Heterogeneous download times in a homogeneous BitTorrent swarm

Fabricio Murai[a], Antonio A. de A. Rocha[b,*],
Daniel R. Figueiredo[a], Edmundo A. de Souza e Silva[a]

[a] *COPPE/Systems Engineering and Computer Science Program,*
Federal University of Rio de Janeiro,
Rio de Janeiro, Brazil
{fabricio,daniel,edmundo}@land.ufrj.br

[b] *Computer Science Department,*
Fluminense Federal University,
Niterói, Brazil
arocha@ic.uff.br

Abstract

Modeling and understanding BitTorrent (BT) dynamics is a recurrent research topic mainly due to its high complexity and tremendous practical efficiency. Over the years, different models have uncovered various phenomena exhibited by the system, many of which have direct impact on its performance. In this paper we identify and characterize a phenomenon that has not been previously observed: homogeneous peers (with respect to their upload capacities) experience heterogeneous download times. This behavior has direct impact on peer and system performance, such as high variability of download times, unfairness with respect to peer arrival order, bursty departures and content synchronization. Detailed packet-level simulations and prototype-based experiments on the Internet were performed to characterize this phenomenon. We also develop a mathematical model that accurately predicts the heterogeneous download rates of the homogeneous peers as a function of their content. In addition, we apply the model to calculate lower and upper bounds to the number of departures that occur in a burst. The heterogeneous download rates are more prevalent in unpopular swarms (very few peers). Although few works have addressed this kind of swarm, these by far represent the most common type of swarm in BT.

Keywords: Peer-to-peer, BitTorrent, Performance evaluation, Modeling

*Fluminense Federal University - Computing Institute
Rua Passo da Patria, 156, Bloco E, sala 501 - Niterói, Brazil - 24020-240
phone: +55(21)25628788 / fax: +55(21)25628789

1. Introduction

Peer-to-peer (P2P) applications have widely been used for content recovery in Internet. Among them, BitTorrent (BT) [1] is one of the most popular, used by millions daily to retrieve millions of files (movies, TV series, music, etc), accounting for large fractions of today's Internet traffic [2]. The mainstream success of BT is closely related to its performance (e.g., fast download times) and together with its high complexity, has triggered the interest of researchers.

Understanding and characterizing the performance of BT through mathematical models has been an active topic of research [3]. Several studies have uncovered peculiar aspects BT's dynamic, many of which have direct impact on system performance. Moreover, models that capture user and system performance under homogeneous and heterogeneous peer population (with respect to their upload capacities) have been proposed for various scenarios [4, 5, 6, 7]. However, most proposed models target large-scale systems, either with a large and fixed initial peer population or relatively high peer arrival rates.

We consider a BT swarm where all peers have identical upload capacities but unconstrained (or large) download capacities. In this context, we identify and characterize a phenomenon that has not been previously observed: homogeneous peers experience heterogeneous download rates. Although this is expected in swarms where peers have different capacities, in homogeneous swarms, peers should, at first, exhibit similar average performance. Thus, we focus in the latter type of swarm, for which the described behavior has not been captured by any prior model (to the best of our knowledge). Moreover, this observation has several important implications, such as high variability of download times, unfairness with respect to peer arrival order, bursty departures and content synchronization among the peers. Two peers are said to be content-synchronized after their content become identical at a given instant. This last consequence is particularly critical since it is closely related to the missing piece syndrome [8, 9, 10], a scenario where a very large number of peers have all except a single missing piece.

We characterize the fact that homogeneous peers experience heterogeneous download rates and its various consequences by using detailed packet-level simulations and prototype-based experiments on the Internet. To underpin critical parameters for this behavior, we consider various scenarios. We show that peer arrival times strongly influence their average download rate. We also develop a mathematical model that explains the phenomenon and predicts the heterogeneous download rates of the homogeneous peers as a function of their content. The comparison of model predictions with simulation results indicate the model is quite accurate. More importantly, the model sheds light on the key insight for this behavior: upload capacity allocation of peers in BT depends fundamentally on piece interest relationship, which for unpopular swarms can be rather asymmetric. We also apply the model to calculate lower and upper bounds to the number of departures that occur in a burst.

Remark: The case for unpopular swarms with seeds

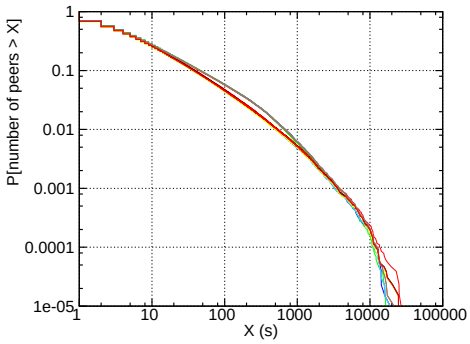
The phenomenon we identify is more prevalent in swarms that have a very small peer

population and a single seed (peer with entire content) with limited bandwidth. However, this is by far the most prevalent kind of swarm in BT, as observed by different and independent measurement studies. In particular, it has been shown that inter-arrival times of peers into swarms increase exponentially with the age of the swarm [11, 12]. Thus, some time after it has been created, swarms receive few peers and therefore have a very small size. A detailed measurement study of swarm sizes in BT considering various repositories and various media types has also recently appeared in the literature [13]. Their results indicate that 70% of active swarms from different repositories have less than 10 peers (Figure 2 in [13]). When considering swarms that do not change size over a relative short time, 97% of them have less than 5 peers (Figure 3 in [13]).

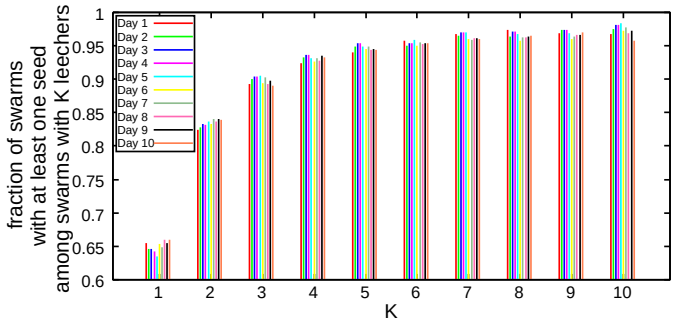
We have also conducted measurements in Torlock.com, one of the most popular Torrent Search Engines available in the Internet nowadays. In particular, we collected information concerning *swarm health* (number of peers, number of seeds, etc) on all available swarms in the website (around 150,000) once a day for ten consecutive days in November 2011. Each swarm has a size which is given by the number of peers connected to the swarm (seeders plus leechers) at the time data was collected. Figure 1a shows the empirical complementary cumulative distribution of swarm sizes for all ten days, considering only swarms that have at least one seed (around 130,000 swarms). Interestingly, swarm size distribution is heavy tailed, with some swarms having a size 1000 times larger than the average. Moreover, most swarms are very small: about 58% of the swarms have less than 5 peers and about 73% have less than 10 peers. Finally, this observation is persistent and consistent over the ten measurement days, indicating that small swarms are very prevalent in BT. Intuitively, swarms without any seeds are not likely to exist in BT since the content may not be fully available in them. Figure 1b shows the fraction of swarms of size K with at least one seed. As expected, the fraction of swarms with at least one seed is very large, more than 90% for all swarm sizes greater than 2. Moreover, as the size of the swarm increases, this fraction also increases. Again, we observe that this is consistent over the ten measurement days, indicating that swarms with at least one seed are very frequent, even when considering unpopular swarms, with sizes less than 5.

Finally, as supported by experimental evidence, unpopular swarms (swarms of very small sizes, e.g., five or less peers) with at least one seed are very common in the real world. Thus, they are the focus point of this paper, although we will present and discuss some generalizations.

The rest of this paper is organized as follows. In §2 we present a brief overview of BT and motivate the phenomenon we have identified. In §3 we characterize the phenomenon and its consequences using simulations and experiments with a real BT application. §4 presents our mathematical model, its validation in comparison with simulations, and some model generalizations. In §5 we apply the model to make predictions about bursty departures. We include a discussion and possible model extensions as well as present some related work in §6 and §7, respectively. Finally, we conclude the paper in §8.



(a) Empirical complementary cumulative distribution of swarm sizes with at least one seed for different days (each curve corresponds to a day).



(b) Fraction of swarms of size K with at least one seed in Torlock.com for different days and swarm sizes (each bar corresponds to a day).

Figure 1: Distribution of swarm sizes and fraction of swarms with at least one seed in Torlock.com for ten consecutive measurement days.

2. BT overview and the observed behavior

2.1. Brief BT overview

BT is a swarm based file sharing P2P application. Swarm is a set of users (peers) interested in downloading and/or sharing the same content (a single or a bundle of files). The content is chopped into pieces (chunks) which are exchanged among peers connected to the swarm. The entities in a swarm may be of three different types: (i) the seeds which are peers that have a complete copy of the content and are still connected to the system altruistically uploading data to other peers; (ii) the leechers which are peers that have not yet fully recovered the content and are actively downloading and simultaneously uploading the chunks; and, (iii) the tracker which is a kind of swarm coordinator, it keeps track of the leechers and seeds connected to the swarm.

Periodically, the tracker distributes lists with a random subset of peers connected to the swarm to promote the interaction among participating peers. In a first interaction, two peers exchange their bitmaps (a list of all file chunks they have downloaded). All updates in their bitmaps are reported by the leecher to its neighbors.

In order to receive new chunks, the leecher must send "Interested" messages to all peers that announced to have the wanted pieces in their respective bitmaps. Because of the rarest first approach specified in BT protocol, leechers prioritize to download first the chunks that are scarcer in the swarm. Once a sub-piece of any chunk is received, the "strict priority" policy defines that the remaining sub-pieces from that particular chunk must be requested before starting the download of any other chunk.

Whenever an "Interested" message is received, peers have to decide whether to "unchoke"

that leecher and serve the piece or to "choke" the peer and ignore the request. Leechers preferentially upload content to other leechers that reciprocate likewise, it is based on a "tit-for-tat" incentive strategy defined by BT's protocol. More precisely, a major fraction of its bandwidth is allocated to serve the peers that have contributed the most to the leecher. However, a minor fraction of its bandwidth must be dedicated to altruistically serve leechers that have never reciprocated. This policy, referred to as "optimistic unchoke", is useful for leechers to bootstrap new reciprocity relationships. As the seeds do not reciprocate, they adopt the "optimistic unchoke" approach all the time. These BT policies were designed with the main purpose of giving all leechers a "fair share" of bandwidth. It means that peers uploading in higher rates will receive in higher download rates, and in a population of leechers uploading at the same rate, they all must reach equal download rates.

