Modeling and Analysis of Seed Scheduling Strategies in a BitTorrent Network

Pietro Michiardi
Institut Eurecom, France
Email: michiardi@eurecom.fr

Krishna Ramachandran and Biplab Sikdar
Department of Electrical, Computer and Systems Engineering
Rensselaer Polytechnic Institute, Troy NY 12180
Email: ramak,sikdab@rpi.edu

***Abstract*— In the current work, we propose an analytical model to characterize various scheduling options from a seed's perspective so as to efficiently disseminate the file parts thereby ensuring that download times for the network is optimized.**

I. Introduction

Peer-to-peer (P2P) networks provide a paradigm shift from the traditional client server model of most networking applications by allowing all users to act as both clients and servers. The primary use of such networks so far has been to swap media files within a local network or over the Internet as a whole. Among current solutions deployed in the Internet, BitTorrent has received a lot of attention from the research community because of its scalability properties and its ability to handle the so called *flash crowd* scenario, a transient phase characterized by a sudden burst of concurrent requests for a popular content. However, recent results [2]–[4], [10], [12] have revealed some inefficiencies of BitTorrent that translate into a transient phase, indicating the source of the content (called the *seed*) as the main cause of a disproportionate distribution of the content among the downloaders. In this paper, we motivate the need to incorporate intelligence into scheduling file pieces at the *seed* and develop an analytic framework wherein the impact of the chosen strategies can be studied for a BitTorrent type P2P network. We propose a novel scheduling policy called Proportional Fair Scheduling (PFS) that improves the content distribution process based both on past scheduling decisions and on the actual distribution of content requests as seen by the seed. Using the proposed analytical framework we compare our scheduling policy with the one used in the *mainline* BT implementation and with the best known scheduling improvement called "smartseed" [4]. Through numerical evaluation we show that PFS outperforms previous policies in the short term. For the long term analysis we built a custom BitTorrent simulator and show that our scheduling algorithm achieves a fair content distribution, and reduces the time needed for the seed to inject the content in the system. To summarize, our contributions in the current work can be stated as follows:

- Present an analytic framework wherein different scheduling policies can be modeled and their behavior analyzed.
- Propose a new algorithm, called Proportional Fair Scheduling (PFS) for piece distribution that performs better than the current proposed scheduling modification for the seed.

A. BitTorrent overview

Before proceeding further, we provide a brief overview of the BitTorrent system. BitTorrent is a P2P application that attempts to replicate content faster by leveraging the upload bandwidth of the peers involved in the download process. Each unique content in the system is associated with a $.torrent$ file, and is independent of the remaining torrents in the system. What this implies is that a peer's view of the BT system is confined to a subset, termed the peer set, of all the hosts associated with a specific torrent. Peers wishing to download a particular content obtain the corresponding $.torrent$ file from a web server and use a centralized entity called the tracker to collect a random subset of hosts currently active in the torrent. Peers involved in a torrent cooperate to replicate the file among each other using swarming techniques. BitTorrent achieves scalable and efficient content replication by employing the choke and rarest first algorithms. The former is used for peer selection, i.e. which peer to upload to, while the latter for selecting the file part scheduled to be transfered. Finally, a peer in BitTorrent exists in two states: seed state wherein it has the entire content or leecher wherein it is in the process of downloading the file. The motivation for the current work arises due to the manner in which a seed and a leecher perceive the torrent. Note that we have limited our description to details relevant to the current work and have glossed over several technicalities of the BT protocol, which may be found in [8].

The rest of the paper is organized as follows: in Section II we survey related literature, while in Section III we discuss on the rationale and motivations of our work. In Section IV we present our analytical model that emulates the various content scheduling strategies for a seed, Section IV-A provides an analytical dissection and addresses issues such as stability and convergence of the scheduling strategies. We present our results in Section IV and draw relevant inferences from them and finally summarize the work in Section VI.

