it initially requests random chunks in order to quickly receive some data and become useful to the system; thereafter, it requests the rarest chunks among those owned by its neighbors, because rare chunks have a higher "trading value" than common chunks.

The main focus of our study is to evaluate several peer and chunk selection strategies, and determine which ones perform best in various deployment scenarios. For the purpose of our evaluation, we only study the extreme case where each peer knows all other peers (fully-connected mesh) and can potentially trade with any of those peers during its lifetime, although we impose a limit on the number of simultaneous active connections. This assumption allows us to observe the asymptotic behavior of the various cooperative strategies.

2.1   Deployment Scenarios

We specifically focus on two deployment scenarios that correspond to real-world applications of cooperative content distribution. In the first scenario, we assume that some critical content need to be quickly replicated on a large number of machines within the private network of a large company. This essentially corresponds to a *push* model where all the peers are known beforehand and distribution stops once the content has been fully replicated on all the machines, which typically have similar connectivity (homogeneous bandwidth).
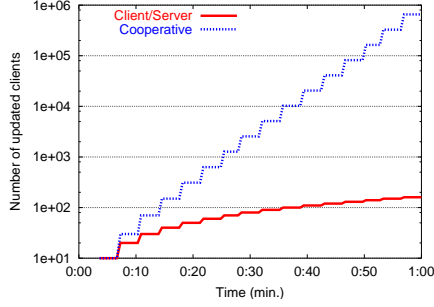
Fig. 1. Scalability of cooperative content distribution: the number of clients that successfully receive a file increases linearly with client/server distribution, and exponentially with cooperative distribution.
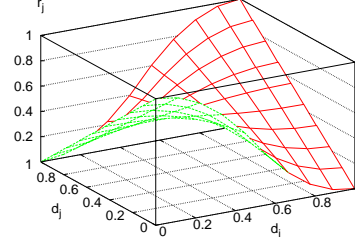
Fig. 2. *Adaptive-missing* peer selection strategy: Peer i computes the rank r_j of every other peer j as a function of d_i and d_j (here, we have $U(0,1) = 0.5$).

The second scenario corresponds to the traditional Internet flash-crowd phenomenon, where a large number of clients access almost simultaneously some large popular content. This corresponds to a *pull* model with continuous arrival of the peers. Distribution continues over several peer "generations," with some peers arriving well after the first peers have already left. The clients typically have heterogeneous bandwidth capacities, ranging from dial-up modems to broadband access (asynchronous and synchronous).

2.2   Notation

We denote by $\mathcal{C}$ the set of all chunks in the file being distributed, and by $\mathcal{D}_i$ and $\mathcal{M}_i$ the set of chunks that peer i has already downloaded and is still missing, respectively (with $\mathcal{M}_i \cup \mathcal{D}_i = \mathcal{C}$ and $\mathcal{M}_i \cap \mathcal{D}_i = \emptyset$). Similarly, $d_i \triangleq |\mathcal{D}_i|/|\mathcal{C}|$ and $m_i \triangleq |\mathcal{M}_i|/|\mathcal{C}|$ correspond to the proportions of chunks that peer i has already downloaded and is still missing, respectively. The function $U(a,b)$ returns a random number uniformly distributed in the interval $[a,b]$.

2.3   Peer Selection

The peer selection strategy defines "trading relationships" between peers and is the key factor to the network's *self-organization* property. In our simplified model, we assume that all the peers know one another. When a peer has some chunks available and some free uplink bandwidth capacity, it will use a peer selection strategy to *locally* determine which other peer it will serve next. In this paper, we propose and evaluate the following peer selection strategies:

  – *Random:* A peer is selected at random. This strategy is expected to achieve good diversity in peer connectivity.

