Impact of Inner Parameters and Overlay Structure on the Performance of BitTorrent

Guillaume Urvoy-Keller
Institut Eurecom, France

Pietro Michiardi
Institut Eurecom, France

***Abstract*— In this paper we adopt a simulation approach to study the performance of the BitTorrent protocol in terms of the entropy that qualifies a torrent and the structure of the overlay used to distribute the content. We find that the entropy of a torrent, defined as the diversity that characterizes the distribution of pieces of the content, plays an important role for the system to achieve optimal performance. We then relate the performance of BitTorrent with the characteristics of the distribution overlay built by the peers taking part in the torrent. Our results show that the number of connections a given peer maintains with other peers and the fraction of those connections initiated by the peer itself are key factors to sustain a high entropy, hence an optimal system performance. Those results were obtained for a realistic choice of torrent sizes and system parameters, under the assumption of a flash-crowd peer arrival pattern.**

I. Motivation

BitTorrent is one of the most popular p2p applications in the Internet. It focuses on content replication using swarming techniques: the content is split into pieces that clients exchange with one another. Its popularity stems from the common belief that BitTorrent is efficient, i.e., fast at replicating content. A number of studies have focused on BitTorrent. Experimental works on single [4] or multiple BitTorrent sessions [7], [3], [5] have shed lights on the salient features of BitTorrent as well as on users' behaviours. Theoretical studies have focused on the scaling properties of BitTorrent [9], [8] using simplified models of the protocol. They confirmed that the built-in mechanisms of BitTorrent have the potential to scale very well. We note however that modeling of the piece exchange between peers has received little attention so far. The reason might be that it requires to keep track of the bitmaps of pieces owned by each peer in the *torrent* (BitTorrent session). The resulting combinatorial complexity soon explodes as the number of peers increases. A noticeable exception is [6] that model swarming based systems as coupon replication systems. The authors focus on the analysis of such systems for large populations.

A simulation approach is appealing to study the dynamics of piece exchange in BitTorrent. In [2], the authors use simulations to investigate the scaling behavior of the protocol for homogeneous and heterogeneous scenarios. They argue that a *smart-seed* approach should be used. With that approach, peers with the full content (a.k.a., *seeds*) upload in priority the pieces they have uploaded less. The objective is to increase the entropy of the torrent, i.e. maximizing the variety of pieces available in the torrent. A possible weakness of the approach in [2] is that they often use peer set (set of peers a given peer interacts with in a torrent) sizes clearly smaller than the ones used by current BitTorrent clients. This might affect some of their conclusions as one typically expects that to maximize entropy, clients must be connected to a large enough number of other peers in the torrent.

The main objective of this paper is to elaborate on the dynamics of piece exchange within a torrent. We address this issue using simulations. Specifically, we study the following questions:

- *What is the impact of the key parameters of BitTorrent on the dynamics of the piece exchange?*
- *What kind of relation exists between the entropy and the performance of a torrent?*
- *What kind of relation exists between the structure of the overlay and the performance of a torrent?*

We focus on a flash crowd arrival scenario. It has been observed in [3] that a typical life cycle of a torrent consists of a initial flash crowd followed by a possibly long post flash-crowd period that lasts until the torrent dies.

We consider the case where all peers have similar characteristics in terms of upload and download capacities. While heterogeneity of peer capacity is an important topic, we have decided to focus on the homogeneous case in the context of this work so as to expose the fundamental characteristics of BitTorrent in this context. Note also that while heterogeneity is the rule in the Internet, it might not be the case in a corporate context.

II. BitTorrent Overview and Terminology

Unlike p2p file sharing applications, BitTorrent creates a dedicated torrent (session) per content. A central entity, called a tracker, maintains a list of active clients in the torrent. When a new client wants to join a torrent, it contacts the tracker that returns a set of peers (a.k.a., *peer set*) with whom it should cooperate. There is a maximum size for the peer set (default to 80) and also a maximum size for the number of connections a peer is allowed to establish (default to 40). We term *outgoing connections* the connections locally initiated. Content replication is based on swarming: the file is split in pieces that clients exchange with one another. Two algorithms govern the behavior of peers. First, the choke algorithm that a peer uses to elect the peers to which it is sending data. The set of peers elected through the choke algorithm is called the *active peer set* of a peer. Second, the rarest first algorithm

that controls the pieces a peer will request from another peer that has unchoked it. We adopt the conventional BitTorrent terminology where a client that is downloading the file is called a *leecher* while a client that has completed the download is termed a *seed*.