II. Related Work

In recent times BitTorrent has received substantial interest from the research community, with several modeling as well as

simulation studies aiming at improving its performance. Mathematical models for BT are presented in [4]–[6]. In [5] a fluid model is used to characterize the performance of BitTorrent like networks in terms of the average number of downloads and download times. The authors in [6] propose to improve upon the aforementioned modeling work using a stochastic differential equation approach, by incorporating more realistic BT network behavior in their study. A Markovian model of a BT network was studied in [4], wherein the authors propose a novel peer selection strategy to improve download times. Along similar lines is another modeling work, [11], wherein a branching process based Markovian model was formulated to study BitTorrent like networks.

Simulation based studies are the focus of the works presented in [2], [3], [7], [9], [10]. In [2], the authors investigate the efficacy of the rarest first and the choke algorithms while [3] documents the impact of various system parameters on the networks performance. Along similar lines, [9] presents the dissection of the performance of the mechanisms and algorithms used by BT over a five month period. In [7], the authors make the case for a network coding scheme to improve content replication, while in [10], the authors study the performance of BT by employing metrics such as file download time, link utilization and fairness.

A common feature shared by the literature surveyed thus far is the attempt at modeling the BT system in its entirety. As a result, not all facets pertaining to efficient content distribution are explored. For instance, the first step in this direction is to ensure that the initial seed is able to inject the $entire$ content among the leechers at the earliest and this calls for specialized scheduling algorithms. Unfortunately, with a wholistic approach, this is difficult to accomplish. In this current work we restrict our attention to a particular type of peers, namely $seeds$, and study the impact of scheduling decisions at their end on the effectiveness of content distribution in the system. This is elaborated further in the following section.

III. Rationale and motivation

Typically when content first appears in a BT network, it is stored at a single host, i.e. there is a single seed. From here on, the lifetime of a torrent can be broadly classified into three stages:

1) flash crowd scenario: the seed experiences a huge volume of concurrent requests for the content.
2) steady state : the arrival of requests is more spaced and can be considered regular.
3) dying out : this stage marks the point where a substantial portion of the leechers complete downloading the content and leave the system. In other words the torrent starts "dying".

The motivation for the current work stems from the findings of various simulation studies [3], [10], [12] revealing an inefficiency in the performance of the protocol during the flash crowd phase of a torrent arising from a disproportionate distribution of content among the leechers. It was found that in the flash crowd scenario, often the distribution from the seed becomes a bottleneck in the replication process. In such a scenario, a lack of intelligence during the upload process at the seed could result in some of the pieces not being replicated at all. This phenomenon is termed *starvation* and can adversely impact the torrent's performance in the following manner: consider the scenario where after a certain time (say t), the seed decides to go offline. At such time, if there are certain parts of the file that have not yet been replicated among any of the leechers, then the torrent would eventually die out since none of the leechers would be able to complete the download. Even otherwise, a disproportionate distribution of the parts would result in a prolonged flash crowd scenario since the leechers have nowhere else to request the parts from. In other words the seed and the leechers hosting the rarer parts would be swamped with a huge volume of upload requests. This problem if further magnified if the seed is bandwidth constrained. Thus, an improved distribution of content at the seed's end would serve to improve the performance of the torrent by

- decreasing the load off the seeds since the leechers now have ample sources among themselves. This in turn provides a better incentive for the seed to stay online a longer time since it's bandwidth is no longer choked
- decreasing the download time of the leechers, since there is a bigger pool of leechers with the same piece.
- increase the "longevity" of the torrent since there is higher level of redundancy in the system. Here, we define longevity of the torrent as the first time instant when the the content in its $entirety$ is not present among the peers in the BT system. Such a situation can arise when all the seeds and a fraction of the leechers disconnect from the system, resulting in missing piece(s).