- *Least missing:* Preference is given to the peers that have many chunks, i.e., we serve in priority peer j with $d_j \geq d_i$, $\forall i$. This strategy is inspired by the SRPT (shortest remaining processing time) scheduling policy that is known to minimize the service time of jobs [6].
- *Most missing:* Preference is given to the peers that have few chunks (newcomers), i.e., we serve in priority peer j with $d_j \leq d_i$, $\forall i$. The rationale behind this strategy is to evenly spread chunks among all peers to allow them to quickly serve other peers.
- *Random least missing:* Similar to *least missing*, but with a random component added in the selection process. We serve in priority peer j with the lowest rank, computed as $U(0, |\mathcal{M}_j|^2)$.
- *Random most missing:* Similar to *most missing*, but with a random component added in the selection process. We serve in priority peer j with the lowest rank, computed as $U(0, |\mathcal{D}_j|^2)$.
- *Adaptive-missing:* Peers that have many chunks serve peers that have few chunks, and vice-versa, with more randomness introduced when download tend to be half complete. A peer i will serve in priority peer j with the lowest rank r_j, computed as:

$$r_j^{Rnd} = U(0, 1)$$
$$r_j^{Det} = \begin{cases} d_j & : & d_i \geq 0.5 \\ m_j & : & d_i < 0.5 \end{cases}$$
$$f = (1 - |2d_i - 1|)^2$$
$$r_j = f r_j^{Rnd} + (1 - f) r_j^{Det}$$

where r_j^{Rnd} and r_j^{Det} are the random and deterministic ranks of peer j, respectively, and $f \in [0, 1]$ is a weight factor that controls randomness and is maximal when peer i is exactly half-way through the download. A graphical representation of r_j as a function of d_i and d_j is shown in Figure 2. This strategy is expected to give good chances to newcomers without artificially slowing down peers that are almost complete.

Although not shown in this paper because of space constraints, we have also experimented with additional peer selection strategies that take into account the free bandwidth capacities of the peers.

2.4   Chunk Selection

The chunk selection strategy specifies which chunks should preferably be traded between the peers. Chunk selection can be performed by the receiver (which requests specific chunks from its neighbors) or by sender (which decides which chunk it will send next on an active connection). With both interaction models, obviously, the chosen chunk must be held by the sender and not by the receiver. In our simplified model, we assume that every peer knows the list of chunks held by its neighbors (i.e., all peers with a fully-connected mesh topology) and that the chunk selection strategy is applied on the sender's side. In this paper, we evaluate the following chunk selection strategies:

– *Random:* The sending peer i selects a chunk $c \in (\mathcal{D}_i \cap \mathcal{M}_j)$ at random among those that it holds and the receiving peer j needs. This strategy ensures good diversity of the traded chunks.
– *Rarest:* The sending peer i selects the rarest chunk $c \in (\mathcal{D}_i \cap \mathcal{M}_j)$ among those that it holds and the receiving peer j needs. Rarity is computed from the number of instances of each chunk held by the peers known to the sender. This strategy is expected to maximize the number of copies of the rarest chunk in the system.

3   Experimental Setup

For the purpose of evaluating cooperative content distribution, we have developed a simulator that models various types of peer-to-peer networks and allows us to observe step-by-step the distribution of large files among all peers in the systems, according to several metrics. Although we have taken extra care to reproduce realistic operating conditions, we have yet made some assumptions in order to simplify and speed up the simulations. In particular, we do not consider failures (peer or network) nor link congestion in any of the experiments, and we do not favor long-running connections overt short connections as real systems usually do. We also intentionally present here the results of the simulations of extreme scenarios (little heterogeneity, limited server bandwidth) that best exhibit the differences between the various aforementioned strategies; more moderate scenarios have shown the same general trends, albeit with lower intensity.

3.1   Simulation Methodology

Our simulator is essentially event-driven, with events being scheduled and mapped to real-time with a millisecond precision. The transmission delay of each chunk is computed dynamically according the link capacities (minimum of the sender uplink and receiver downlink) and the number of simultaneous transfers on the links (bandwidth is equally split between concurrent connections).

Once a peer i holds at least one chunk, it becomes a potential server. It first sorts its neighboring peers according to the specified peer selection strategy. It then iterates through the sorted list until it finds a peer j that (1) needs some chunks from $\mathcal{D}_i$ ($\mathcal{D}_i \cap \mathcal{M}_j \neq \emptyset$), (2) is not already being served by peer i, and (3) is not overloaded. We say that a peer is overloaded if it has reached its maximum number of connections *and* has less than 128 kb/s bandwidth capacity left. Peer i then applies the specified chunk selection strategy to choose the best chunk to send to peer j. Peer i repeats this whole process until it becomes overloaded or finds no other peer to serve.