III. Simulator Overview

Our simulatior runs in rounds where at each round, a peer transfers pieces to all the peers in its active peer set. A round lasts for 10 seconds. Note that 10 seconds is a typical duration between two calls to the choke algorithm in BitTorrent clients. In leecher mode, the choke algorithm is implemented as follows. A leecher sorts the peers from which it receives data based on the rate they offer. It will unchoke (send data to) the first x peers (default $x = 3$). Every three rounds, another peer is chosen at random and joins the active peer set. This is called an *optimistic unchoke*. In practice, during the "optimistic unchoke round", a leecher sends data to one more peer than it does on average.

In seed state, the choke algorithm is similar to the one in leecher state except that peers are sorted according to their receiving rates.

We assume that bottlenecks, if any, are the downlinks/uplinks of the peers. We do not model any delay, jitter, routing or failure effect. This allows us to concentrate on the very mechanisms of BitTorrent while not obscuring the results with complex low level network effects.

Simulation time directly depends on the number of peers in the torrent. With an off-the-shelf machine with a 2 GHz processor and 1 GB of main memory, it takes approximately 10 hours to simulate a torrent with 1000 clients. For larger torrents, the simulation times quickly explodes. However, we believe that results for up to 1000 peers already allow to get good insights on the actual performance of BitTorrent in the Internet, as we argue in the next section.

IV. Reference Scenarios

A. Peers Arrivals

As explained in Section I, the focus of this work is on flash crowd scenarios. We assume that all peers join simultaneously a torrent at round 1 and that there is only one initial seed . We consider two variants of the flash crowd that we term *serial flash crowd* and *random flash crowd*. Assume that peers are numbered from 1 to N, peer 1 being the initial seed. With the serial flash-crowd scenario, peer i is allowed to establish outgoing connections with peers 1 to $i-1$. For the random flash crowd case, peer i is allowed to connect to peer 1 to N, under the constraint of the maximum number of outgoing connections. The serial flash crowd scenario mimics what happens in a real torrent where the tracker returns to a given peer a list of addresses randomly chosen among the peers that have joined so far. The random flash crowd is less realistic. It will serve to illustrate the impact of the structure of the BitTorrent overlay on the overall performance of the protocol in Section VII.

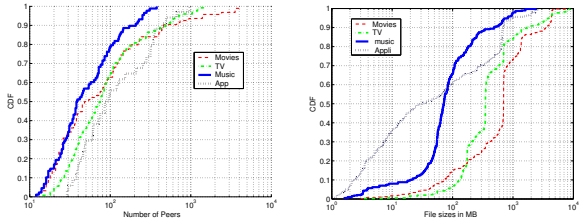
We assume that once a leecher becomes a seed, it remains in the torrent until all downloads are completed. This assumption might not be realistic for the case of torrents in the Internet. Note however that the situation might be different in a corporate context. We have adopted this convention to ease the comparison between different parameter choices for BitTorrent.

B. Clients Capacities

We consider files consisting of pieces of 256 Kbytes. We consider a scenario similar to the one in [2]. The content to be replicated is a file of size 100 Mbytes. The initial seed has upload capacity of 6 Mbits/s. Leechers have an upload capacity of 400 Kbits/s, and download capacity of 1.5 Mbits/s.