A relevant doubt at this stage would be to question the rationale behind distinguishing between scheduling decisions at a seed and those at a leecher. In other words, *why would not a common scheduling algorithm work for both ?* The answer to this lies in the difference between the view of the torrent as seen by a leecher and a seed. While the leecher has complete information on the part distribution among the peers in it's peer set, this knowledge is hidden from the seeds. This is primarily due to a mechanism used to reduce the control message overhead named the HAVE suppression technique. HAVE messages are used to disseminate information on the piece distribution among leechers: each time a leecher finishes to download a piece, she will inform all peers in her peer set about the new piece availability. The HAVE suppression technique inhibits the transmission of HAVE messages to those peers that currently have a replica of the announced piece. The consequence is that seeds will have no information on the piece distribution in her peer set. In fact, in the current *mainline* implementation of the BT protocol, a seed simply replies to piece request originated at the leechers without any scheduling decision (hence the name random scheduling (RS) used hereafter). Thus, lack of a global snapshot constrains a seed to base scheduling decisions on it's own past history and

hence the motivation behind the current work. The endeavor in the current work is arrive at a mathematical framework generic in nature so as to facilitate the performance quantification of various scheduling strategies that could be implemented at the seed. In this paper we try and address the following problem: *How best can a seed incorporate the limited view of the BT system into it's scheduling decisions so as to ensure better content distribution among the downloaders?*

To this end, as a part of their simulation study of BT, the authors in [10] propose the local rarest first (LRF) policy, termed "smart seed" scheduling policy, as an improvement over the current scheduling scheme. However, the optimality of such a strategy is not guaranteed and there could be other schemes that might yield better performance. Carrying out simulations for each scenario could be quite tedious and moreover provide no intuition into the long term behavior/distribution of file parts among the leechers. In this paper, we provide a theoretical grounding for the problem through a framework based on stochastic approximation algorithms. In particular, we compare the performance of our scheduling strategy, the proportional fairness scheme (PFS), with the current $proposed$ modification, local rarest first (LRF), and the $existing$ policy, random scheduling (RS).

IV. Analytical Framework

In this section, we present our analytic framework to study the performance of content distribution schemes implemented at the seed. While the framework is generic in nature and applicable to a large class of scheduling policies, for illustrative purposes, we confine the example scenarios in this paper to three cases, viz. local rarest first (LRF), proportional fairness scheme (PFS) and the random scheduling (RS) policy. The essence of each can be summarized as follows

- LRF: In this policy users are served on a first come first serve basis. Leechers request the seed for a set of blocks and the seed uploads the rarest amongst them.
- Proportional Fairness Scheme (PFS): In this scheme, the seed takes into account the requests coming in for each part and the corresponding past throughput and uploads that piece with the maximum ratio of the two.
- Random Scheduling (RS): There is no intelligence on the seed's part in this policy. It blindly serves the first leecher's requested piece. Note that this is the current scheduling policy implemented in the seed state by BT.

Before proceeding with the description of the model, we outline our assumptions: The content to be replicated (denoted by F) is divided into p equal parts and is stored at a $single$ seed, S. The seed is modeled by a single server queue with no buffer space. Time is slotted in intervals with the granularity of each round chosen to accommodate the transfer of a single file part. The seed serves only one part in a round, with the decision on the piece to be uploaded in the next round made based on the requests that arrive during the *current* time slot. The peer satisfying the scheduling criteria is served in the next slot while the rest of the requests are dropped. The above assumptions are a reasonable mapping to a bandwidth constrained seed where it makes sense to dedicate the entire bandwidth to serve a particular request instead of increasing the latency by dividing it.

Let the request vector at the end of slot n (start of slot $n+1$) be represented as $\mathcal{R}(n+1) = [r_{1,n+1}, r_{2,n+1}, \cdots, r_{p,n+1}]$, where $r_{i,n+1}$ denotes the number of times part i was requested for in round n. In other words, each entry in $\mathcal{R}(n+1)$ represents the number of leechers requesting for that particular part during the previous round, i.e. round n. Let the throughput vector be denoted as $\mathcal{T}(n) = [t_{1,n}, t_{2,n}, \cdots, t_{p,n}]$, where $t_{i,n}$ represents the number of times part i was served in n rounds. Similarly, let $\theta(n) = [\theta_{1,n}, \theta_{2,n}, \cdots, \theta_{p,n}]$ denote the vector of sum of requests for the different parts, each time it was served, averaged over the past n rounds. The average throughput and request rate for part i after n rounds are defined as follows:

$$\mathcal{T}_{i,n} = \frac{\sum_{k=1}^{n} \mathcal{I}_{i,k}}{n} \qquad \theta_{i,n} = \frac{\sum_{k=1}^{n} r_{i,k}\mathcal{I}_{i,k}}{n}$$

where $\mathcal{I}_{i,k}$ is an indicator variable with

$$\mathcal{I}_{i,k} = \begin{cases} 1 & \text{if part } i \text{ is scheduled in round } k \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Thus, at the end of each round, each entry in vectors θ and $\mathcal{T}$ can be updated as follows:

$$\theta_{i,n+1} = \theta_{i,n} + \epsilon_n[\mathcal{I}_{i,n+1} r_{i,n+1} - \theta_{i,n}] \tag{2}$$

$$\mathcal{T}_{i,n+1} = \mathcal{T}_{i,n} + \epsilon_n[\mathcal{I}_{i,n+1} - \mathcal{T}_{i,n}] \tag{3}$$

with $\mathcal{I}_{i,n+1}$ as in Equation (1) and $\epsilon_n = \frac{1}{n+1}$. Given the above system parameters, the seed scheduling algorithm we propose (PFS) can be summarized as follows:

- Among the non-zero request entries that arrive in a round, select that part maximizing the following ratio:

$$\arg\max_{i} \left\{ \frac{r_{i,n+1}}{\theta_{i,n} + d} \right\} \tag{4}$$

  If there are multiple parts satisfying the above criterion, break ties arbitrarily. Here, d_i is arbitrarily close to zero and is chosen to avoid the divide by zero error in the initial stages of the torrent when the throughputs for nearly all the parts are close to or equal to zero.
- Upload the chosen part from the previous step to the requesting peer. Again, break ties arbitrarily

It is quite natural to question the soundness, be it theoretical or practical, of a formulation as in Equation (4). The proposed format can be justified if the content replication process were to be viewed, from a seed's perspective, as a variant of the utility maximization problem. Note that in a BT system, the onus is primarily on the seed to ensure the spread of content among the peers in the system. Thus, a seed seeks to maximise the replicas of each piece among the leechers and therefore it is reasonable to assume that the utility function chosen is concave in nature (since a concave function is of the maximisation type) and is representative of the problem. In this context

consider the utility function to be the sum of the logarithm of average number of requests of the individual pieces, i.e.

$$U(\theta) = \sum_{i=1}^{p} \log(\theta_i + d) \tag{5}$$

Then it can be shown [14] that for this particular choice of utility maximization, the policy outlined in Equation (4) yields optimal results. We further note that the seed is not constrained to choose the policy of Equation (4). Any reasonable representative concave function can be chosen as the utility function and the scheduling policy appropriately tailored to obtain optimal results.

A. Convergence Analysis

The formulation of Equations (2) and (3) is in the framework of stochastic approximation algorithms [13]. Notably, under certain assumptions, which can be shown to be valid in a BitTorrent scenario, it can be shown that the stochastic approximation algorithm in Equation (3) can be described by a $deterministic$ mean field ordinary differential equation (ODE) system. This enables us to characterise the behavior of the proposed algorithm and is also a useful tool to study the asymptotic properties such as the long term throughput of the respective file pieces. An important consequence of the convergence proof is that concerning the stability of the system. For example, a scheduling policy that converges asymptotically also characterises a stable system. We now outline the assumptions required for the ODE convergence:

- Stationarity of the request distribution: $\{\mathcal{R}(l), l < \infty\}$. Note that in a BitTorrent type system, the requests generated by leechers for the missing file pieces depend only on the current distribution of the parts among each other. For instance, if a system snapshot at time t were to be translated to a different instant, say t_1, the pattern of requests generated would be similar. Define the stationary expectation for part i as

$$\hat{h}_i(\theta) = E[\mathcal{I}_{\{\frac{r_i}{\theta_i+d_i} \geq \frac{r_j}{\theta_j+d_j}\}, \forall j \neq i}] \tag{6}$$