3.2   Setup

Our simulator allows us to specify several parameters that define its general behavior and operating conditions. The most important ones relate to the content being transmitted (file size, chunk size), the peer properties (arrival rates,

Parameter	Value
Chunk size	256 kB
File size	200 chunks (i.e., 51.2 MB)
Peer arrival rate	
Simultaneous (push)	5000 peers at t_0
Continuous (flash-crowd)	Poisson with rate $\lambda = \frac{1}{2.5 \text{ s}}$
Peer bandwidth (downlink/uplink)	
Homogeneous, symmetric	100% peers: 128/128 kb/s
Homogeneous, asymmetric	100% peers: 512/128 kb/s
Heterogeneous, asymmetric	50% peers: 512/128 kb/s
	50% peers: 128/64 kb/s
Peer lifetime	
Selfish	Disconnects when complete
Altruistic	Remains 5 minutes online
Active connections per peer	5 inbound and 5 outbound
Number of origin peers	1 (bandwidth: 128/128 kb/s)
Duration of simulation	12 h or more
Peer selection strategy	*Varies*
Chunk selection strategy	*Varies*

Table 1. Parameters used in the simulations.

bandwidth capacities, lifetimes, number of simultaneous active connections), and global simulation parameters (number of initial servers or "origin peers," simulation duration, peer selection strategy, chunk selection strategy). Table 1 summarizes the values of the main parameters used in our simulations.

3.3 Metrics

We have considered several metrics in our evaluation of cooperative content distribution. We briefly outline below the major properties that we have observed during the simulations:

– *Download times:* The duration of the file download as experienced by individual peers. In general, shorter times are better and variance should be minimized.
– *Download progress:* The progress of the file download over time by each of the peers. In general, regular progress is desirable (i.e., peers should not be stalled for long periods of time).
– *Chunk capacity:* The evolution over time of the number of chunks in the system. Larger numbers of chunks usually correspond to greater "service potential."
– *Chunk distribution:* The evolution over time of the frequency of the chunks in the system. The variance of chunk frequencies should be minimized.
– *Overall efficiency:* The ratio of the effective throughput of the system to its optimal throughput, computed as the sum of the bandwidth capacities of all active peers. Higher values are better.

4   Simulation Results

We now present our simulation results. Due to space constraints, we only discuss here a small selected subset of these results.

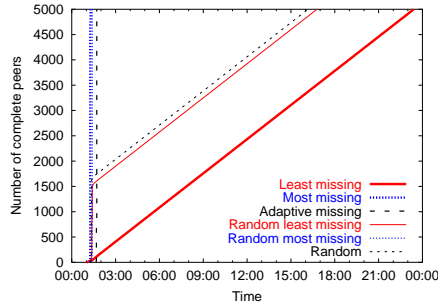### 4.1   Simultaneous Arrivals

Fig. 3. Completion times for the *random* chunk selection strategy, with simultaneous arrivals, homogeneous and symmetric bandwidth, and selfish peers.
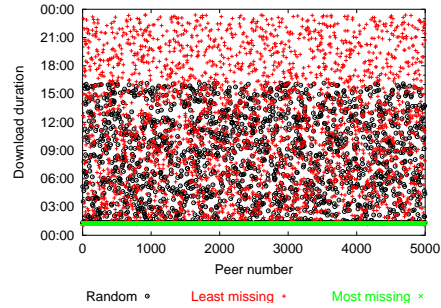

Fig. 4. Download duration for the *random* chunk selection strategy, with simultaneous arrivals, homogeneous and symmetric bandwidth, and selfish peers.