C. Torrents Characteristics

Our simulator is optimized to study the performance of torrents with less than a few thousands clients. The reason why we do not focus on larger torrent sizes is that we believe that most torrents in the Internet consist of less than 1000 active clients at a given time instant. To support this claim, we collected data on torrents advertised by the isohunt website. We collected, for four categories of content, namely movies, music, TV and applications, the description of the first 400 torrents of each category (torrents are sorted by age). We filtered out only the torrent with at least 10 leechers and 1 seed. We ended up having around 100 torrents in each category. In Figure 1(a), we plot the distribution of the torrent sizes for each category. We observe from this figure that most of the instantaneous torrent sizes range between a few 10s and 1000 peers. Similar observations on the torrent size distributions have been made in [3]. We also plotted the sizes of the content to be downloaded (this time considering all the torrents, even non active ones). Results are plotted in Figure 1(b). We do observe as expected that movie and TV categories generate the largest content sizes. Overall, most of the mass is located in the range $[10MB, 1GB]$.



(a) Instantaneous torrent sizes (b) Content sizes

Fig. 1.

V. Fundamental Parameters

In this section, we investigate the impact of two key parameters that control the BitTorrent overlay structure. Those parameters are: the peer set size (PS) and the percentage of outgoing connections (OC) that a peer is establishing upon its arrival in the torrent. We also considered the active peer

set size (APS). However, variations observed when tuning this parameter are less significant than when tuning PS and OC. We leave for future work a more in depth study of this parameter.

Our objective here is to investigate how PS and OC affect the performance of the protocol under the constraint of realistic values, i.e. implementable by a BitTorrent client. We consider PS values up to 200. Note that it can already be difficult for a client to be consistently connected to 200 peers, depending on the churn rate of the torrent. As for OC, any value between 0 and 100% is possible. However, we observe that performance worsens when OC is greater than 50%, which is the default value of the protocol. Hence we focus on OC values smaller than 50%.

Simulation results presented in this section are obtained with the serial flash-crowd scenario. To compare the impact on the system of the parameters, we consider two performance metrics: the average time to download the file and the maximum time to download the file. Depending on the context, one of those metrics might have more importance than the other one. For instance, if BitTorrent is to be used in a corporate context, e.g., to patch softwares on a set of machines, the maximum completion time might be the critical parameter. In the Internet, on the other side, one might prefer to consider the average completion time. We apply the classical strategy of varying a single parameter while keeping others constant. For each set of parameters, we perform 10 independent simulations and present results averaged over those 10 sample paths. We present in Table I the results for three different values of the torrent size N, namely $N = 100$, $N = 500$, and $N = 1000$.

We make the following observations from Table I:

1) $PS = 20$ consistently results in the worse performance for all N values, confirming that for the rarest first policy to do a good job, clients must be connected to a large enough number of peers in the torrent;

2) For $N = 100$, results are relatively insensitive to the different parameter values. This is an important result as torrents with up to 100 clients are common in the Internet (see section III).

3) For $N = 500$ and $N = 1000$, optimizing the average download time is (roughly) equivalent to optimizing the maximum download time.

4) Some significant performance improvements are achieved for values that significantly differ from the default ones of most clients.

5) Tuning OC has more impact than tuning PS, as for instance, for $N = 1000$, the best performance are obtained for $OC = 10\%$ rather than $PS = 200$. Note that as we did observe cases where it takes quite a long time for the last client to complete its download (esp. for the case when OC is decreased), we present in Table I, along with the maximum value of the download time, the 99^{th} percentile of the distribution. We leave for future work the investigation of this phenomenon.

Results obtained in this section indicate that an informed choice of the parameters of BitTorrent can lead to a significant performance increase. They also raise the question of the extend to which performance can be increased. Clearly, the optimal performance should be obtained when all peers fully and continuously utilize their upload capacity. We formalize this notion in the next section and investigate the relation between the optimum performance and the entropy of a torrent.