- Lipschitz continuity of $\hat{h}_i(.), 1 \leq i \leq p$. We demonstrate this with the help of a simple case where the file consists of two parts and the joint probability density is given by $p(r_1, r_2)$. Then, for part 1, Equation (6) can then be simplified as

$$\hat{h}_1(\theta) = \int \mathcal{I}_{\{\frac{r_1}{r_2} \geq w\}} p(r_1, r_2) dr_1 dr_2 \tag{7}$$

where $w = (\theta_1 + d)/(\theta_2 + d)$. Note that in the above equation we have used a continuous density function for the request generation process, which is in fact discrete. This is because, it has been shown in [15], that the requests for the parts can be approximated by a Gaussian distribution which is continuous. In the current work, we employ the same approximation and hence the formulation of Equation (7). Now, Eqn. (7) is Lipschitz continuous with respect to w, since the area of the region where the indicator function is not zero is a differentiable function of w [14]. Similar is the case for $\hat{h}_2(\theta)$. Further, the deivatives of $\hat{h}_1(\theta)$ and $\hat{h}_2(\theta)$ will be continuous if $p(r_1, r_2)$ is bounded and continuous.

- Bounded density of $\mathcal{R}(n)$. This is trivially satisfied since the number of users in a BT system is finite thus ensuring that the requests generated during each round of time remain bounded.

Under the above assumptions, the stochastic approximation algorithm of Equation (3) can be approximated by the ODE given by:

$$\dot{\mathcal{T}}_i^{PFS} = E[\mathcal{I}_{\{\frac{r_i}{\theta_i+d_i} \geq \frac{r_j}{\theta_j+d_j}\} \forall j \neq i}] - \mathcal{T}_i^{PFS} \tag{8}$$

B. Modeling other policies

The analytic framework provides a generic setting wherein a wide class of scheduling policies can be modeled and quantified. We illustrate the robustness of the framework by modeling the LRF scheme in [10] as follows:

- For each piece i in the request block (RB) set $r_{i,n+1} = 1$
- Choose piece i such that: $\arg\max_{i \in RB} \left\{ \frac{1}{\theta_{i,n}+d_i} \right\}$; break ties arbitrarily
- Upload the piece from the previous step

The corresponding throughput formulation for part i, $\mathcal{T}_i^{LRF}$, is then given by:

$$\mathcal{T}_{i,n+1}^{LRF} = \mathcal{T}_{i,n}^{LRF} + \epsilon_n [\mathcal{I}_{\{\frac{1}{\theta_i+d_i} \geq \frac{1}{\theta_j+d_j}\} \forall j \neq i} - \mathcal{T}_{i,n}^{LRF}] \tag{9}$$

and the equivalent ODE by:

$$\dot{\mathcal{T}}_i^{LRF} = E[\mathcal{I}_{\{\frac{1}{\theta_i+d_i} \geq \frac{1}{\theta_j+d_j}\} \forall j \neq i}] - \mathcal{T}_i^{LRF} \tag{10}$$

V. Results

In this section we present results comparing the efficiency of the PFS scheme against LRF. To prove the robustness of the proposed framework, we quantify the performance gains obtained in the short term as well as in the long run. For the short term analysis we perform a numerical evaluation of the PFS scheduling using the stochastic approximation algorithm as described in Section IV. On the other hand, we perform the long term evaluation using a custom simulator of the entire BitTorrent system. The rationale behind this choice lies in the lack of a realistic characterization of the piece request rate $\mathcal{R}(n) = [r_{1,n}, r_{2,n}, \cdots, r_{p,n}]$ to be used in the analytical evaluation presented in Section IV-A. Our implementation, which is detailed in Section 2, also provides a global perspective of the system, as opposed to the seed's perspective offered by the analytical model.

A. Short term behavior

Since the primary objective in the initial stages of a torrent is to minimize starvation of pieces, a natural benchmark for comparing the policies would be to measure the number of starved pieces at a certain point of time under each policy.