The chunk selection strategy can have a significant impact on the effectiveness of cooperative content distribution, especially when considering selfish peers. As shown in Figure 3, several of the peer selection strategies need a long time to replicate the file on all clients. First consider that the transmission of all 200 chunks of the file over a 128 kb/s connection requires $\frac{200 \cdot 256 \cdot 8 \text{ kb}}{128 \text{ kb/s}} = 3200$ seconds, i.e., slightly less than one hour. If we could construct a linear chain, with each client receiving the file from the previous peer in the chain and serving it *simultaneously* to the next one, we could theoretically approach this asymptotic limit. In practice, because we only consider the transmission of complete chunks and we share bandwidth capacities between several connections, we expect to experience lower efficiency.

We can explain the low performance of the *least missing* peer selection strategy by the fact that the server will initially only serve the same 5 peers that are closest to completion. These peers will in priority exchange chunks with each other and then slowly propagate some chunks to the other peers, which remain mostly idle because they have no rare chunks to trade. As completed peers leave immediately the system, we essentially have one server (the initial peer) that iteratively serves batches of 5 peers at a time, which explains the low efficiency of the *least missing* strategy. One should note, however, that this strategy minimizes the download time of the first complete peer. Figure 4 shows, indeed, that the download times have the highest variance with the *least missing* strategy.
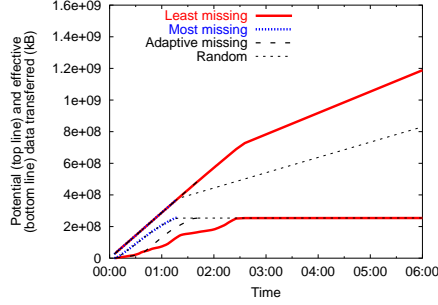
Fig. 5. Potential and effective data transferred for the *random* chunk selection strategy, with simultaneous arrivals, homogeneous and symmetric bandwidth, and selfish peers (the four top lines are all stacked initially).
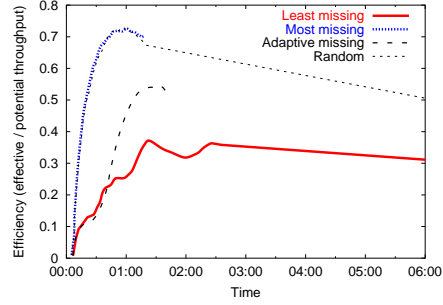
Fig. 6. Overall efficiency ratio for the *random* chunk selection strategy, with simultaneous arrivals, homogeneous and symmetric bandwidth, and selfish peers.

At the other extreme, the *most missing* peer selection strategy tries to make all clients progress simultaneously, thus making them quickly and equally useful to others. This results in a better utilization of the available resources, as can be seen in Figures 5 and 6. By "artificially" delaying the departure of the peers, we always keep a large service capacity and ensure that all peers complete approximately at the same time. In the case of simultaneous arrivals, we can observe that the *most missing* strategy minimizes the download time of the last complete peer.
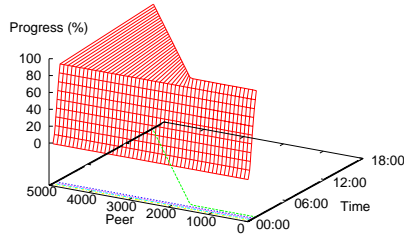

Fig. 7. Download progress for the *random* peer selection strategy, with the *random* chunk selection strategy, simultaneous arrivals, homogeneous and symmetric bandwidth, and selfish peers.
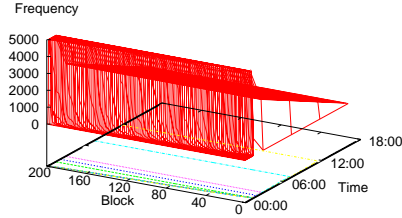
Fig. 8. Chunk distribution for the *random* peer selection strategy, with the *random* chunk selection strategy, simultaneous arrivals, homogeneous and symmetric bandwidth, and selfish peers.