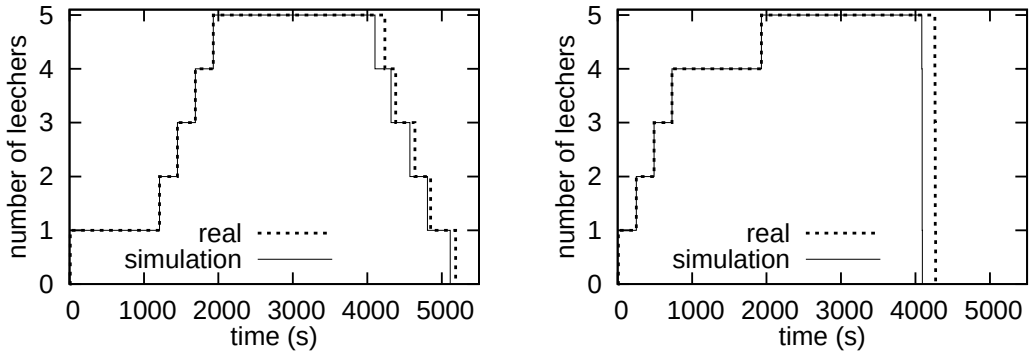
2.2. The observed behavior

Having presented BT's mechanisms, we now illustrate the heterogeneous download rate phenomenon and its consequences with two simple examples. Consider a swarm formed by a seed and 5 leechers. All peers, including the single seed, have identical upload capacity (64 kBps), but large (unconstrained) download capacity. The leechers download a file containing 1000 pieces (256 MB) and exit the swarm immediately after download completion. The seed never leaves the swarm. This system was evaluated using an instrumented implementation of the BitTorrent mainline 4.0.2 client (also used in [14]) running on PlanetLab as well as a detailed packet-level simulator of BT. Both the PlanetLab experiments and the simulations use fully functional BT clients that implement all BT control messages and policies, including peer selection strategies: TFT, optimistic unchoke; and piece selection modes: random-first, rarest-first, strict priority.

The simulation model was developed in the modeling tool Tangram-II [15] (open source and publicly available software). The model we developed is very detailed and faithfully implements the protocol of the BitTorrent mainline 4.0.2 client, including all control messages and policies. In accordance with Tangram-II's modeling paradigm, entities that participate in the system are implemented as separate objects that communicate by message passing. Thus, peers (leechers and seed) and tracker are represented by objects that can be fully parametrized (upload rate, file size, seed after download, etc).

In the following simulations and experiments, leechers start to join the swarm only after the seed is connected and they leave immediately after finishing the download. The simulation/experiment ends when the last leecher leaves. Figures 2a and 2b show the evolution of the swarm size as a function of time for both simulation and experimental results and two different leecher arrival patterns. In Figure 2a, peers leave the swarm in the order they arrived (i.e., FIFO) and have a relatively similar download time. Thus, the download time is relatively indifferent to arrival order (with the exception of the first peer).

Figure 2b shows the same metric just for different arrival times (in fact, the inter-arrival times of peers are also mostly preserved). Surprisingly, an unexpected behavior can be observed



(a) Arrival intervals: 10 min, 4 min, 4 min, 4 min.

(b) Arrival intervals: 4 min, 4 min, 4 min, 10 min.

Figure 2: Evolution of the number of leechers in the swarm.

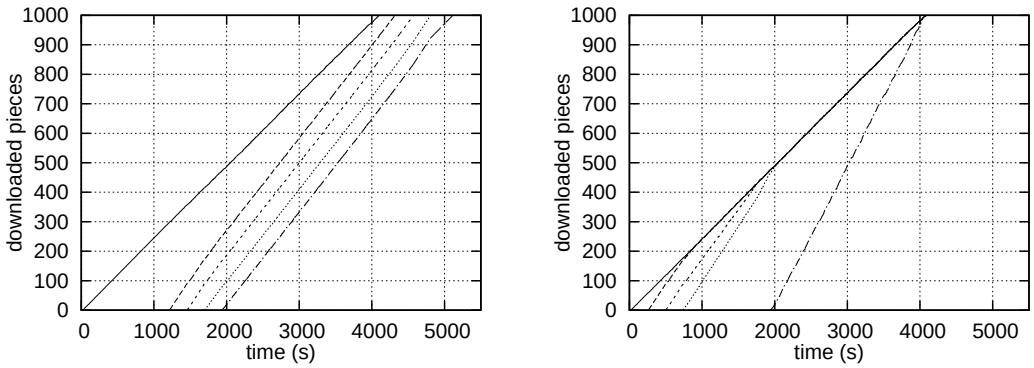
in the system dynamics: despite the significant difference on arrival times, all five leechers completed their respective download nearly at the same time. The time inter departures is small comparing to the download time, which characterizes bursty departures. It means that peers that arrive later to the swarm have a smaller download time. In fact, the fifth peer completed the download in about half the time of the first leecher. Thus, the system is quite unfair with respect to the arrival order of leechers, with late arrivals being significantly favored. What is happening? Why does BT exhibit such dynamics? We answer these questions in the next sections.

3. Heterogeneity in homogeneous BT swarms

In order to understand the behavior exhibited by BT in Figures 2a and 2b, we will analyze the total number of pieces each leecher has downloaded over time. Consider Figures 3a and 3b where each curve indicates the total number of pieces downloaded by a given peer for the corresponding scenario in Figures 2a and 2b, respectively. Note that the slope of each curve corresponds to respective leecher's download rate.

We start by considering Figure 3a. Despite the slope of the first leecher being smaller than that of the remaining peers, the curves never meet. In particular, a leecher finishes the download (and leaves the swarm) before the next leecher reaches the number of blocks it has. We also note that all other leechers have very similar slopes. In addition, we observe a peculiar behavior: the slope of the fifth leecher suddenly decreases when it becomes the single leecher in the system.

The results illustrated in Figure 3b which correspond to the scenario considered in Figure 2b show a very different behavior. Several interesting observations can be drawn from this figure. The slope of the first peer is practically constant, remaining unchanged by the arrival of other



(a) Corresponding to Figure 2a. (b) Corresponding to Figure 2b.

Figure 3: Evolution of the number of downloaded pieces.

peers. The slope of all other peers is larger than that of the first peer, meaning the curves may eventually meet. When two curves meet, the corresponding leechers have the same number of blocks and possibly the same content (we will comment on this point in the following section). The figure also shows that a younger peer does not overcome the first peer, but instead the two maintain the same number of downloaded pieces, possibly with their contents synchronized. Finally, the slope of the second, third and fourth peer are rather similar. However, the slope of the fifth peer is slightly larger than the others, meaning a higher download rate and consequently smaller download time.

In summary, we make the following general observations:

- The first leecher downloads approximately at constant rate.
- Subsequent leechers download at a faster rate than the first.
- Once a leecher reaches the total number of pieces downloaded by the first leecher, their download rates are identical.
- The greater is the number of leechers with the same number of pieces of the first leecher, the higher is the download rate of the other leechers.

All these observations are related to the dynamics of BT and will be discussed and explained in Section 4 using a simple mathematical model. In the remainder of this section, we discuss the consequences of the observed phenomenon and illustrate that it happens even when peer arrival is random (i.e., Poisson process).

3.1. Consequences of heterogeneity in homogeneous swarms

Despite the homogeneous upload capacity of peers, the observations above lead to the following consequences:

- **Variability in download times.** Since peers can experience a consistently different download rate, their download times can also differ.
- **Unfairness with respect to peer arrival order.** Since peers download rates, and thus download times, may depend on their arrival order, the system is inherently unfair, potentially benefiting latecomers in a swarm.
- **Content synchronization.** Due to different download rates and BT's piece selection mechanisms (most notably rarest-first), leechers can synchronize on the number of pieces they have and, more strongly, on the content itself. This means that peers may end up with exactly the same content at some instant, despite arriving at different points in time.
- **Bursty departures.** A direct consequence of content synchronization is bursty departures. This means that peers tend to leave the swarm within a small interval of time despite arriving at the swarm at relatively far apart instants.

Although the figures do not show the content synchronization explicitly, since the first leecher is downloading the file at the same rate at which the seed pushes new pieces into the swarm (seed upload capacity), whenever a leecher reaches the same number of pieces than it, they have exactly the same content.

Of course, the prevalence of the phenomenon and its consequences depend directly on the parameters of the swarm. In particular, the arrival times of peers is certainly the most determinant. However, parameters like upload capacity of seed and leechers and number of pieces are also fundamentally important. Intuitively, a file with a larger number of pieces or a seed with a lower upload capacity increase the probability that the consequences above occur. In fact, for any arrival order of a small set of peers, one can always find system parameters for which this behavior and its consequences occur.

3.2. Heterogeneity under Poisson arrivals

The behavior above does not require deterministic arrivals or any crafted leecher arrival pattern. It arises even when arrival patterns are random. In this section we characterize the consequences of the heterogeneous download rates phenomenon under Poisson arrivals.

We conducted a large amount of evaluations using detailed packet-level simulations. In particular, we consider a BT swarm where a single seed is present at all times, while leechers arrive according to a Poisson process and depart the swarm as soon as their download is completed. In the evaluation that follows, all leechers have the same upload capacity of 64 kBps (and very large download capacities) and download a file with 1000 pieces. The upload capacity of the seed (c_s) varies between 48 kBps, 64 kBps, and 96 kBps, and the leecher arrival rate (λ) is 1/1000 s. These scenarios generate a swarm that has a time average size of 3.7, 3.4 and 3.0 leechers, respectively.

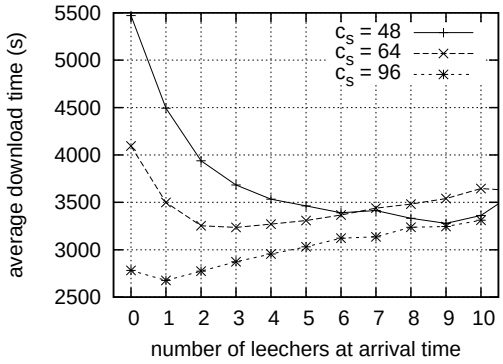
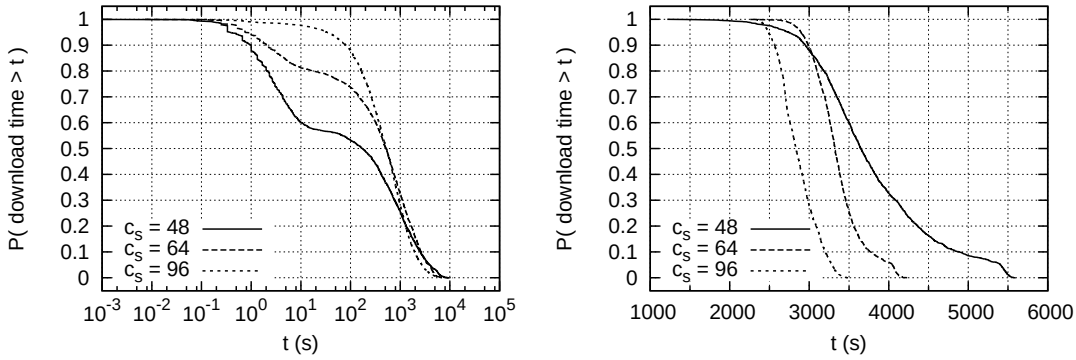


Figure 4: Average download time as a function of arrival order in a busy period.

We start by characterizing the variability in the download times and the unfairness with respect to leecher arrival order. Figure 4 illustrates the average download time for a peer as a function of the number of leechers in the swarm at its arrival time. Thus, if a peer joins the swarm when i leechers are present, it is mapped to index i. The different curves correspond to different upload capacities of the seed. The results clearly indicate that the download time depends on leecher arrival order. In particular, for the case $c_s = 64$ kBps, the average download time tends to decrease with increasing arrival order, and so the first arrival has the largest average download time. Moreover, the download time differences are also significant, and can reach up to 30% (e.g., difference between first and fourth arrival).