	$N = 100$			$N = 500$		
PS	20	80	200	20	80	200
Mean	178.3	173.6	N/A	277.5	224.9	218.4
p_{99}	221.3	196.3	N/A	320.25	259.0	230.0
Max	221.7	196.6	N/A	320.4	260.6	232.7
OC	10%	25%	50%	10%	25%	50%
Mean	170.8	173.4	173.6	210.4	214.3	224.9
p_{99}	197.4	195.9	196.3	229.4	226.6	259.0
Max	198.1	195.9	196.6	239.3	230.6	260.6
	$N = 1000$					
PS	20	80	200			
Mean	332.5	258.4	232.5			
p_{99}	379.3	300.0	2			
Max	379.8	301.7	253.1			
OC	10%	25%	50%			
Mean	218.3	225.9	258.4			
p_{99}	236.0	235.4	300.0			
Max	254.9	246	301.7			

TABLE I

IMPACT OF FUNDAMENTAL PARAMETERS OF BITTORRENT

VI. PERFORMANCE VS. ENTROPY

Our objective in this section is to introduce a metric of performance for BitTorrent and relate it to the piece replication process of the protocol. Our main finding is that to achieve the best performance, BitTorrent must ensure that all peers have approximately the same number of pieces over time, and pieces are equally replicated in the torrent.

We now introduce the metric we use to depict the dynamics of a torrent and next discuss the results for some key scenarios.

A. Metrics

The first metric we use is the utilization $\mathcal{E}$ of the total upload capacity of a torrent over time. If there are N peers in a torrent with respective upload capacities U_i with $i \in \{1, \ldots, N\}$ and if $R_i(r)$ is the actual upload rate of peer i at round r, then the utilization $\mathcal{E}(r)$ at round r is:

$$\mathcal{E}(r) = \frac{\sum_i R_i(r)}{\sum_i U_i} \tag{1}$$

A typical utilization graph, e.g. Figure 2(a), exhibits 3 different periods:

1) A warm-up period during which peers obtain their first pieces. During this phase, the utilization ramps up from low utilization values to large ones, say 60 to 100%.

2) A central period where all peers in the torrent, except the initial seed, are leechers. If the utilization is consistently equal or close to 100%, this means that performance are optimal as a given peer finds at each round a peer with whom it can exchange pieces at full rate.

3) A termination phase where leechers become seeds. During this phase, the utilization decreases more or less sharply due to the fact that when some leechers become seeds, it is possible that they can't connect to any leecher as all leechers are already connected to exactly PS other peers. We observe that the download time spans on a relatively large or relatively small range of values. The latter case suggests that all leechers evolved similarly during the torrent lifetime and thus receive their last piece at approximately the same time. For this last

condition to hold, it is necessary that pieces are gracefully distributed among peers.

To evaluate how pieces and their replica are distributed over time in a torrent, we introduce an entropy metric consisting of a pair of variables. First, the coefficient of variation[1](CoV) of the cumulative number of pieces obtained by each peer over time (CoV_Peer). Second, the CoV of the number of replica of the pieces that have been uploaded at least once by the initial seed per round (CoV_Replica). Low values for both metrics indicate that the number of replica of all the pieces in the torrent is roughly the same and each peer has roughly the same number of pieces.

The shapes of the CoV_Peer curves (Figures 2(b) and 3(b)) typically consist of initial peaks followed by decaying tails. Initial peaks result from the discrepancy between the upload capacity of the initial seed and the peers in its peer set. Indeed, it takes a significant number of rounds for the peers that receive pieces from the initial seed to redistribute them in their peer set. As time is passing, this discrepancy has less impact as more peers are engaged in the distribution of pieces. As a consequence, the CoV_Peer values decrease until reaching zero when all peers have completed their download of the file.

The shapes of the CoV_Replica curves also exhibit initial peaks due to only a few pieces initially available for replication as the peers in the peer set of the initial seed constitute a bottleneck for the torrent. Similarly to the previous case, this phenomenon vanishes as time is passing and more pieces are available and served by more peers.

B. Reference Scenarios and Results

We consider 4 scenarios:

- Scenario BS (Baseline): serial flash crowd, PS=80, OC=50%;
- Scenario OC (Small OC): serial flash crowd, PS=80, OC=10%;
- Scenario PS (Small/Large PS): serial flash crowd, PS=8 if N=100 and PS=200 otherwise, OC=50%;
- Scenario RD (Random): random flash crowd, PS=80.

We investigated the performance of those 4 scenarios for $N = 100, 500, 1000$. Due to space limitation, we concentrate here on $N = 100$ and $N = 1000$. The trends in the results for $N = 500$ are similar to the ones for $N = 1000$. Scenario BS is the baseline scenario as the default values of the BitTorrent protocol are used.