The *random* peer selection strategy is expected to let all peers progress at approximately the same rate, and thus to behave roughly like the *most missing* strategy. We observe, however, that only one third of the peers complete simultaneously and the rest essentially follow the same pattern as the *least missing* strategy. This problem can be tracked down to the *random* chunk selection. Indeed, the chunks that were injected first in the system exist in many instances, while the latter chunks are very rare, with the server doing nothing to correct this imbalance. Most of the peers quickly reach near completion, as shown in Figure 7, but many require much time to obtain the few missing chunks—often just one—that are only held by the origin server.
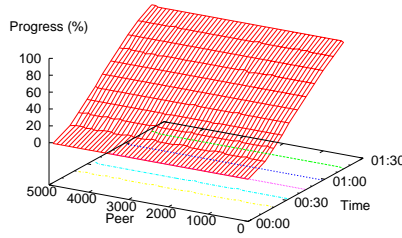

Fig. 9. Download progress for the *most missing* peer selection strategy, with the *random* chunk selection strategy, simultaneous arrivals, homogeneous and symmetric bandwidth, and selfish peers.
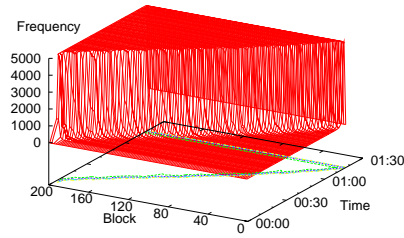
Fig. 10. Chunk distribution for the *most missing* peer selection strategy, with the *random* chunk selection strategy, simultaneous arrivals, homogeneous and symmetric bandwidth, and selfish peers.

This problem can be observed more clearly in Figure 8, which shows the evolution of the frequency of each chunk in the system. We remark that the very first chunk on the right reaches a maximum frequency of approximately $1,200$ copies after 1 hour and falls back to zero after the first batch of peers have left. Thereafter, that extremely rare chunk is served only by the origin peer, because clients behave selfishly and leave as soon as they have downloaded the chunk. In contrast, the *most missing* peer selection strategy ensures regular progress of all the peers and a quick and even dissemination of the chunks after they have been injected in the system, as can be seen in Figures 9 and 10. Indeed, the server gives rare chunks to peers that are expected to remain online for some time and help to their dissemination. In similar settings but with altruistic peers that remain online for some time after completion, this pathologic situation does not arise anymore under random peer selection.

As previously mentioned, the download times of the *least missing* strategy have a high variance, with some peers progressing very fast and other very slowly. This can be clearly observed in Figures 11 and 12. The random variants of the *least missing* and *most missing* peer selection strategies exhibit some of the
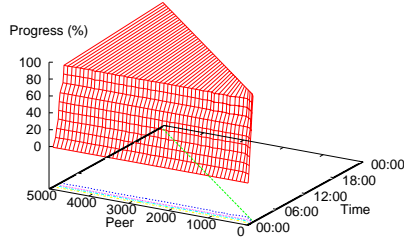
Fig. 11. Download progress for the *least missing* peer selection strategy, with the *random* chunk selection strategy, simultaneous arrivals, homogeneous and symmetric bandwidth, and selfish peers.
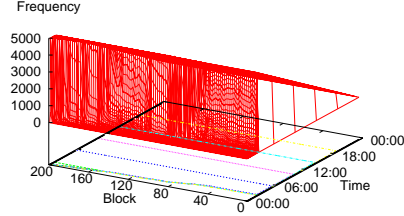

Fig. 12. Chunk distribution for the *least missing* peer selection strategy, with the *random* chunk selection strategy, simultaneous arrivals, homogeneous and symmetric bandwidth, and selfish peers.

trends of their deterministic counterpart, but with less intensity. We will not discussed them further in this paper. Finally, the *adaptive missing* strategy is interesting because it seems to inherit some of the good properties of each of the extreme *least missing* and *most missing* strategies. It initially quickly and evenly replicates blocks in the system and, at the same time, does not artificially prevent near-complete peers to finish their download (this problem is of greater important in the case of continuous arrivals, as we shall see shortly).
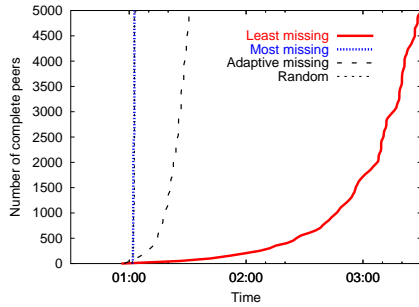