Figure 4 also indicates that variability in download times strongly depends on the seed upload capacity. In particular, a fast seed yields the reverse effect: leechers' download times tend to *increase* with arrival order. Intuitively, when a slow seed is present, late arrivals to a busy period obtain large download rates from other leechers, thus exhibiting a lower download time. However, when a fast seed is present, the first leecher has the larger upload capacity of the seed until the second arrival, thus exhibiting a lower download time. The results also illustrate second order effects. For instance, a very late arrival can have an average download time slightly larger (or smaller) than a late arrival (e.g., the sixth leecher arrival has longer download time than fourth for $c_s = 64$ kBps). Intuitively, this occurs because a very late arrival is likely to be alone in the busy period, having to resort to the seed for finishing the download. Since the upload capacity of the seed can be smaller (larger) than the aggregate download rate it receives from other leechers, its download time can increase (decrease). This behavior and its consequences will be explained and captured by the mathematical model presented in the next section.

We now characterize the burstiness in the leecher departure process. Figure 5a shows the empirical CCDF (Complementary Cumulative Distribution Function) of the leecher inter-departure times conditioned on a busy period (i.e., not including the inter-departure time between the last leecher in a busy period and the first leecher of the next). Note that the peer inter-arrival times follow an exponential distribution with rate 1/1000. However, the results indicate a very distinct departure process. In particular, many peers tend to leave the swarm at roughly the same time: up to 30% of peers leave the swarm within a couple of seconds from



(a) Empirical CCDF of peer inter-departure time conditioned on a busy period.

(b) Empirical CCDF of the download time.

Figure 5: Characterization of consequences of heterogeneous download rates for different values of seed capacity.

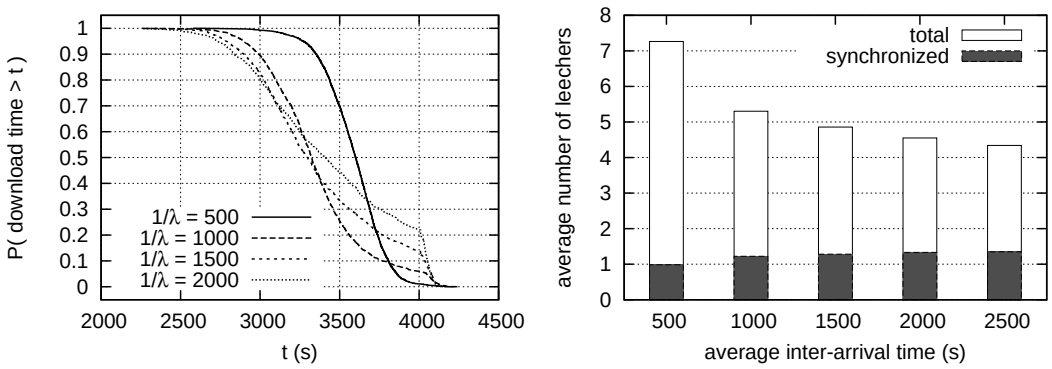
each other when $c_s = 64$ kBps. Moreover, the departure process also exhibits high variability and some peers take as much as ten times more to leave the system after a departure than the average (when $c_s = 64$ kBps). The figure also clearly shows that this observation strongly depends on the seed upload capacity, and is more pronounced when the seed is slow. Intuitively, a slower seed increases the average download time, thus increasing the chances that leechers synchronize their content during the download and depart almost at the same time. Finally, we also note that a fast seed yields a much less bursty departure process, although still favoring short inter-departure times.

Table 1: Average number of leechers and average number of synchronized leechers conditioned on intervals where the number of leechers is greater than 1.

c_s (kBps)	cond. avg. number of leechers	cond. avg. number of synch. leechers
48	4.45	2.40
64	3.86	1.44
96	3.57	0.87

One consequence of the heterogeneous download rates that is closely related to the bursty departures is content synchronization. Here we refer to as synchronized, leechers that are not interested in more than 50 pieces (5% of the file) of any other. In this context, we compare the average number of leechers in the system and the average number of those which are synchronized. These metrics are conditioned on time intervals where the number of leechers is greater than 1, because synchronization is not defined otherwise. Table 1 shows the results of our simulations. The conditional average number of synchronized leechers corresponds to 53.9%, 37.3% and 24.4% for c_s equal to 48, 64 and 96 kBps respectively. While the synchronization is less pronounced when the seed capacity is high, it is very significant when $c_s \leq c_l$.

It is possible to have different download times even when all peers that are simultaneously in the swarm have the same instantaneous download rate. Since peers join the system at



(a) Box plot of download time of leechers.

(b) Conditional average number of leechers and conditional average number of synchronized leechers.

Figure 6: Characterization of consequences of heterogeneous download rates for different values of average inter-arrival time.

different times, they observe the swarm in different sequences of states, in some of which there is more bandwidth available. Those peers will have smaller download times. Nevertheless, as we discussed in Section 3.1, heterogeneous download rates also contribute to the variability in the download times. Figure 5b shows the empirical CCDF of the leecher download time for different values of seed capacity (c_s). While the maximum download time is 45.5% and 52.8% higher than the minimum respectively for c_s equal to 96 and 64 kBps, it is 218.7% higher for $c_s = 48$ kBps. Surprisingly, the minimum download time is the smallest when the seed capacity is the lowest (i.e., 48 kBps). This is because leechers synchronize with high probability under these circumstances and, as we will see in Section 4.2, non-synchronized leechers receive at very high download rates in the presence of many synchronized ones.

We observe that the seed capacity plays an important role on the occurrence of the described consequences under unpopular swarms. In the following, we characterize the impact of another important aspect on these consequences, namely the content popularity, which can be captured through leecher arrival rate. For this purpose, we conducted simulations where the seed and the leechers have the same upload capacity of 64 kBps and average inter-arrival time (i.e, $1/\lambda$) varying between 500, 1000, 1500, 2000 and 2500 s.

We consider the influence of the average inter-arrival time of leechers on the download times, independently of arrival order. Figure 6a shows the empirical CCDF of the download times of peers as a function of the average inter-peer arrival time (i.e., the inverse of arrival rate), for $c_s = 64$ kBps. Note that there are sharp drops for $t > 4000$ which correspond to leechers whose average download rate is approximately equal to c_s. These sharp drops are more pronounced when the inter-arrival time is large. In addition, as the inter-arrival time grows, the 10th-percentile decreases and the 90th-percentile increases, indicating that the download times become less concentrated around the average. However, the variability between minimum and maximum download time does not diminish with the inter-arrival time.

Figure 6b illustrates the intensity of content synchronization for different arrival rates. It shows the average number of leechers in the system and the average number of those which are synchronized. We observe that, the number of synchronized leechers remains practically the same as we increase the inter-peer arrival time, indicating that a larger fraction of peers have similar content when popularity decreases.

As with content synchronization, the fraction of bursty departures is also strongly dependent on the leecher arrival rate. While approximately 5% of the intervals between departures are smaller than 10 seconds for an arrival rate $\lambda = 1/500$, more than 30% of intervals are smaller than 10 for $\lambda = 1/2500$. On the other hand, the unfairness with respect to the arrival order in a busy period is almost insensitive to the leecher arrival rate (considering $1/2500 \leq \lambda \leq 1/500$).

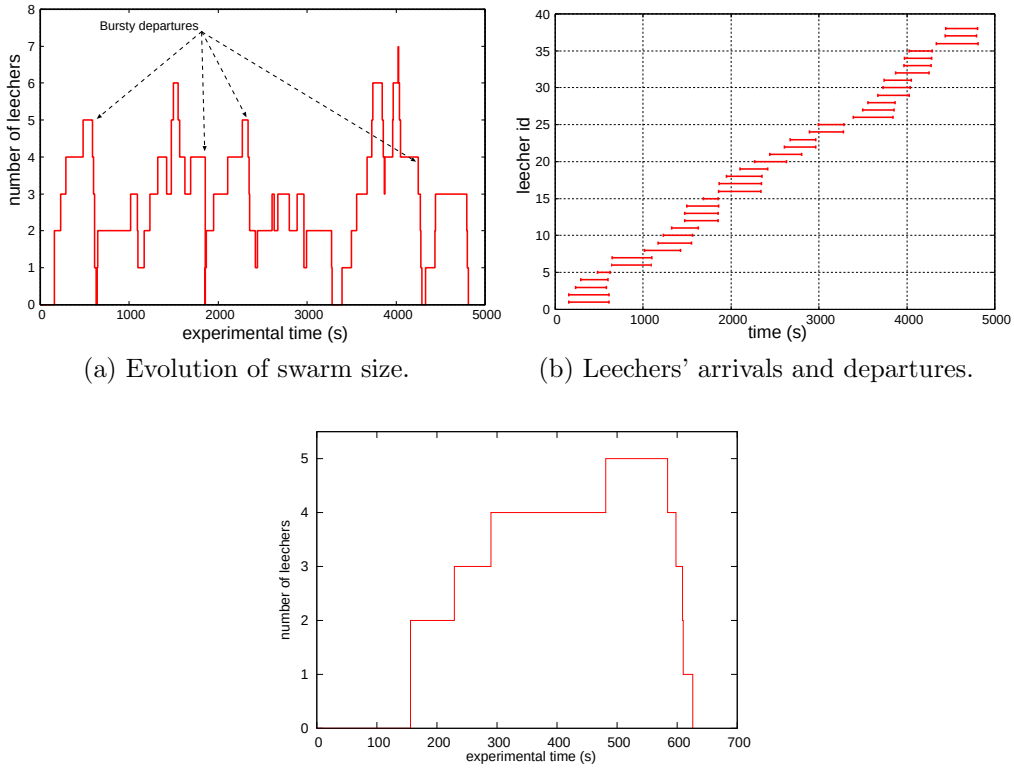
3.3. Real experimental evaluation

The results shown above were all obtained through simulations and we now present results from prototype-based experiments deployed in more realistic scenarios. The real experiments were performed in the Internet using machines from Planetlab[16] and running an instrumented version of a BT client[14]. Although a large number of experiments were conducted, we report only on a limited set of these results due to space constraints. The goal here is to validate the phenomenon of heterogeneity in homogeneous BT swarms and its consequences in real BT application running over the Internet.

In the experiments, the PlanetLAB machines were selected using a quick and simple performance test. Before starting every experiment, a controller dispatches a command via ssh for a set of few hundred machines randomly chosen from the complete list of all PlanetLAB machines. The command line basically makes the machines to download and install all the necessary files (including BT client and scripts) to execute locally the experiment. The set of machines that had the best performance downloading and installing the files was used in the experiments. This performance test was enough to avoid using machines overloaded or connected through congested links.

We consider only private swarms in the experiment, in the sense that only peers controlled by the experiment can connect to the swarm for uploading and downloading content. Each private swarm consists of a single file of size S MB which is owned by a single seed that is always available and has upload capacity of c_s. Leechers interested in downloading the content arrive to the swarm according to a Poisson process with rate λ. All leechers that arrive to the swarm are homogeneous and have upload capacity equal to c_l. The maximum upload capacities used in the experiments are defined as parameters of any BT client (including the one we use). Note that those upload capacity values used for the experiments were far below the limit imposed for each slice (user) in PlanetLAB. Each experiment run is executed for $t = 5,000$ seconds and leave the swarm once the download is completed.

We start by analyzing the evolution of the swarm size for an unpopular swarm. Figure 7a shows the number of leechers in the swarm over time for the duration of the experiment, with



(a) Evolution of swarm size.