We present in Figures 2(a) to 3(c) the results for the above scenarios. We make the following observations from those figures:

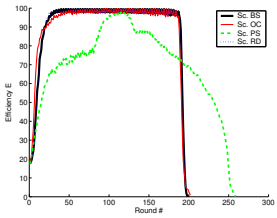
1) For $N = 100$, Scenarios BS, OC and RD exhibit similar behaviors w.r.t the $\mathcal{E}$, CoV_Peer and CoV_Replica metrics. Scenario PS with a small PS value of 20, is inefficient in the central period and hence leads to longer completion times. Note that it also results in larger CoV_Peer and CoV_Replica metrics values.

[1]the coefficient of variation is the ratio of the standard deviation and the mean of a distribution. Roughly speaking, it expresses the variability of a distribution in mean unit.

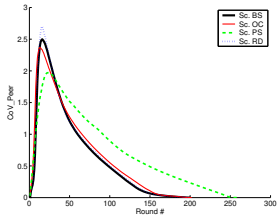
2) For $N = 1000$, there is a direct relationship between achieving an utilization consistently close to 100% and maximizing the entropy (i.e. minimizing the CoV values). This means that to achieve the best peformance, BitTorrent must ensure that all peers have approximately the same number of pieces over time and pieces must be equally replicated in the torrent. If these conditions are (roughly) consistently met over time, all peers should complete their download close to each other in time. This is confirmed by Figure 3(a), where we observe short and sharp termination phases for Scenarios PS and OC, as compared to Scenario BS.

3) Scenario RD, where the default values of the protocol are kept but the technique to build the overlay differs, achieves performance close to the ones of Scenario OC.

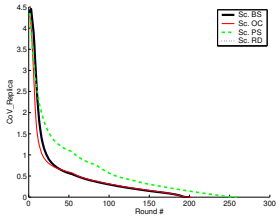
As for the last result, we further note that for $N = 1000$, Scenario RD apparently performs slightly worse than Scenario OC during the warm-up period and, as a consequence, offer slightly larger completion times. Study of the entropy (CoV_Peer and CoV_Replica curves) does not reveal significant differences between those two scenarios. As the major difference between them lies in the structure of their overlay, we further investigate the relation between performance and overlay structure in the next section.



(a) N=100 - Utilization



(b) N=100 - CoV of the number of pieces per peer



(c) N=100 - CoV of the number of replica of per pieces

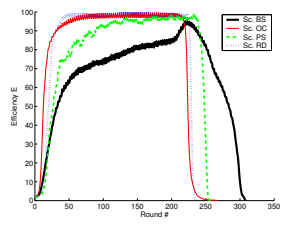
Fig. 2.

VII. Performance vs. Overlay Structure

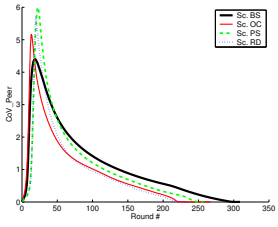
We focus in this section on the relation between the performance of BitTorrent and the structure of its overlay.

As observed in Section VI, Scenarios OC and RD differ mostly in their warm-up phases. During this phase, Scenario OC is faster than Scenario RD. In addition, both scenarios are faster than Scenario PS. They also outperform Scenario PS during the central phase (see Figure 3(a)).

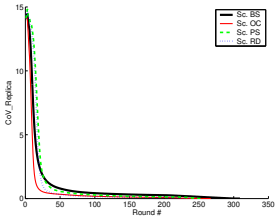
The main result of this section is that the structure and, to a lesser extend, the distances among peers in the overlay,



(a) N=1000 - Utilization



(b) N=1000 - CoV of the number of pieces per peer



(c) N=1000 - CoV of the number of replica per pieces

Fig. 3.

directly impact the performance of BitTorrent.