Fig. 13. Completion times for the *rarest* chunk selection strategy, with simultaneous arrivals, homogeneous and symmetric bandwidth, and selfish peers.
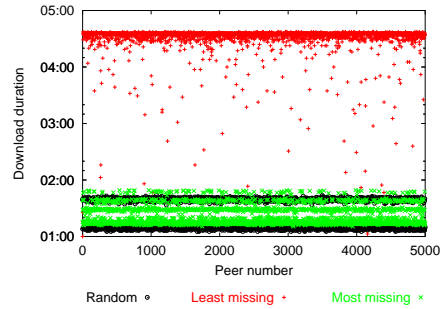

Fig. 14. Download duration for peers with heterogeneous and asymmetric bandwidth, with the *rarest* chunk selection strategy, simultaneous arrivals, and selfish peers.

When switching to the *rarest* chunk selection strategy, we observe in Figure 13 significant performance improvements, particularly for the *random* peer strategy that becomes as efficient as *most missing*, and the *least missing* strategy

that shows a seven-fold improvement. In contrast to the *random* chunk selection strategy, we do not experience the pathological situation where the origin sequentially serves the rare missing chunks to almost-complete peers.

If we consider heterogeneous bandwidths with 128 and 512 kb/s downlink capacities, we can clearly see in Figure 14 the two distinct classes of peers on the basis of their download durations, best visible with the *random* peer selection strategy. The *most missing* strategy tends to diminish this gap by enforcing peers to progress at approximately the same speed. Finally, the *least missing* strategy behaves as for homogeneous bandwidths, with few peers completing very fast and many peer much later.
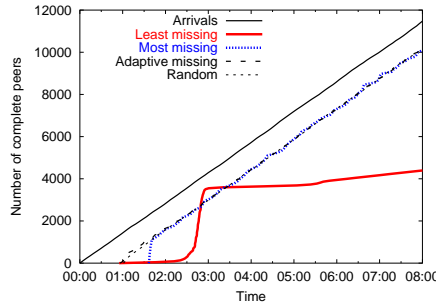
4.2 Continuous Arrivals

Fig. 15. Completion times for continuous arrivals, with the *rarest* chunk selection strategy, homogeneous and asymmetric bandwidth, and selfish peers.
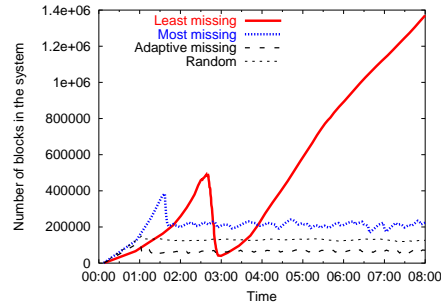
Fig. 16. Chunk capacity of the system, with the *rarest* chunk selection strategy, homogeneous and asymmetric bandwidth, and selfish peers.

We have studied the case of continuous arrivals and asymmetric bandwidth (512/128 kb/s ADSL) with both selfish and moderately altruistic peers. We observed interesting behaviors that were consistent across both settings but more pronounced in the case of altruistic peers. We can see in Figures 15 and 17 that the *random* and *adaptive missing* peer selection strategies keep up with the arrival rate of the clients, with the latter looking empirically better initially. The *most missing* strategy delays the completion of a first batch of clients, before following the same slope as the arrivals but with small steps, most notable with altruistic peers. Finally, the *least missing* strategy shows an odd behavior: the number of complete peers is slow to "take off," then makes a big step to overtake all other strategies, then stalls again for a longer period of time before another even higher step, and so on. To better understand this behavior, consider that the origin peer will iteratively serve groups of 5 peers until they complete their download. The peers of a group will exchange chunks with each other in priority,
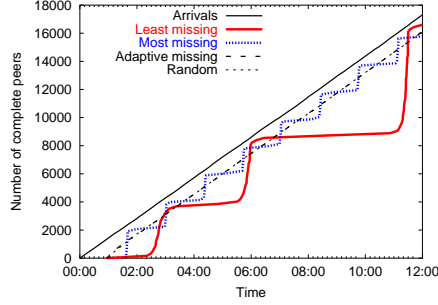
Fig. 17. Completion times for continuous arrivals, with the *rarest* chunk selection strategy, homogeneous and asymmetric bandwidth, and altruistic peers.
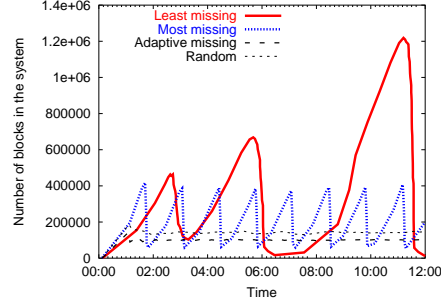
Fig. 18. Chunk capacity of the system, with the *rarest* chunk selection strategy, homogeneous and asymmetric bandwidth, and altruistic peers.