Here, we choose to make the comparison after p rounds, where p denotes the number of pieces the content is divided into. The rationale behind this is as follows: since we assume that the seed schedules one piece per round, in the ideal case it would require p rounds to ensure that the file in it's entirety is present among the leechers. Figure 1 graphs the performance of the various policies in the flash crowd stage. In Figure 1(a), the number of starved parts of a 30 part file are plotted for each policy over 100 runs of our algorithm while Figure 1(b) quantifies the impact of the file size on the number of starved parts. Each point on the graph of Fig. 1(b) is an average of 100 runs. As seen from the plots, the proportional fair scheme offers significant gains over the other two policies. Even with increasing file sizes, the performance degradation is not very substantial. In fact, for a file consisting of 100 parts, the ratio of starved pieces in the "flash crowd" phase is about 1:3 for PFS and LRF, while it is around 1:18 when comparing PFS and the RS schemes. We believe the better performance of the algorithm could be attributed to the following factors:

- The seed makes a scheduling decision taking into account all the requests that are made in a particular round, unlike LRF and RS where users are served in a first come first served manner. For instance, if a large number of leechers request for a particular piece there is a higher probability of it being a rare piece as compared the rarity of a piece requested by a single user.
- In an open BT system the local rarest piece need not reflect reality, from the seed's perspective, due to leechers entering and leaving the system. Thus, even when a seed bases her scheduling decisions only on her past history like in the LRF case, due to peers' dynamics a seed may have a stale vision of what is rare and what is not in the system. The PFS scheme accounts for this by using the number of requests for a piece as the system's indicator of rarity and makes the scheduling decision accordingly.

B. Long term behavior

As a final validation of our theoretical formulation presented in Eqn (4), we present a simulation comparison of the proposed PFS algorithm against the LRF scheme, especially the behavior over long time periods. Since we only modify the seed scheduling algorithm, it only makes sense to quantify the impact within the seed's peer set and not globally. The main objective in the long term is to prevent a high variance in the number of replicas of each part, i.e. prevent a disproportionate piece replication in the peer set since it is the root cause of all problems. In other words the scheduling process should be "fair" to the individual pieces. The intuition behind this is that ensuring a balanced replication of the pieces can help improve download times since there is a higher level of redundancy and also distribute the load more evenly among the leechers thereby providing As a measure of the degree of fairness, we employ the Max-Min Fairness Index [] given by $\frac{min_{\forall i}(x_i)}{max_{\forall i}(x_i)}$, where x_i denotes the number of replicas of part i at the end of a round *in the seed's peer set*. The two important benchmarks under this metric are a) the distance of the ratio from 1 and b) the rate of convergence to 1. The closer the ratio is to 1, and the faster it converges to 1, greater is the fairness.

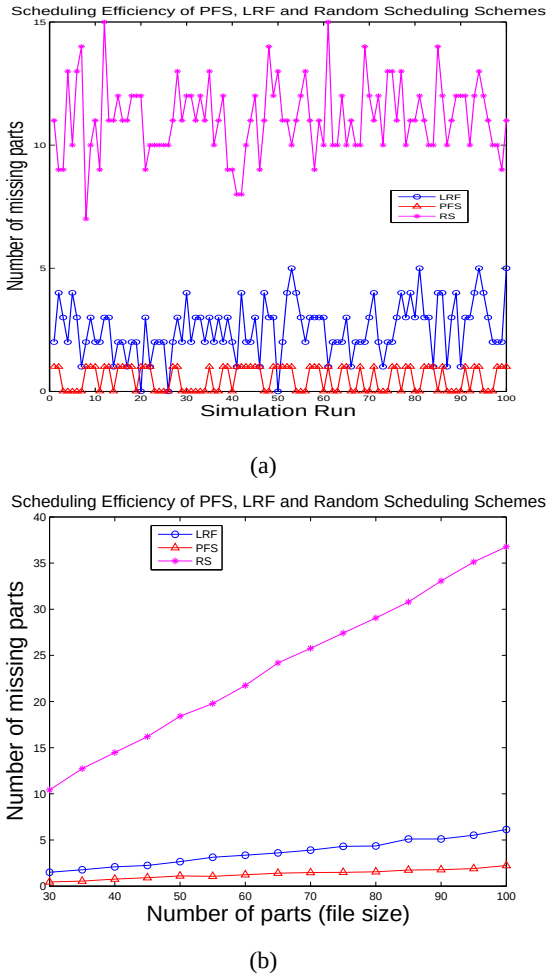


Fig. 1. Performance evaluation in the flash crowd phase

Before discussing the long term results we provide a brief description of the custom simulator we designed.