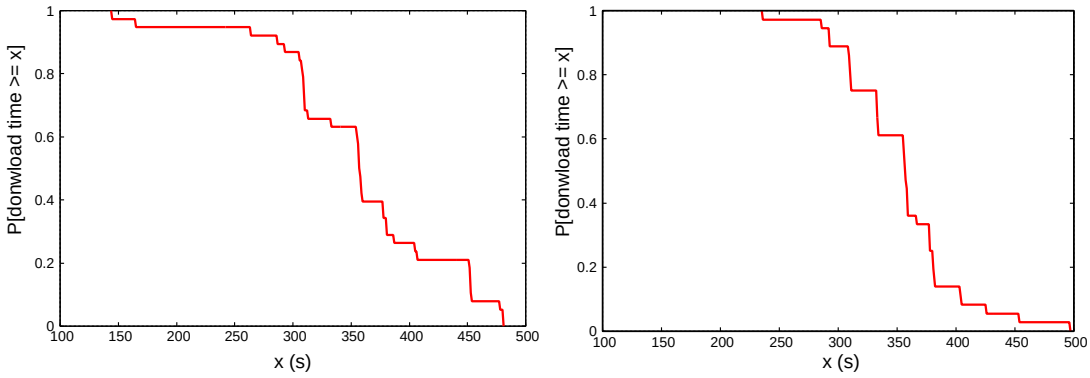
(b) Leechers' arrivals and departures.

(c) Zoom-in of the first busy period.

Figure 7: Swarm dynamics in real experiments.

parameters $\lambda = 1/125$ peers/s, $S = 20$ MB, and $c_s = c_l = 50$ kBps. We can observe several occurrences of bursty departures, even if leechers arrive according to a Poisson process. As previously discussed, bursty departures are consequence of content synchronization among the leechers in the swarm.

Using the same experiment as above, we investigate the impact of the leechers' arrival order on their download times. Figure 7b illustrates the dynamics of the swarm, where each horizontal line corresponds to the lifetime of a leecher in the swarm, starting when the peer arrives and ending when it departs the swarm. Note that peers exhibit significantly different download time (which corresponds to their lifetime in the system). In particular, in many cases leechers arrive at different time instants but depart in the same burst. For instance, the fifth leecher to arrive to the swarm departs in a burst almost together with all four prior arrivals (see Figure 7c for a zoom-in of the first busy period). Thus, the fifth leecher has a much smaller download completion time, when compared to the first leecher. Similar behavior occurs between the fifteenth leecher and the three leechers that arrived immediately before. Besides illustrating the variability of the download times, this observation also indicates the unfairness with respect to leecher arrival order. In particular, late arrivals to a busy period tend to have smaller download times.



(a) $c_s = 50$ kBps and $c_l = 50$ kBps. (b) $c_s = 60$ kBps and $c_l = 50$ kBps.

Figure 8: CCDF of download time from real experiments.

We now focus on the distribution of the leechers' download times to illustrate their relative high variability. Figures 8a and 8b show the complementary cumulative distribution function (CCDF) of download times computed for two experiments with distinct upload capacities for the seed ($c_s = 50$ kBps and $c_s = 60$ kBps, respectively, with all other parameters the same). In both results, download times exhibit a high variance, as shown in the figures. In the case $c_s = 50$ kBps (Figure 8a), the minimum and maximum values are 145 and 480 seconds, respectively, with the maximum being more than three times the minimum. When the upload capacity of the seed is higher than that of the leechers, Figure 8b shows that the variance in download times decreases, as expected, since the system capacity is increased. Finally, we note several discontinuities (i.e., sharp drops) in both CCDF curves which are caused by sets of leechers that have approximately the same download time.

4. Model

We develop a simple model attaining to understand the origin of the heterogeneous download times and its consequences. Our model obtains an approximation to the average upload and download rates observed by each leecher on different time intervals for unpopular swarms.

Consider a homogeneous swarm of some unpopular content with a single seed to which leechers arrive sequentially and depart as soon as they complete their download, such as the one illustrated in Figure 2a. By unpopular content we imply a swarm with an arrival rate that is small enough such that there is never too many peers in the swarm. In particular, our modeling framework assumes that the maximum number of upload connections of peers is always larger than (or equal to) the swarm size. In such scenario, Tit-for-Tat (TFT) and optimistic unchoke algorithms have no effect, since all peers upload to one another. Thus, such mechanisms are not present in our model. However, note that rarest-first mechanism continues to operate since is not affected by this assumption and is therefore captured by our model.

In the described scenario, bursty departures can only happen if younger leechers obtain roughly the same number of pieces as older ones, and leave the swarm at about the same instant. This in turn implies that younger leechers must have higher download rates than older ones, at least for some periods of time. Why is that? At a given moment, an older leecher i may have all pieces owned by a younger leecher j. Thus, leecher's j uplink capacity will be used to serve other leechers until j receives a piece that i does not have. During this period of time, j simply cannot serve i, even if it has no other leecher to serve. Therefore, the sets of pieces owned by each leecher are the root causes for heterogeneous download rates and must be considered.

In order to capture the observation above, each peer, either a seed or a leecher, is represented by a queueing system with multiple queues (see Figure 9), one for each neighbor, under a processor sharing discipline. Queue j of peer i contains the pieces interesting to peer j (i.e., all pieces that i has that j has not). When peer j downloads one of these pieces, from i for instance, this piece is removed from the j-th queue of i, and from the j-th queues of other peers where the piece was present. On the other hand, whenever a peer downloads a piece that other neighbors are interested in, this piece is placed in the queues corresponding to those neighbors, increasing their queues sizes. Finally, the queues of the seed always have all pieces that are needed by the leechers. As a leecher downloads pieces from the seed and other leechers, this queue decreases, eventually becoming empty when the leecher downloads the entire content and departs the swarm. We note that the order at which these pieces are served from these queues depend on the piece selection policy.

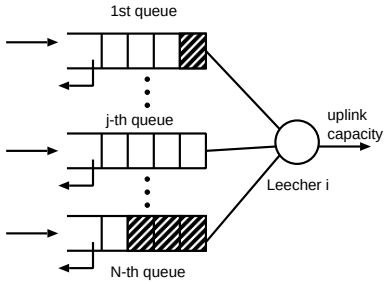


Figure 9: Leecher i can be represented as server with multiple queues, one for each neighbor, containing pieces that are interesting to them.

Let c_s and c_l be the seed and leechers' uplink capacities, respectively. Assume that the leechers' downlink capacities are much larger than c_s and c_l. Let $N(t)$ be the number of leechers in the system at time t. Since the seed always has interesting pieces to every leecher, all the $N(t)$ queues in the seed are backlogged. Thus, all queues will be served at rate $c_s/N(t)$. Note that, since the swarm is unpopular, we assume the swarm size is small enough such that every leecher is neighbor of every other peer (including the seed) and can serve all of them simultaneously.

A leecher may not have interesting pieces to some of its neighbors at time t. Let a leecher be identified by its arrival order, thus leecher i is the i-th leecher to join the swarm. Also let $n_i(t) \leq N(t) - 1$ be the number of leechers interested in pieces owned by i. The instantaneous

upload rate from i to any of these leechers is $c_l/n_i(t)$.

Whether a leecher has or has not pieces interesting to another depends on the leechers' respective *bitmaps*, i.e. the current subsets of pieces owned by a leecher at time t. The set of bitmaps of all leechers would precisely determine the exact pieces in each queue. However, the dynamics of the bitmaps are intricated and to keep track of them would be unnecessarily complicated for modeling the phenomenon we are interested in. Instead, we consider the number of pieces owned by each leecher i, $b_i(t)$, and infer whether a leecher has interesting pieces to other leechers.

For the sake of simplicity, let $b_i(t) = b_i$ and $N(t) = N$. Two remarks can be made with respect to b_i and the interest relationship among leechers:

Remark 1. *If $b_i > b_j$, then i has at least $b_i - b_j$ interesting pieces to j.*

Remark 2. *If $0 < b_i \leq b_j$, it is impossible to determine whether i has or has not interesting pieces to j without further information.*

In the following, we will use these two remarks to derive a simple model to capture the upload and download rates between peers. With respect to Remark 2, we will assume that no further information is available, and hence the piece interest relationship among peers will be ignored in this case. Nevertheless, we will see that a peer with less pieces than other can still upload pieces to the latter.

4.1. A simple fluid model

We assume that content is fluid, or equivalently, that pieces can be subdivided in infinitely many parts that can be exchanged (uploaded and downloaded) continuously.

To simplify the explanation, assume that $b_1 > b_2 > \ldots b_N$, i.e. an older leecher has strictly more pieces than a younger one. We will relax this assumption later on this section, allowing the model to represent swarms where two peers arrived at the same time, or more generally, where some leechers have the same number of pieces. We now make the following assumptions:

- Even if leecher i has joined the swarm after j, i.e. $i > j$, i can still upload pieces to j as long as i downloads pieces from any peer k that has more pieces than j, i.e. $k < j$. Thus, younger peers can upload to older peers.
- Every piece downloaded from the seed by a leecher is immediately interesting to all other leechers, independently of their arrival time. The rarest-first piece selection policy provides support for this assumption. Figure 10 depicts the idea that a younger peer can upload pieces to an older one. In this scenario, peer 4 can upload to peer 2, since it is downloading pieces interesting to peer 2 from the seed and from peer 1.

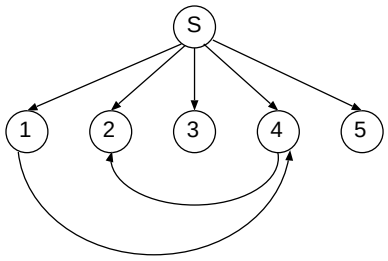


Figure 10: Peer 4 can upload pieces to peer 2, since it is downloading pieces interesting to the latter from the seed and from peer 1.

- Since the seed's upload capacity is c_s, each leecher downloads from it at rate c_s/N. Now let g_{ij} be the rate at which peer i could potentially upload data to peer j provided that there is no capacity constraints (i.e. independently of upload and download capacities of peers i and j, respectively). If a leecher i is older than j, i has interesting pieces to j. Therefore, from the perspective of the multiple queueing system, queue j in leecher i is backlogged and $g_{ij} = \infty$. On the other hand, if i is younger than j, the rate g_{ij} is given by the rate at which i downloads interesting pieces to j.

We draw the reader's attention to the first two assumptions. They account for the upload rate that a younger leecher can sustain to an older leecher, even though we cannot say that the former has interesting pieces to the latter just from the number of pieces they own.

From these assumptions, we can conclude that the rate g_{ij} at which a peer i uploads interesting pieces to an older peer j is equal to the rate at which peers older than j upload to i plus the rate at which i downloads from the seed. We thus have:

$$g_{ij} = \begin{cases} \infty, & \text{if } i < j \quad (1a) \\ \dfrac{c_s}{N} + \displaystyle\sum_{k<j} u_{ki}, & \text{if } i > j \quad (1b) \end{cases}$$

where u_{ki} is the rate at which leecher k uploads to i.

For instance, in Figure 10, $g_{4,2}$ is the sum of the rates at which peer 2 downloads from the seed ($c_s/5$) and from peer 1 ($u_{1,4}$). Hence, $g_{4,2} = c_s/5 + u_{1,4}$. Again we see that the proposed model accounts for the fact that a younger leecher can upload pieces to the older ones. In a real swarm however, peer 4 may upload to peer 2 pieces downloaded from younger leechers as well, such as peer 5. Although the pieces that peer 5 downloads from the seed are immediately interesting to both peers 2 and 4, they will not start and finish downloading this piece from peer 5 at the same time. Thus, leecher 4 may finish first the download of such a piece and then help serve the remaining sub-pieces to peer 2, violating our assumption. Intuitively however, the contribution of peer 4 in uploading this piece to peer 2 is small, since peer 4 must fully finish the download before it can start uploading, by which time peer 2 will have downloaded most of the piece from peer 5. Thus, we claim that such effects are negligible and can be ignored since the model is accurate when compared to simulations and experiments, as discussed in Section 4.2.