Let us first focus on the warm up period. At first sight, the warm-up period appear to be a function of the distance of the peers to the initial seed, where the distance is defined as the minimum number of hops from peer to peer to reach the seed. Peers directly connected to the initial seed are at distance 1. Peers directly connected to peers at distance 1 are either at distance 2 or at distance 1 (as 2 peers connected to the seed might be in each other peer set as well), etc. Intuitively, the closer peers are to the initial seed, the sooner they receive their first piece and thus start uploading content.

A. Distances to the Initial Seed

We present in Table II the average and maximum distances for Scenarios OC, PS and RD and $N = 100, 500, 1000$. We first remark that the smallest values for those metrics are often obtained with Scenario RD. This is because, with this model, the seed is connected to PS peers, each of those peers being connected to PS other peers chosen at random among the remaining $(N-2)$ peers in the torrent. Hence, up to $PS \times (N-2)$ are at distance 2 from the seed. While collisions (2 peers electing the same peer for their peer set) in the drawings prevent to reach this figure, $PS \times (N-2)$ is so large as compared to N that all peers should be at a distance of at most 2. This is what we observe from Table II.

Scenarios OC and PS achieve in general larger average and maximum distances than RD because they are based on the serial flash crowd model. With this arrival model, peer i is only allowed to connect to peers 1 to $i-1$, which results in larger distances to the seed for the last peers integrated in the overlay, as confirmed by Table II.

We also observe from Table II that for Scenario PS with N=100, the distances are clearly large as compared to the other scenarios. This surely explains the bad performance of this scenario as compared to the other ones (see Figure 2(a)).

The main conclusion we draw from Table II is that while large distances to the initial seed clearly leads to long warm-up periods, small distances are not a sufficient condition to achieve small warm-up periods. Indeed, while Scenario RD consistently offers smaller (though comparable) distances to the seed than Scenario OC, it offers slightly longer warm-up phases. Also, in the case of $N = 500$, Scenario PS has smaller distances than Scenario OC while its warm-up period is always longer, for all values of N. We next focus on the overlay structure to better understand the performance of Scenario OC.

B. Matrix of Connections

We show in Figures 4(a) and 4(b) the matrix of connections for all the peers when $N = 1000$ and Scenario OC and PS respectively. A matrix of connections[2] is a graph with a dot at coordinates (i,j) if peer i is in the peer set of peer j. We see a clear difference between those two matrices. We did not plot the corresponding matrix for the case of Scenario RD. It is similar to the one of Scenario OC though more uniformly shaded due to the randomization technique used to build the overlay.

Scenarios OC and RD have more distinct peers at distance 2 (from the initial seed) than Scenario PS, and this explains their better performance. Peers at distance 2 are important as the initial seed upload pieces to the 3 or 4 peers in its active peer set and the larger the fan out of those peers, the more replication power (upload capacity) the system has to quickly replicate the content. To take a concrete example, imagine that the initial seed is delivering piece p_1 to peer i and piece p_2 to peer j. If there is a significant overlap between the peer sets of i and j, there will be globally less upload capacity for the two pieces. This is what happens with Scenario PS, where the average number of peers at distance 2 is 284.5. In contrast, with Scenario OC (resp. RD), we obtain 497.6 (resp. 920) peers at distance 2.

The last point to explain is why Scenario OC slightly outperforms Scenario PS during the warm-up phase. The main difference between an overlay obtained with Scenario OC and an overlay obtained with Scenario PS is that with Scenario OC, the peers connected to the initial seed are more likely to be in each other peer sets. Hence those peers are less likely to query the same piece from the initial seed. Downloading the same piece many times from the seed is inefficient, especially if the initial seed has not yet uploaded at least one copy of each piece of the file. Note that even if there is a lot of blocks as compared to the number of peers, the birthday paradox (see, e.g., [1], p. 32) tells us that the probability of collision (two peers choosing the same piece) is not negligible.