1) BitTorrent Simulator: We developed a synchronous simulator working in rounds wherein we implemented both seed and leecher algorithms following the BitTorrent specification (see [] for details). Namely, we implemented the rarest-first and the choke/unchoke algorithms executed at the leechers. We then implemented two scheduling policies at the seed side, the PFS and the LRF. The only limitation we imposed on the simulator follows the one of the analytical model: only one peer is unchoked in each simulation round. Our implementation accounts for a tracker component in charge of setting up the initial peer connections. The peer set size for a peer is set to the default value of the mainline BitTorrent implementation, that is 80 peers. To quantify the impact of the scheduling decisions, we assume that leechers that finish downloading leave the torrent, i.e. there is a single seed in the system at all times.

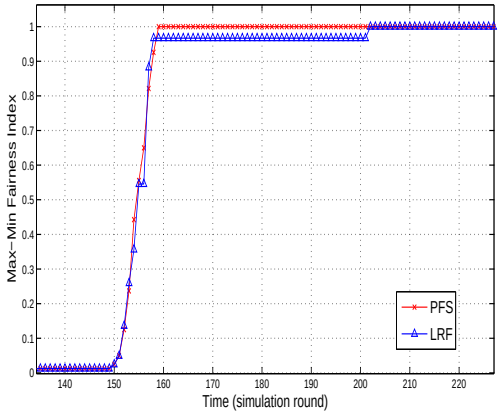
It is worthwhile noticing that, as compared to the LRF

scheduling policy which requires modifying both the seed and the leecher side of BitTorrent as well part of the protocol specification, PFS scheduling can be seamlessly integrated in BitTorrent with a simple modification at the seed side only (more details are available in our Technical Report []).

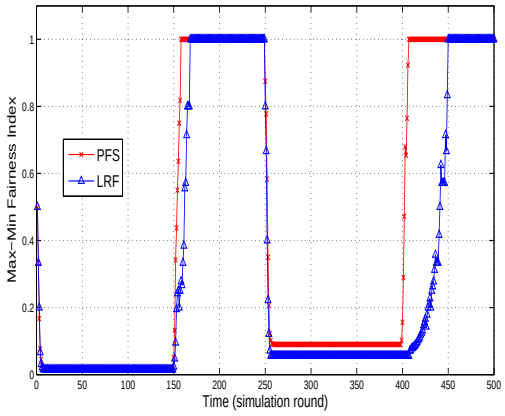
2) Simulation results: We compare the LRF and the PFS scheduling algorithms assuming the content to be split in $p = 150$ pieces. We simulate the presence of one seed only in the system and study two representative scenarios and realistic scenarios: the first where the torrent experiences a heavy flash crowd and the second indicative of a torrent with a high churn rate..

To simulate the flash crowd setting, 160 peers are injected into the system in the first and then there are no further additions. Further, compliant with our assumptions stated above, leechers depart the torrent once they have the entire content. The objective here is to study the algorithm's sensitivity towards acheiving a balanced replication in the wake of huge volume of requests. Note that the max-min fairness plots can also be used to infer and compare the download times experienced by the leechers. Since we assume that leechers with the entire content depart, the time T when the graph reaches one also denotes the instant when all the leechers in the system have finished downloading. Therefore, the quicker the graph peaks to one, the better it is in terms of fairness as well as download times. In Figure 2(a) we plot the Max-min fairness index versus time (in simulation rounds) for the flash crowd scenario described above. content. When using PFS scheduling, $T = 159$ while for the LRF case $T = 202$. A similar trend was observed over multiple repetitions of the experiment, showing an improvement of the total time to download the content in favor of PFS of roughly and this was more pronounced in the case of smaller files [**?**, PKB06]

In the second simulation study, we study the responsiveness of scheduling decisions at the seed when substantial variations in the population of peers downloading the content arise, i.e. a system with high churn. In particular, we consider 80 peers joining the system at round 1, then 30 randomly chosen peers leaving the system at round 150, and finally 80 new peers joining the system at round 250. Although both PFS and LRF scheduling reach the highest fairness index, Figure 2(b) clearly shows that PFS reacts consistently faster to peer dynamics as compared to LRF. Similar results (not reported due to lack of space) have been obtained for different runs of the same scenario.