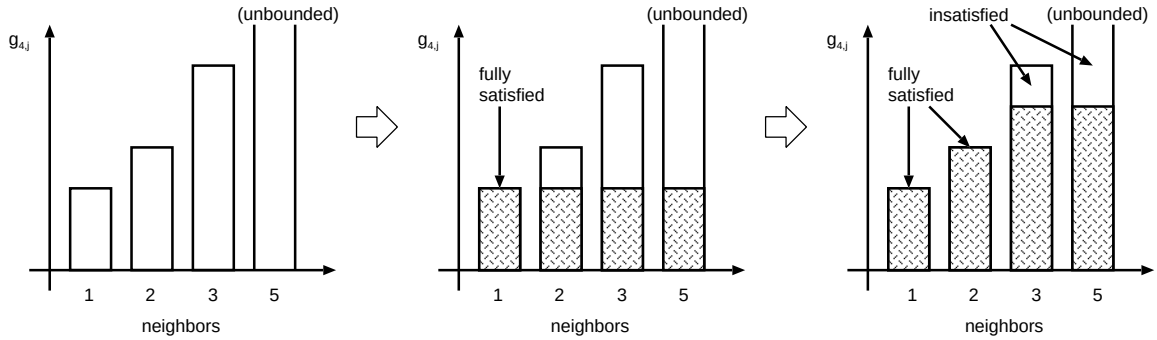


Figure 11: The upload bandwidth allocation of leecher i follows a progressive filling algorithm.

We now make an important observation concerning Equation (1b). Consider leecher i and some other leecher j. The older j is with respect to i the smaller is the rate at which i can upload to j, that is, the smaller is g_{ij}. If j is younger than i, then $g_{ij} = \infty$. This observation implies that $g_{i1} \leq g_{i2} \leq \ldots \leq g_{iN}$.

In addition, note that $g_{ij} > 0$ for all i, j. As we consider a small swarm, all peers upload to one another. Since the upload capacity of peers is finite, we must now determine how the capacity of a given peer i will be divided to serve each of the $N-1$ other leechers. In particular, recall that u_{ij} is the upload rate from peer i to peer j and note that $\sum_k u_{ik} \leq c_l$, where c_l is the upload capacity of a leecher. To determine u_{ij} given the values of g_{ij}, where $1 \leq j \leq N$, we use a bandwidth allocation mechanism that follows a progressive filling algorithm. This mechanism determines the outcome of the processor sharing discipline. Figure 11 illustrates the progressive filling algorithm for the example presented in Figure 10. Roughly, infinitesimal amounts of bandwidth are allocated to each neighbor until (1) the leecher's capacity is completely allocated or (2) a leecher j is satisfied with respect to the g_{ij} constraint. In the former case, the algorithm stops. In the latter, it continues to distribute the remaining capacity among the non-satisfied leechers until one of the two conditions occurs again.

Due to the fact that $g_{i1} \leq g_{i2} \leq \ldots \leq g_{iN}$, the final bandwidth allocation for leecher i can be efficiently obtained by computing the following equation in the order $j = 1, \ldots, N$:

$$u_{ij} = \min\left(g_{ij}, \frac{c_l - \sum_{k<j} u_{ik}}{N - 1 - |\{k|k < j, k \neq i\}|}\right) \tag{2}$$

where $|A|$ is the cardinality of a set A. Now recall from Equation (1b) that g_{ij} depends on $u_{1,i}, u_{2,i}, \ldots, u_{j-1,i}$, for $i > j$. Therefore, by calculating u_{ij} in the order $i = 1, \ldots, N$, we assure that every variable in Equation (2) has been previously computed.

As an example, consider the calculation of the matrix $\mathbf{U} = (u_{ij})$, which determines upload rates between peers at a given moment, for a small swarm containing a single seed and $N = 3$ leechers. Let their upload capacities be equal to $c_s = 60$ kBps and $c_l = 96$ kBps, respectively, and assume $b_1 > b_2 > b_3$. Matrix $\mathbf{U}$ and the order of computation of its elements are depicted in Figure 12. The download rate d_i for peer i is simply c_s/N plus the sum of the elements in

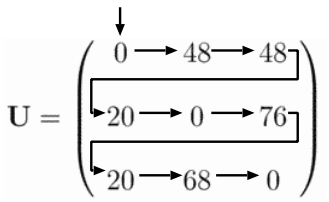


Figure 12: Example of matrix $\mathbf{U} = (u_{ij})$ showing the right order of calculation.

column i:

$$d_i = \frac{c_s}{N} + \sum_{j=1}^{N} u_{ji}. \tag{3}$$

Hence,

$$d_1 = 60/3 + 0 + 20 + 20 = 60 \tag{4}$$
$$d_2 = 60/3 + 48 + 0 + 68 = 136 \tag{5}$$
$$d_3 = 60/3 + 48 + 76 + 0 = 144 \tag{6}$$

Equations (4)-(6) corroborate the idea that homogeneous peers can exhibit heterogeneous download rates which depend on the number of pieces owned by each leecher. Moreover, younger leechers tend to have a higher download rate, as they obtain a higher upload rate from other leechers. This is the opposite of what happens in large swarms, where the older leechers usually manage to keep the TFT for longer periods, hence achieving higher download rates.

Eventually the number of pieces owned by a leecher may reach the number of pieces owned by an older one. In particular, this is bound to occur since younger leechers tend to have a higher download rate. In this case, these two leechers will no longer have pieces interesting to each other. Thus, Equations (1) and (2) must be rewritten as functions of $b_i, \forall i$:

$$g_{ij} = \begin{cases} \infty, & \text{if } b_i > b_j \quad (7a) \\ \dfrac{c_s}{N} + \displaystyle\sum_{b_k > b_j} u_{ki}, & \text{if } b_i \leq b_j \quad (7b) \end{cases}$$

$$u_{ij} = \min\left(g_{ij}, \frac{c_l - \sum_{k|b_k > b_j} u_{ik}}{N - 1 - |\{k|b_k > b_j, k \neq i\}|}\right). \tag{8}$$

Intuitively, Equation (8) combines the two constraints on the rate at which i upload pieces to j. The first term stands for the maximum instantaneous rate irrespective of capacity limitations. The second term reflects the fraction of i's uplink capacity that can be dedicated to j given that some bandwidth has already been allocated. In this case, $c_l - \sum_{k|b_k > b_j} u_{ik}$ is the remaining capacity of i and $N - 1 - |\{k|b_k > b_j, k \neq i\}|$ is the number of peers that will share it (including j). Note that the equations above relax our initial assumption that $b_i, \forall i$ had to be distinct at all times, allowing for leechers to join the system simultaneously or more generally, for leechers to have the same number of pieces.

In Equations (7) and (8), variables N, b_i and b_j change over time, representing arrivals, departures and the acquiral of new pieces. Instead of writing those variables as a function of time, we dropped the t variable for the sake of simplicity. Therefore, these equations can be computed for each time interval by assigning to these variables their corresponding values at that time. However, note that a change in b_i does not necessarily imply in a change in the download rate of leechers as what matters is the relationship between b_i and b_j, for all i, j. Thus, as the system evolves the variables that govern the equations will change value, but not the equations themselves, which can be used to compute the current download rate of leechers given the state of the swarm.

We will see in Section 4.2 that the proposed model given by Equations (7) and (8) yields accurate results for unpopular swarms, indicating that for this scenario it is sufficient to know the number of pieces each peer possesses. Nevertheless, we further discuss two useful generalizations to this model in Section 4.3.

4.2. Model Validation

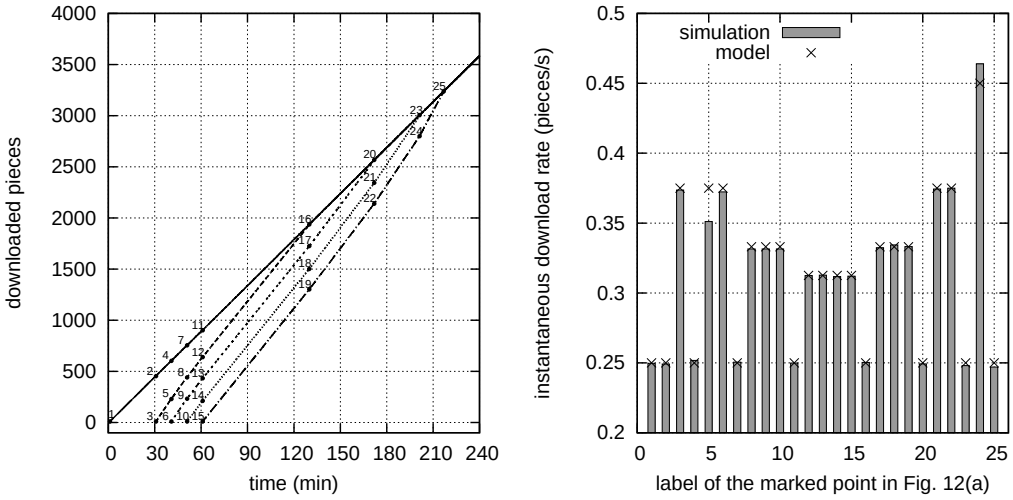
Our model gives an approximation to the average download rate experienced by a leecher in a unpopular swarm, which depends on the relationship between the number of pieces owned by the peers and upload capacities. In this section, we validate the model comparing its predictions with simulations results. We will see that even though the model does not take the TFT and other mechanism into account, its results are very similar to those obtained from our simulator, which implements a fully functional version of the BT protocol (see simulator description in Section 2.2).

We consider homogeneous swarms with $c_s = c_l = 64$ kBps, where exactly $N = 5$ arrivals occur. In addition, all leechers arrive before the first one completes the download and all the arrivals occur before any two leechers synchronize their contents. In our simulations, we say that two leechers i and j are synchronized if they have roughly the same number of pieces, i.e., $|b_i - b_j| < 3$. We use deterministic arrivals to reproduce the exact scenarios we intend to compare.

Consider the evolution of number of downloaded pieces in such a swarm illustrated in Figure 13a. The first leecher arrives at time $t = 0$ and four other leechers join the swarm at $t = 30, 40, 50, 60$. After $t = 120$, leechers start to synchronize. We chose several points from curves in this figure corresponding to instants of time where an event that can change peers' download rates occurs. More precisely, we labeled points in these curves with numbers when new leechers arrive or when two leechers synchronize.

Figure 13b shows peers' download rates from simulations and model for the labeled points indicated in 13a. We have simulated five runs for each scenario including the one depicted in this figure. The confidence intervals obtained are relatively small and are omitted.

The simulation results for points 1, 2, 4, 7, 11, 16, 20, 23 and 25 show approximately the



(a) Evolution of downloaded pieces.

(b) Comparison between simulation and model results.

Figure 13: Model validation and comparison with simulation.

same download rate, what is correctly captured by the model. The download rates obtained from the model are exactly the same due to fact that the corresponding peers already have every piece that was previously pushed by the seed into the swarm. Thus, neighbors of such peer can only upload to it new pieces they receive directly from the seed, i.e., their upload rate is constrained by c_s/N. Since this constraint is below the capacity that can be allocated to serve a neighbor when $c_s = c_l$ (which is, at least $c_s/(N-1)$), every peer in the swarm will upload to one such peer with rate equal to c_s/N. Therefore, the average download rate predicted by the model for peers in A is $c_s/N + (N-1)c_s/N = c_s = 0.25$. In particular, the relative error is less than 1.5% for all these points. The model is quite accurate even for other values of N.