but also slowly propagate some chunks to other less-complete peers, which will quickly disseminate them among all remaining peers (they cannot indeed serve more-complete peers as the *least missing* strategy would require, because they only have blocks that the more-complete peers also hold). Therefore, we have few peers that complete very fast, and a large majority of peers that progresses slowly but steadily and eventually complete all together.

We can better understand the behavior of the peer selection strategies by considering the chunk capacity of the system with respect to time, shown in Figures 16 and 18. The *random* and *adaptive missing* strategies maintain a nearly constant number of chunks in the system. We can note that the latter looks more efficient than the former in this deployment scenario, as it achieves the same completion rate with a lower average chunk capacity. The *most missing* strategy creates a higher chunk capacity by delaying peers until the first batch completes, which corresponds to the sharp drop of chunk capacity. Thereafter, the capacity oscillates with a constant period, driven by the batches of peers that progress and complete together. Finally, the *least missing* strategy exhibits the highest volatility in chunk capacity. The system traverses phases during which it builds an extremely large chunk capacity, and then completely empties it by letting almost all peers terminate simultaneously. Interestingly, the frequency and amplitude of the oscillations increase over time. This corresponds to the steps that we have observed in Figures 15 and 17.

To better understand how each of the peer selection strategies lets the peers progress through their download, one can consider Figure 19 that shows snapshots of the progress of the peers every 15 minutes from 45 to 120 minutes. We see some clear trends: the *random* peer selection strategy lets all peers progress uniformly; with the *most missing* strategy, a large proportion of peers are at the same well advanced stage in their download; the *least missing* strategy pushes a few peers quickly to completion, and maintains the majority of peers early in their download; finally, with the *adaptive missing* strategies, we observe two
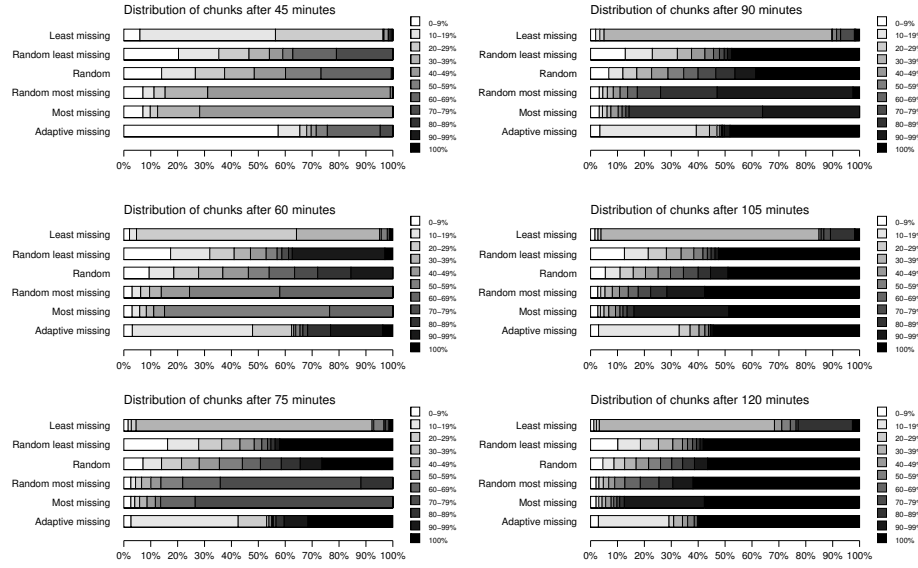
Fig. 19. Snapshot of the download progress of the peers after 45, 60, 75, 90, 105, and 120 minutes, with the *rarest* chunk selection strategy, homogeneous and asymmetric bandwidth, and selfish peers. Each segment of a bar represents the proportion of the peers that have the quantity of chunks specified by the associated color.