VIII. Conclusion

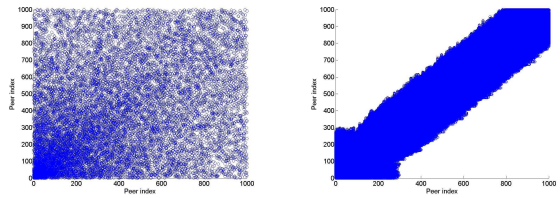
The simulation approach used this paper has revealed that the size of the peer set (PS) and the fraction of outgoing

[2]This matrix can be considered as static as we hypothesize in our simulations that peers stay in the torrent until all peers complete their download. Only at the end of a simulation can some connections be broken and some other established as some peers become seed, disconnect from the seeds they are connected to and might look for other leechers to service.

	$N = 100$		$N = 500$	
	Mean(d)	Max(d)	Mean(d)	Max(d)
Sc.OC	1.7	2.5	2.1	3
Sc.PS	5.7	10.9	1.6	2.8
Sc.RD	1.2	2	1.8	2
	$N = 1000$			
	Mean(d)	Max(d)		
Sc.OC	2.4	3.5		
Sc.PS	2.6	5		
Sc.RD	1.9	2		

TABLE II

DISTANCES TO INITIAL SEED



(a) Matrix of connections - Scenario OC, N=1000

(b) Matrix of connections - Scenario PS, N=1000

Fig. 4.

connections that a peer is allowed to establish (OC) significantly impact the performance of a torrent. Delving into the actual piece transfer, we have highlighted the relation between optimizing the performance and maximizing the entropy of a torrent. An in-depth comparison between PS and OC has revealed that decreasing OC is more efficient than increasing PS. To the best of our knowledge, this work is the first one to shed light on the crucial impact of the OC parameter. We have further demonstrated that the structure of the overlay created by BitTorrent is directly impacted by the choice of the parameters; and that it is the actual difference between the overlay structure obtained when tuning PS or OC that explains why the latter outperforms the former.

We also observed that the best combinations of parameters result in optimal utilizations (consistently close to 100%) of the capacities of all peers in the torrent. This is achieved without any change to the piece selection algorithm. Especially, we saw little need in our simulations for a smart-seed technique, as advocated in [2].

As future work, we intend to study heterogeneous scenarios. We would like also to consider the case of selfish users that leave the torrent as soon as their download is completed. We note that the simulation results that we have obtained already provide a partial answer to this issue. Indeed, as the best performance are obtained for cases where the entropy is maximized, early departure of seeds should not be too harmful as replicas of pieces are gracefully distributed among the leechers that remain in the torrent.

REFERENCES

[1] A. O. Allen, *Probability, statistics, and queueing theory with computer science applications*, Academic Press Professional, Inc., San Diego, CA, USA, 1990.

[2] A. Bharambe, C. Herley, and V. N. Padmanabhan, "Analyzing and Improving BitTorrent Performance", In *Proc. Infocom 2006*, April 2006.

[3] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang, "Measurements, Analysis, and Modeling of BitTorrent-like Systems", In *Proceedings of the ACM/SIGCOMM Internet Measurement Conference (IMC-05)*, 2005.

[4] M. Izal, G. Urvoy-Keller, E. Biersack, P. Felber, A. Al Hamra, and L. Garcés-Erice, "Dissecting BitTorrent: Five Months in a Torrent's Lifetime", In *Passive and Active Measurements 2004*, April 2004.

[5] A. Legout, G. Urvoy-Keller, and P. Michiardi, "Understanding BitTorrent: An Experimental Perspective", , Technical Report (inria-00000156, version 2 - 19 July 2005), INRIA, Sophia Antipolis,, July 2005.

[6] L. Massoulié; and M. Vojnovic;, "Coupon replication systems", In *SIGMETRICS '05*, pp. 2–13, New York, NY, USA, 2005, ACM Press.

[7] J. Pouwelse et al., "The Bittorrent P2P File-sharing System: Measurements and Analysis", In *Proc. IPTPS*, February 2005.

[8] D. Qiu and S. Srikant, "Modeling and Performance Analysis of BitTorrent-Like Perr-to-Peer networks", In *Proc. SIGCOMM 2004*, August 2004.

[9] X. Yang and G. de Veciana, "Service Capacity of Peer-to-Peer Networks", In *Proc. Infocom 2004*, March 2004.