On the other hand, simulation results for the other points exhibit a great variety of download rates. However, those points which correspond to the same moment in time display similar download rates (e.g. 8, 9 and 10). We observe that the download rates decrease with new arrivals. We also note that as more leechers become synchronized, non-synchronized leechers achieve higher download rates (see points 21, 22 and 24). This increase in the download rates occurs because the greater is the number of synchronized peers, the greater is the remaining capacity to serve leechers with less pieces. This is due to the fact that the rate at which synchronized leechers can transmit to each other is very constrained as we discussed before. The relative error of the model is less than 1% for all points, but the 5-th (7%) and the 24-th (3%).

From these figures we conclude that when $c_s = c_l$, at a given moment in time, it is possible to partition the set of leechers in two subsets: leechers with the same number of pieces as the oldest leecher (subset A), and those with less pieces than the oldest one (subset B). When $c_s = c_l$, the model predicts that all leechers in each of these subsets will have identical download rates. Moreover, a leecher in B will have a higher download rate than one in A and this difference

depends on the set sizes. In particular, larger swarms imply lower values of the minimum download rate and higher values of the maximum download for leechers in B. This tendency can be observed both in simulation and model.

Considering all the simulations performed, we conclude that the model is quite accurate, with differences being unnoticeable in most scenarios and less than 10% in all cases. More importantly, the model captures well the trends observed in simulation.

4.3. Model generalizations

Some of the assumptions of the model we propose are: (1) unconstrained (or large) download capacities, and (2) leechers with identical upload capacities. We now relax the former assumption by providing an upper bound for the download rate of a peer. This bound is a function that does not grow fast on the system parameters. Clearly, if the download capacities are greater than this function then all the previously presented results hold.

In what concerns the latter assumption, we indicate how to adapt the model to cope with similar (but not identical) upload capacities. Furthermore, we present some simulation results that show that the general behavior of the system under this scenario is similar to the one presented in Section 3.

4.3.1. Finite (and small) download capacities

When the oldest peers are synchronized, they can only send to each other what they receive directly from the seed (see Equation (7b)). This constraint leads to more capacity available to serve those peers that are not synchronized. In particular, if there is only one non-synchronized peer, it can benefit from this idle bandwidth alone and consequently achieve the highest possible download rate. In what follows we compute an upper bound for this maximum download rate.

Consider an unpopular swarm with $N > 1$ peers, such that the $N-1$ oldest peers are synchronized. From Equation (7) we can compute the maximum instantaneous upload rate of a synchronized leecher i to the other peers irrespective of capacity limitations:

$$g_{ij} = \begin{cases} \infty, & \text{if j is not synchronized} \quad (9a) \\ \dfrac{c_s}{N}, & \text{if j is synchronized} \quad (9b) \end{cases}$$

According to Equation (8), each leecher i will upload to each of the other $N-2$ synchronized peers at rate $\min\{\frac{c_l}{N-1}, \frac{c_s}{N}\}$. The remaining capacity of i that can be used to serve the younger leecher N is $c_l - (N-2)\min\{\frac{c_l}{N-1}, \frac{c_s}{N}\}$. Since there are $N-1$ synchronized leechers, the capacity that can be used to serve only non-synchronized leecher is $(N-1)[c_l - (N-2)\min\{\frac{c_l}{N-1}, \frac{c_s}{N}\}]$. In addition, the younger leecher downloads from the seed at rate $\frac{c_s}{N}$. Therefore, the maximum

download rate is given by

$$d_{\max} = (N-1)c_l - (N-2)\min\{c_l, \frac{c_s(N-1)}{N}\} + \frac{c_s}{N}$$

Thus, we have:

$$d_{\max} = \begin{cases} c_l + \frac{c_s}{N} & \text{if } c_l \leq c_s(N-1)/N \quad (10a) \\ (c_l - c_s)(N-1) + 2c_s - \frac{c_s}{N} & \text{otherwise} \quad (10b) \end{cases}$$

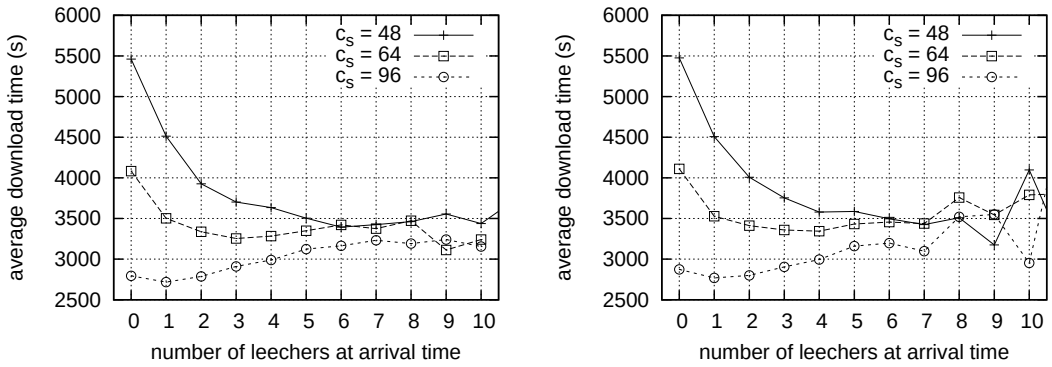
Note that in both cases the maximum download rate is a value that does not grow fast in any of the system parameters. In particular, for small N, which is the case of interest, $d_{\max}$ has a relative small value with respect to the upload capacities or leechers and seed. Thus, if the download capacities of leechers are larger than $d_{\max}$, then results predicted by the model are just as good. This condition replaces the requirement of unbounded (or arbitrarily large) download capacities assumed earlier in the model.

To illustrate, consider the example in Section 4.2 where $c_s = c_l = 64$ kBps and $N = 5$. In this case, the highest download rate would be $d_{\max} = 2 \times 64 + 64/5 = 140.8$ kBps. Thus, if download capacities of leechers are larger than 140.8Kbps, then the results predicted by the model would be unchanged.

4.3.2. Similar but not identical upload capacities

Although we have assumed upload capacities of peers to be identical, this is certainly not necessary for the piece distribution process in unpopular swarms to lead to heterogeneous download rates. Note that our modeling framework allows for peers to have different upload capacities, as c_l could depend on i in Equation 2 (equivalently, in Equation 8). Clearly, this would have an impact on the heterogeneity of the performance and would depend on the values of c_{l_i} and the order of their arrivals to the swarm. However, if $c_{l_i}\, \forall i$ are close to one another, for example, chosen uniformly at random from a small range, then we expect not to see much differences with respect to the constant c_l value.

In order to support this last claim, we repeat the simulations described in Section 3.2 but allowing the upload capacity of leecher i to be drawn uniformly at random from the range $[c(1-\epsilon), c(1+\epsilon)]$, where $c = 64$ kBps and $\epsilon \in \{0.25, 0.50\}$. Figures 14(a-b) show the average download time of peers binned according to the number of leechers in the swarm at the arrival time for $\epsilon = 0.25$ and 0.50, respectively. We conclude that, when the upload capacities are close to each other, the system exhibits a very similar behavior to that we observed when the upload capacities are the same (see Figure 4). Not surprisingly, the larger the range of upload capacities, the the greater the impact on the results, when compared to a constant upload capacity.



(a) $\epsilon = 0.25$. (b) $\epsilon = 0.50$.

Figure 14: Average download time as a function of arrival order in a busy period, when the uplink capacity is $c_{l_i} \sim U(c_l(1-\epsilon), c_l(1+\epsilon))$, with $c_l = 64$ kBps

.

5. Predicting bursty departures

The model presented in Section 4 can be used to estimate the number of departures that occur in a burst. In particular, consider the arrival of a leecher that initiates a busy period (i.e., the first arrival to a swarm with no leechers). In the following, we estimate the average number of peers that depart the swarm in a burst together with the leecher that initiated the busy period.

In practice, bursty departures do not occur exactly at the same time due to variations inherent to the network and to the inexistence of mechanisms that enforce synchronization between peers implemented in the protocol (e.g.: they do not request pieces at exactly the same time). Nonetheless, our model does not take these factors into account and, thus, we focus on leechers that leave the swarm at exactly the same time as the first leecher.

Let f denote the first leecher of a busy period and assume that the leecher arrival follows a Poisson distribution with rate λ. Also, as assumed by the model, a seed is always present and has uplink capacity of c_s, while leechers have identical uplink capacities equal to c_l. Finally, let S denote the number of pieces of the content.

We know that each leecher downloads pieces from the seed at rate c_s/N, where N is the number of peers in the swarm. These pieces are interesting to all the other $N-1$ peers and can be sent to them. Thus, if $c_l < c_s \times \frac{N-1}{N}$, leechers will start to accumulate pieces received from the seed which cannot be uploaded to the other peers. Therefore, every leecher will own pieces interesting to all of its neighbors. Consequently, the upload rate between any two leechers i and j will be equal to $u_{ij} = c_l/(N-1)$, since $g_{ij} = \infty$ (see Equation (8)). We conclude that when $c_l < c_s \times \frac{N-1}{N}$, all leechers have the same download rate which prevents other leechers from departing in a burst with f.

Conversely, when $c_l \geq c_s \times \frac{N-1}{N}$, the neighbors of f can upload to it the pieces they download from the seed. Since leecher f downloads from the seed at rate c_s/N and each of its $N-1$ neighbors receives pieces from the seed and uploads them to f at the same rate, f will download the content at a constant rate equal to c_s, independently on the number of peers in the swarm. Note that c_s is also the upper bound on the average download rate, as the seed cannot uploads pieces into the swarm at a faster rate. Hence, leecher f will take $T = S/c_s$ seconds to finish the download.

We now show how to calculate the lower and upper bounds to the number of bursty departures when $c_l \geq c_s \times \frac{N-1}{N}$. Consider arrivals that occur while peer f is in the swarm. The number of such arrivals, say $n = N-1$, is a random variable and follows the Poisson distribution with parameters λ and T. The download rates of these leechers are a function of n and also of their instants of arrival. Moreover, as discussed in Section 4.2, larger values of n imply a larger spread in the download rates. To obtain a conservative lower and upper bound on these download rates, we will consider a sufficiently large value for n. In particular, we use the 99-th percentile of n, namely n_{99}, and thus, $P[n \leq n_{99}] \leq 0.99$.

Given that exactly n_{99} leechers will join the swarm before the departure of f, we can use the model to obtain the minimum and maximum download rates of these peers, independently of their inter-arrival timing. Let $d_{\min}$ and $d_{\max}$ be, respectively, the minimum and the maximum download rates obtained from the model given that the swarm has $n_{99}+1$ leechers.

Consider again the subsets presented in the previous section, namely A (leechers with the same number of pieces than f) and B (leechers with less pieces than f). The minimum download rate is obtained by a leecher m in B when the only leecher in A is f. In this case, the download rate $d_{\min}$ is given by

$$d_{\min} = \frac{c_s}{n_{99}+1} + u_{f,m} + \sum_{i \in B} u_{i,m}, \tag{11}$$

where $\sum_{i \in B} u_{i,m}$ corresponds to the sum of the rates at which m downloads from peers in B.