classes of peers with opposite completion status—early and late—and few peers in between. The random variants of *least* and *most missing* are blends of the *random* peer selection strategy and their deterministic counterpart.

4.3 Discussion

We can draw several conclusions from our simulation results. First, the peer selection strategy, which drives the way the network *self-organizes*, has an huge impact on the efficiency of content distribution. Further, the complexity of the interactions between the peers, as well as the many factors to consider in real-world networks (in particular dealing the dynamics and heterogeneity of the peers) make it hard to develop analytical models. Our simulations raise many open questions, and some of the observed behaviors will require further study for being fully understood.

The peer selection strategies that we have studied have been kept intentionally very simple. In basic scenarios they can be shown to be optimal but, in more complex environments, they need to be extended to take additional factors into account. For instance, in homogeneous settings with simultaneous arrivals, the *most missing* strategy will replicate content in the most efficient manner possible, with the number of copies of each chunk increasing exponentially in time

(see [7] for an in-depth analysis). If we now consider two populations of peers of identical sizes but with widely different bandwidth capacities. It can easily be shown that the fast peers should first replicate each chunk among them before transferring it in parallel to the slow peers. This can be achieved by tuning the *most missing* strategy so that it serves in priority, among the peers that have fewest chunks, those with the highest bandwidth capacity. We should additionally take into account dynamic factors such as the number of active connections of each peer, the popularity of the chunks that it holds, or its age. It is therefore necessary to adapt the strategies to the real complexity of the peer-to-peer network in order to optimize content distribution.

5   Conclusion

The *self-scaling* and *self-organizing* properties of peer-to-peer networks offer the technical capabilities to quickly and efficiently distribute large or critical content to huge populations of clients. Cooperative distribution techniques capitalize the bandwidth of every peer to dramatically increase the service capacity of the system. Based on the extensive simulations that we have performed, and the limited set of results shown in this paper, it appears clearly that the deployment scenarios and the cooperative strategies in use have strong influence on the effectiveness of content distribution. In particular, the chunk and peer selection strategies directly impact the delay experienced by the clients and the global throughput of the system. There is no clear "best" strategy, as each of them offers various trade offs and may prove most adequate for specific deployment scenarios. Overall, the *random* and *adaptive missing* peer selection strategies coupled with *rarest* chunk selection consistently deliver good performance and may be safely utilized as general-purpose cooperative strategies for content distribution.

References

1. Napster. (`http://www.napster.com`)
2. Gnutella. (`http://gnutella.wego.com`)
3. Biersack, E., Rodriguez, P., Felber, P.: Performance analysis of peer-to-peer networks for file distribution. In: Proceedings of the 5th International Workshop on Quality of future Internet Services (QofIS'04). (2004)
4. Cohen, B.: Incentives to build robustness in BitTorrent. Technical report, `http://bitconjurer.org/BitTorrent/bittorrentecon.pdf` (2003)
5. Izal, M., Urvoy-Keller, G., Biersack, E., Felber, P., Hamra, A.A., Garces-Erice, L.: Dissecting BitTorrent: Five months in a torrent's lifetime. In: Proceedings of the 5th Passive and Active Measurement Workshop. (2004)
6. Schrage, L.: A proof of the optimality of the shortest remaining service time discipline. Operations Research **16** (1968) 670–690
7. Yang, X., de Veciana, G.: Service capacity of peer-to-peer networks. In: Proceedings of INFOCOM. (2004)