(a)



(b)

Fig. 2. Simulation results for the long term analysis.

VI. Conclusion and Future Work

In this work, we motivated the need for improved scheduling algorithms at the seed in a BT system and quantified the performance gains obtained thus. A generic analytical framework to model such algorithms was presented and a novel seed scheduling strategy to acheive better content replication was proposed. Through numerical simulations of the model as well as simulations the improved performance of the proposed PFS algorithm over existing strategies in the literature (LRF and the existing mainline random scheduling schemes) was demonstrated.

As a natural extension to this paper, which we are considering as part of our future work, we aim to assess the impact of PFS on a real deployment of a BitTorrent network through measurements using modified clients deployed on Planet Lab.

References

[1] Characterization of Internet traffi c loads, segregated by application, `http://www.caida.org/analysis/workload/byapplication/`.

[2] A. Legout, G. Urvoy-Keller and P. Michiardi, *Rarest First and Choke Algorithms Are Enough*, Technical Report (inria-00001111, version 1 - 13 February 2006), INRIA, Sophia Antipolis, July 2005.

[3] G. Urvoy-Keller and P. Michiardi, *Impact of Inner Parameters and Overlay Structure on the Performance of BitTorrent 9th IEEE Global Internet Symposium 2006*, In Conjunction with IEEE INFOCOM 2006 Barcelona, Spain, 28 - 29 April 2006.

[4] Y. Tian, D. Wu and K. W. Ng, *Modeling, Analysis and Improvement for BitTorrent-Like File Sharing Networks*, Proceedings of IEEE INFOCOM, Barcelona, Spain, August 2006.

[5] D. Qiu and R. Srikant, *Modeling and performance analysis of BitTorrent-like peer-to-peer networks*, Proceedings of ACM SIGCOMM, Portland, OR, August 2004.

[6] B. Fan, D-M. Chiu and J. C. Sl Lui, *Stochastic Differential Equation Approach to Model BitTorrent-like P2P Systems*, ICC 2006, Turkey.

[7] C. Gkantsidis and P. Rodriguez, *Network Coding for Large Scale Content Distribution*, Proceedings of IEEE INFOCOM, Miami, 2005.
[8] B. Cohen, *Incentives Build Robustness in BitTorrent*, Proceedings of First Workshop on Economics of Peer-to-Peer Systems, Berkeley, 2003.
[9] M. Izal, G. Urvoy-Keller, E. W. Biersack, P. Felber, A. A. Hamra and L. Garces-Erise, *Dissecting BitTorrent: Five Months in a Torrent's Lifetime*, Proceedings of PAM'04, Antibes, 2004.
[10] A. Bharambe, C. Herley and V. N. Padmanabhan, *Analyzing and Improving a BitTorrent Network's Performance Mechanisms*, Proceedings of IEEE INFOCOM, Barcelona, 2006.
[11] X. Yang and G. de Veciana, *Service capacity in peer-to-peer networks*, Proceedings of IEEE INFOCOM, pp. 1-11, Hong Kong, China, March 2004.
[12] F. Mathieu and J. Reynier, *Missing Piece Issue and Upload Strategies in Flashcrowds and P2P-assisted Filesharing*, I dont know where, 2006
[13] H. J Kushner and G. Yin, *Stochastic Approximation Algorithms and Applications*, 2nd ed. Berlin, Germany: Springer-Verlag, 2003.
[14] H. J Kushner and P. A Whiting, *Convergence of Proportional-Fair Sharing Algorithms Under General Conditions*, IEEE Transactions on Wireless Communications, Vol. 3, No. 4, July 2004
[15] BitTorrent has continuous request dynamics.
[16] Technical Report.