On the other hand, a leecher m in B obtains the maximum download rate $d_{\max}$ when it is the only peer in B, i.e, $|A| = n_{99}$. In this case, the download rate is given by

$$d_{\max} = \frac{c_s}{n_{99}+1} + \sum_{i \in A} u_{i,m}. \tag{12}$$

The minimum and maximum time for the leechers to download the content is, respectively, S/d_{max} and S/d_{min}. Therefore, at least all leechers that arrive before $T - S/d_{min}$ will leave the swarm together in a burst with f. The expected number of peers that will arrive within this time period, B_{min} is simply given by

$$B_{min} = \lambda \left(T - \frac{S}{d_{min}} \right). \tag{13}$$

Similarly, at most all leechers that arrive before $T - S/d_{max}$ will leave the swarm in a burst

Table 2: Bounds for the expected number of leechers that depart in a burst with f, for $\lambda = 1/1000$.

c_s (kBps)	$E[N]$	B_{min}	B_{max}	$\frac{B_{min}}{E[N]}$	$\frac{B_{max}}{E[N]}$
48	5.333	1.667	4.378	0.312	0.821
64	4.000	0.400	1.895	0.100	0.474

with f. The expected number of peers that will arrive within this time period, B_{max} is simply given by

$$B_{max} = \lambda \left(T - \frac{S}{d_{max}} \right). \tag{14}$$

Finally, B_{min} and B_{max} provide a lower and upper bound for the average number of leechers that will depart the swarm in a burst with f.

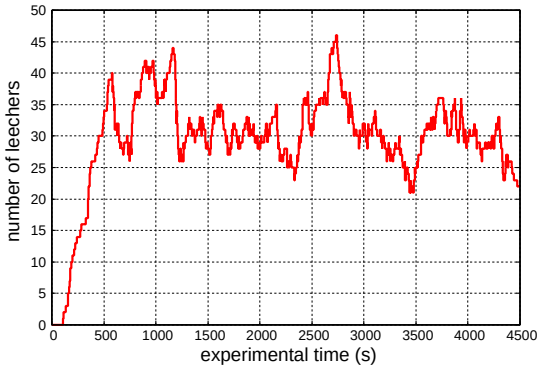
Table 2 shows the expected number of arrivals to the swarm before f departs, $E[N]$, which is simply λT, and both the lower and upper bounds B_{min} and B_{max}, respectively. The table shows numerical results for different c_s values but with $c_l = 64$ kBps and $\lambda = 1/1000$. The results indicate that average number of peers that depart the swarm in a burst with f can be significant: between 31% and 82% of all arrivals when the seed is slower than the leechers and between 10% and 47% when they have the same upload capacity. We also observe that these ratios reduce as c_s increases, indicating that bursty departures are less likely to occur with faster seeds. Recall that, as indicated above, there is a minimum value of c_s for which bursty departures do not occur.

6. General discussions

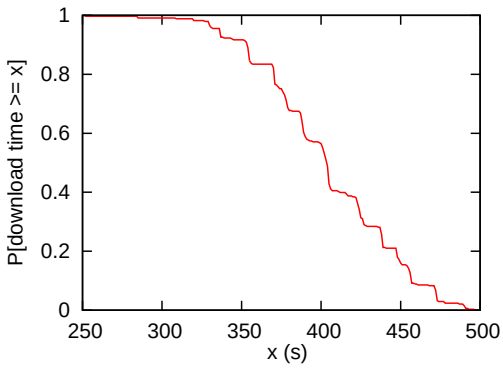
We now discuss other aspects related to the described phenomenon such as different arrival processes, what happens if the seed is not available all the time, what happens when leechers stay as seeds for some time and the missing piece syndrome.

6.1. General arrival processes

It is interesting to consider the occurrence of the observed phenomenon in more general scenarios. Although we have shown its prevalence under a crafted peer arrival process and under Poisson arrivals, we claim that homogeneous peers can have heterogeneous download rates under very general arrival patterns. In particular, given any arrival pattern of peers into a swarm, it is possible to choose system parameters (i.e., seed upload capacity, leechers upload capacity, and file size) such that the effects described in this paper will be very prevalent. Intuitively, by choosing a fast enough seed, peers will not be able to disseminate old pieces before new ones are pushed into the swarm, and thus will have significantly different number of



(a) Number of leechers over time



(b) Empirical CCDF of the download time.

Figure 15: Experimental results under a popular swarm ($\lambda = 1/12$, $c_s = 50$ kBps, $c_l = 50$ kBps)

blocks. In a sense, the behavior observed and described in this paper is quite general, although the requirement of the swarm being unpopular is important, as we next describe.

What happens if we consider very popular swarms, where the peer arrival rate is very large, yielding very large swarm sizes? Figure 15a shows experimental results of the dynamics of leecher arrivals and departures for this scenario (Poisson arrivals with rate $\lambda = 1/12$, uplink capacities of $c_s = 50$ kBps and $c_l = 50$ kBps) and file size $S = 80$ pieces (i.e., 20 MB). The empirical CCDF of the download time is depicted in Figure 15b. Interestingly, we can still observe the consequences of having heterogeneous download rates, such as bursty departures, content synchronization and high variability of download times (peers that leave in a large burst have different download times, as arrival is well-behaved), for example, at times 600 s and 1200 s. In a sense, the phenomenon is quite prevalent even during the busy period, but not strong enough to end the busy period. The characterization and modeling of the phenomenon in this scenario is much more entailed, given the complicated dynamics of piece exchange of BT and consequently the interest relationship among peers. We leave the investigation of these scenarios (popular swarms) as future work.

6.2. When the seed is not available all the time

We have considered so far swarms that have a single seed which is always connected. However, what happens if the seed alternately joins and leaves the swarm? Intuitively, leechers start to synchronize their contents right after the seed leaves because no new pieces are being placed into the swarm. After they become fully synchronized, they will stall until the seed comes back. Then, since they are synchronized, they will have relatively low download rates and will leave almost at the same time. Therefore, the intermitent seed makes the average download rates even more heterogeneous.

In order to support this claim, we modify the simulation model such that the state of the seed (connected/disconnected) is given by an ON-OFF source. We assume that the time until the seed leaves the ON state (leaves the swarm) is exponentially distributed with rate

$1/\lambda$ (arbitrarily set to be equal to the leecher arrival rate). Furthermore, we choose the rate at which the seed goes from the OFF back to the ON state (rejoins the swarm) so that the availability of the seed is 0.75, 0.50 and 0.25.

Table 3 summarizes the results for $\lambda = 1/1000$ s, $c_s = c_l = 64$ kBps, $S = 1000$ pieces. Each scenario was simulated during $800,000$ s. We observe that the mean, variance and maximum download time monotonically increase, what is expected as there are less resources on average. Interestingly, the minimum download time becomes smaller. This is due to the fact that a a new leecher may arrive when some peers are stalling in the absence of the seed, just before finishing the download. The new leecher then benefits from the spare bandwidth capacity and might complete the download right after the seed comes back. Finally, it is clear the download time (equivalently, download rate) becomes more heterogeneous.

$P(S = \text{ON})$	Mean	Variance	Minimum	Maximum
1.00	3.360×10^3	9.319×10^4	2.295×10^3	4.247×10^3
0.75	3.865×10^3	1.003×10^6	1.276×10^3	8.431×10^3
0.50	5.307×10^3	1.062×10^7	5.640×10^3	2.518×10^4
0.25	1.045×10^4	7.671×10^7	3.640×10^3	3.321×10^4

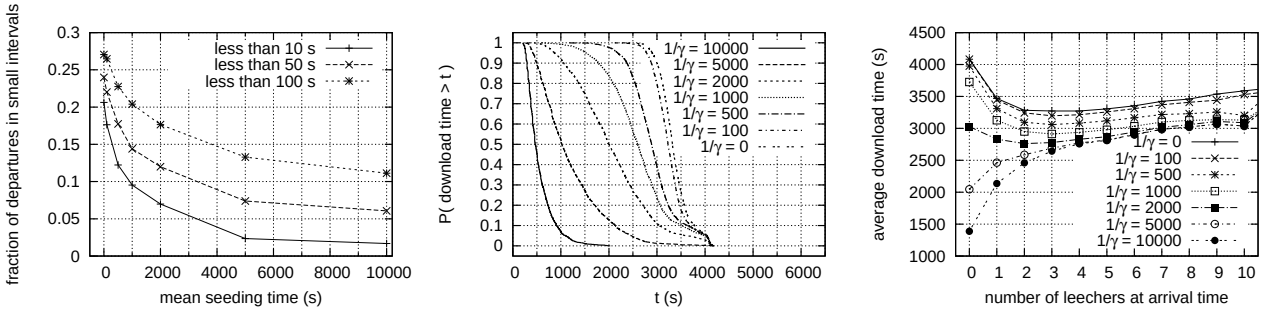
Table 3: Statistics of the empirical distribution of the download time, when the seed leaves and joins the swarm.

Note that, in fact, the proposed analytical model can cope with this scenario as long as peers do not accumulate pieces interesting to each other, i.e. $c_l \geq c_s \times \frac{N-1}{N}$ (see Section 5). In particular, if the seed departs, then this will only affect the upload rate among peers, given by Equation (1) (or Equation (7)). More precisely, the term corresponding to the download rate from the seed (c_s/N) should be set to zero in the equations that describe time periods where the seed is not present. Thus, given the state of the swarm with respect to the seed's presence and number of leechers, we can still apply our modeling framework and determine the download rates of leechers (under the condition above).

6.3. When leechers stay as seeds for some time

Another aspect that can be taken into account is that leechers may stay as seeds for a period of time after they finish the download and before leaving the swarm. Intuitively, since the capacity available to disseminate the file increases as leechers stay as seeds, peers concurrently downloading a file tend to receive pieces at similar download rates, possibly reducing the consequences of heterogeneous download rates. We performed simulations for scenarios where $\lambda = 1/1000$ s, $c_s = c_l = 64$ kBps, $S = 1000$ pieces and the time which leechers stay seeding is deterministic and equal to $1/\gamma$. Each scenario was simulated 10 times during 400,000 s, but the first 100,000 s were discarded to avoid transient effects. Figures 16(a)-(c) depict the results for many values of $1/\gamma$.

As indicated in Figure 16a, bursty departures are less likely to occur when leechers stay in the swarm after downloading the entire content. However, for small values of $1/\gamma$ (with respect to $1/\lambda$), the difference is barely noticeable and the departure process is still very bursty.



(a) Bursty departures characterization.

(b) Empirical CCDF of the download time.

(c) Impact of arrival order.

Figure 16: System performance when leechers stay seeding.

Table 4: Statistics of the empirical distribution of the download time, when leechers stay as seeds for some time.

$1/\gamma$	Mean	Variance	Minimum	Maximum
0	3.360×10^3	9.319×10^4	2.295×10^3	4.247×10^3
100	3.262×10^3	1.093×10^5	2.120×10^3	4.181×10^3
500	2.926×10^3	2.446×10^5	1.174×10^3	4.183×10^3
1000	2.612×10^3	4.394×10^5	6.028×10^2	4.181×10^3
2000	2.067×10^3	6.278×10^5	3.197×10^2	4.170×10^3
5000	1.190×10^3	4.306×10^5	2.600×10^2	4.209×10^3
10000	5.508×10^2	7.614×10^4	2.100×10^2	2.028×10^3

Figure 16b shows the CCDF of leechers' download times for different values of $1/\gamma$ while Table 4 contains statistics of these distributions, namely the sample mean and variance, minimum and maximum values. Intuitively, leechers that find two or more seeds at arrival time have significant better performance and hence the minimum download time decreases as the seeding time increase. On the other hand, the maximum download time is the approximately same for the majority of the curves. This is because there is a non-zero probability that a leecher downloads the content entirely from a single seed. However, this probability becomes smaller with the seeding time. Initially the variance increases with $1/\gamma$, but when leechers stay as seeds for a long period of time, it is unlikely that a leecher will have a download time much larger than the average and thus, the sample variance diminishes (see $1/\gamma = 5{,}000$ and $1/\gamma = 10{,}000$ in Table 4).

Finally, Figure 16c shows that while early arrivals are detrimented for small values of $1/\gamma$, they are benefited when $1/\gamma$ is high. The presence of multiple seeds has the same effect as a single seed with higher capacity. This can be observed by comparing the curves for $1/\gamma$ equal to 5,000 and 10,000 and the curve corresponding to $c_s = 96$ kBps in Figure 4.

6.4. Missing piece syndrome

Last, we now comment on the relationship of our findings and the phenomenon known as *missing piece syndrome*. This phenomenon states that in swarms where the arrival rate is large enough, the system can become unstable (i.e., number of leechers grows unboundedly) if the upload capacity of the seed is not large enough [8, 9, 10]. The key aspect of this syndrome is content synchronization, where a large fraction of peers have all but one and the same piece. This situation is particularly bad to the performance of the swarm, as the departure rate of the swarm will be equal to the seed upload capacity (assuming peers depart as soon as they acquire the last block). Our work has shown that peers can synchronize their content in such a way that several identical pieces are missing which eventually leads to the missing piece syndrome. In some sense, this generalizes the syndrome to a *piece synchronization syndrome*, which is inherent to BT dynamics due to the heterogeneous download rates as discussed in this work. Once peers have synchronized their content, they can only acquire new pieces from the seed, at the upload capacity of the seed. In this scenario, the *missing piece syndrome* is bound to occur.

7. Related prior works

Modeling P2P file sharing systems and in particular BT has been an active area of research in the past few years, driven mainly by the high complexity, robustness and user-level performance of such systems. One of the first BT models to predict the download times of peers was presented in [5]. This simple fluid model based on differential equations assumes homogeneous peer population (with respect to download and upload capacities) and Poisson arrivals, but yields analytical steady state solution for performance metrics. Several subsequent BT models have been proposed in the literature to capture various system characteristics, among them heterogeneous peer population (with respect to upload and download capacities) [17, 6, 18, 7]. However, to the best of our knowledge, all models predict that identical peers (with respect to their upload capacities) simultaneously downloading a file will have similar performance (with respect to download rates), contrary to the findings in this paper. Moreover, BT models generally assume either a rather large peer arrival rate (e.g., Poisson) or a large flash crowd (all peers join the swarm at the same time). This is somewhat surprising, given that most real BT swarms are rather small in size and quite unpopular [11]. Finally, one perverse effect of this lack of popularity, known as content unavailability, is shown to be a severe problem found in most of BT swarms [19].

Another interesting aspect of BT has been the discovery and characterization of some non-trivial phenomena induced by its complex dynamics. For example, peers in BT swarm tend to form clusters based on their upload link capacities, exhibiting a strong homophily effect. In particular, peers with identical upload capacities tend to exchange relatively more data between them [20, 21, 22]. Yet another peculiar behavior is the fact that arriving leechers can continue to download the entire content despite the presence of any seed in the swarm, a property known as self-sustainability [23]. More recently, the *missing piece syndrome* has

been characterized mathematically [9, 10]. In [24] is presented an evaluation for the impact of different peer selection strategies on the stability of the system. Such strategies may reduce the effect of content synchronization among peers. However, to the best of our knowledge, we are not aware of any prior work that has alluded the phenomenon we describe in this paper, namely, that homogeneous peers can have heterogeneous download rates.

8. Conclusion

This paper identifies, characterizes and models an interesting phenomenon in BT: homogeneous peers (with respect to their upload capacity) experience heterogeneous download rates. The behavior is pronounced in unpopular swarms (few leechers) and has important consequences that directly impact peer and system performance. The mathematical model proposed captures well these heterogeneous download rates of peers and provides fundamental insights into the root cause of the phenomenon. Namely, the allocation of system capacity (aggregate uplink capacity of all peers) among leechers depends on the piece interest relationship among peers, which for unpopular swarms is directly related to arrival order of peers and can be significantly different among them.

Acknowledgment

This research was supported in part by grants from CAPES, CNPq FAPERJ (Brazil) and NSF under the grant CNS-1065133.

References

[1] B. Cohen, Incentives build robustness in BitTorrent, in: P2PECON, 2003.

[2] Ipoque internet study, `http://www.ipoque.com/news_&_events/internet_studies/internet_study_2007` (2007).

[3] R. L. Xia, J. Muppala, A survey of bittorrent performance, IEEE Communications Surveys & Tutorials.

[4] X. Yang, G. de Veciana, Service capacity of peer to peer networks, in: IEEE INFOCOM, Vol. 4, 2004, pp. 2242 – 2252.

[5] D. Qiu, R. Srikant, Modeling and performance analysis of bittorrent-like peer-to-peer networks, in: ACM SIGCOMM, 2004, pp. 367–378.

[6] W.-C. Liao, F. Papadopoulos, K. Psounis, Performance analysis of bittorrent-like systems with heterogeneous users, Performance Evaluation 64 (9-12) (2007) 876 – 891, IFIP Performance 2007.

[7] A. Chow, L. Golubchik, V. Misra, Bittorrent: An extensible heterogeneous model, in: IEEE INFOCOM, 2009, pp. 585 –593.

[8] F. Mathieu, J. Reynier, Missing piece issue and upload strategies in flashcrowds and p2p-assisted filesharing, AICT/ICIW 0 (2006) 112.

[9] B. Hajek, J. Zhu, The missing piece syndrome in peer-to-peer communication, in: IEEE ISIT, 2010, pp. 1748 –1752.

[10] J. Zhu, B. Hajek, Stability of peer to peer systems, in: ACM PODC, 2011.

[11] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, X. Zhang, A performance study of Bittorrent-like peer-to-peer systems, IEEE JSAC 25(1) (2007) 155–169.

[12] S. Kaune, R. C. Rumín, G. Tyson, A. Mauthe, C. Guerrero, R. Steinmetz, Unraveling bittorrent's file unavailability: Measurements and analysis, in: IEEE Tenth International Conference on Peer-to-Peer Computing (P2P 2010), 2010, pp. 1–9.

[13] T. Hossfeld, F. Lehrieder, D. Hock, S. Oechsner, Z. Despotovic, W. Kellerer, M. Michel, Characterization of bittorrent swarms and their distribution in the internet, Computer Networks 55 (5) (2011) 1197–1215.

[14] A. Legout, G. Urvoy-Keller, P. Michiardi, Rarest first and choke algorithms are enough, in: ACM IMC, 2006, pp. 203–216.

[15] E. de Souza e Silva, D. R. Figueiredo, R. M. Leão, The TANGRAM-II integrated modeling environment for computer systems and networks, SIGMETRICS Perform. Eval. Rev. 36 (2009) 64–69.

[16] L. Peterson, A. Bavier, M. E. Fiuczynski, S. M. Ly, Experiences building planetlab, in: USENIX OSDI, 2006, pp. 351–366.

[17] F. Lo Piccolo, G. Neglia, The effect of heterogeneous link capacities in bittorrent-like file sharing systems, in: Hot-P2P, 2004, pp. 40 – 47.

[18] M. Meulpolder, D. Epema, H. Sips, Replication in bandwidth-symmetric bittorrent networks, in: IEEE IPDPS 2008, 2008, pp. 1 –8.

[19] D. Menasche, A. Rocha, B. Li, D. Towsley, A. Venkataramani, Content availability and bundling in swarming systems, in: ACM CoNEXT, 2009, pp. 121–132.

[20] A. Legout, N. Liogkas, E. Kohler, L. Zhang, Clustering and sharing incentives in bittorrent systems, ACM SIGMETRICS Perform. Eval. Rev. 35 (1) (2007) 301–312.

[21] A. R. Bharambe, C. Herley, V. N. Padmanabhan, Analyzing and improving a bittorrent networks performance mechanisms, in: IEEE INFOCOM, 2006, pp. 1 –12.

[22] F. Murai, A. Rocha, D. R. Figueiredo, E. de Souza e Silva, Can identical BitTorrent peers experience different download times?, in: IFIP WG 7.3 Performance 2010 Poster Abstracts, Namur, Belgium, 2010, pp. 29–30, extended Abstract.

[23] D. S. Menasché and A.A. Rocha and E. de Souza e Silva and R.M. Leo and D. Towsley and A. Venkataramani, Estimating self-sustainability in peer-to-peer swarming systems, Performance Evaluation 67 (2010) 1243–1258.

[24] D. Menasche, A. Rocha, E. de Souza e Silva, D. Towsley, R. Leao, Implications of peer selection strategies by publishers on the performance of p2p swarming systems, Performance Evaluation Review.

Fabricio Murai received the B.Sc. degree in computer science and the M.Sc. degrees in computer and system engineering both from Federal University of Rio de Janeiro, Brazil, in 2007 and 2011, respectively. He is currently a Ph.D. student in the Computer Science Program at University of Massachusetts Amherst (UMass). His areas of interest include modeling and performance evaluation, P2P systems and Complex Networks.

Antonio A. de A. Rocha received the B.Sc. degree in computer science from Salvador University, Brazil, in 2001 and the M.Sc. and D.Sc. degrees in computer and system engineering from Federal University of Rio de Janeiro, Brazil, in 2003 and 2010, respectively. Currently, he is an Assistant Professor in the Computer Science Department at the Fluminense Federal University, Niterói, Brazil. His areas of interest include modeling, analysis and performance evaluation of computer systems, traffic engineering and network inference and measurement. Dr. Rocha received the best paper award at CoNEXT09.

Daniel R. Figueiredo obtained M.Sc. and Ph.D. degrees in computer science from University of Massachusetts Amherst (UMass) in 2005 and M.Sc. degree in computer and system engineering from Federal University of Rio de Janeiro, Brazil in 1999. He worked as a researcher (post-doc) at the École Polytechnique Fédérale de Lausanne (EPFL) from 2005 to

2007. Since 2007, he is Assistent Professor in the computer and system engineering department from UFRJ. His areas of interests lie in Complex Networks and Computer Networks, in particular, mathematical modeling of systems and Internet applications.

Edmundo A. de Souza e Silva received the B.Sc. and M.Sc. degrees in electrical engineering, both from Pontifical Catholic University of Rio de Janeiro (PUC/RJ), and the Ph.D., degree in computer science from the University of California, Los Angeles in 1984. He has been a Visiting Scientist and Visiting Faculty at the IBM T.J. Watson research Center, a Visiting Scientist at the IBM Tokyo Research Laboratory, a Lecturer with the UCLA Department of Computer Science and Computer Science Department at USC. He has also been a visiting researcher at the Politecnico di Torino, Chinese University of Hong Kong and IRISA/INRIA-Rennes. He has participated in several technical committees of international conferences. He was a PC co-Chair of the IFIP Third Int. Conf. on Data Commun. Syst. and their Performance (1987), PC co-Chair of IEEE/GLOBECOM99, PC vice-co-Chair of ITC2001 and PC co-Chair of ACM/Sigmetrics2002. He was a member of the Board of Directors of ACM/SIGMETRICS during 2001-2005. Currently he is a professor at the Federal University of Rio de Janeiro, COPPE and Computer Science Department. He is also vice-chair of the IFIP WG 7.3. His areas of interest include the modeling and analysis of computer systems and computer communication multimedia networks.