A Measurement Study of Piece Population in BitTorrent

Cameron Dale
School of Computing Science
Simon Fraser University
Burnaby, BC, Canada
Email: camerond@cs.sfu.ca

Jiangchuan Liu
School of Computing Science
Simon Fraser University
Burnaby, BC, Canada
Email: jcliu@cs.sfu.ca

***Abstract*—BitTorrent is the most popular peer-to-peer software for file sharing, which has contributed to a significant portion of today's Internet traffic. Many measurement studies have been devoted to the BitTorrent system at the peer-level; yet few have examined the microscopic piece-level, in particular, the piece populations. This information is very useful in understanding the dynamics and evolution of BitTorrent swarms, and especially the effectiveness of its rarest-first policy that strives to ensure an even distribution of pieces.**

In this paper, we present a systematic measurement study on the distribution and evolution of the piece population in BitTorrent. Our measurement is based on real BitTorrent data gathered from both the Internet and controlled PlanetLab swarms. The data is collected by multiple administrated clients distributed in different parts of the network, which collectively offer a global view of the piece distribution. We analyze both snapshot data of the near-instantaneous population of pieces in BitTorrent swarms, and long-term data of the evolution of the piece population over several days, especially during the early phases of the swarm's lifetime. Our results validate that the downloading policy of BitTorrent is quite effective from a piece distribution and evolution perspective; yet enhancements are still possible to achieve the ideal piece distribution.

I. Introduction

Among all the peer-to-peer Internet applications available, BitTorrent [1] has become the most popular for file sharing. Recent reports have indicated that half of all the current Internet traffic is due to BitTorrent [2]. One of the reasons it has become so popular is that the sharing is very efficient [3], allowing downloads to scale well with the size of the downloading population. This efficiency is obtained by breaking up each large file into hundreds or thousands of segments, or *pieces*, which, once downloaded by a peer, can be shared with others while the downloading continues.

An important consideration in controlling this sharing is deciding the order of pieces to download. Each peer will have to make this decision based only on the local knowledge it has of the system. An inadequate policy could lead to some pieces becoming poorly replicated, and therefore almost unavailable, while others are overly replicated, leading to starvation in areas of the system where new pieces are needed. To understand how the policy for choosing pieces in BitTorrent affects the system, it is necessary to examine the system-wide population of pieces available. This microscopic information would help to understand the dynamics and evolution of the BitTorrent swarm, and especially the effectiveness of the policy used by BitTorrent to ensure an even distribution of pieces.

Though there have been many recent research projects investigating specifically the sharing present in BitTorrent, most have focused on the macroscopic measurements of peers in the system. Izal et al. [4] gathered and analyzed long-term data from a BitTorrent tracker, but since the tracker has no knowledge of pieces, they did not further explore this information. Pouwelse et al. [5] studied BitTorrent through tracker logs and also a large number of administered clients. However, they were interested mainly in measurements of the uptimes and download rates of the contacted peers. Veciana et al. [6] created a Markov Chain model to numerically study the service capacity of a BitTorrent-like P2P system. This model was expanded on by Qiu and Srikant [3] to create a simple deterministic fluid model for the peer population of the system. Using the model, they also examined the effectiveness of the file sharing in BitTorrent by a peer-level probabilistic model that assumes peers have global knowledge.

In this paper, we present a systematic measurement study on the distribution and evolution of the piece population in BitTorrent. Our measurement is based on real BitTorrent data gathered from both regular Internet and controlled PlanetLab swarms. The data is collected by multiple administrated clients distributed in different parts of the network, which collectively offer a global view of the piece distribution. To the best of our knowledge, the closest work to ours is from Legout et al. [7], who administrated a single client and connected separately to 26 torrent swarms of differing characteristics. Their results thus reflect the piece availability only in peers the single client connected to during the experiment, which may not be representative of the entire swarm, nor does it offer the global knowledge of the piece population.

We examine snapshots of the population of pieces in swarms, and the evolution of the piece population over several days, mostly during the early phases of a swarm's lifetime. We find that the piece distributions are generally very narrow, and progress to a more ideal distribution quite quickly. This shows that the downloading policy of BitTorrent is effective from a piece distribution and evolution perspective, though we do find that some enhancements are possible to achieve an ideal piece distribution, especially for larger torrent swarms.

II. BitTorrent Overview

A BitTorrent *swarm* is the set of all peers currently downloading pieces from each other. It is made up of two types of peers, those who have the complete file (*uploaders* or *seeders*), and those who are still downloading it (*downloaders* or *leechers*). The BitTorrent system coordinates file sharing through the use of a centralized *tracker*. Upon receiving a request from a downloading peer's client, the tracker will provide a random list of peers for the client to contact. The client will then contact each of the peers to gather information about which pieces the peers have available for download.

Throughout the lifetime of a BitTorrent swarm, three phases or states are evident. The first is a *startup state* occurring at the very beginning of the swarm, at which time only the initial seed has all the pieces of the file. Once a single copy of all pieces is uploaded to the swarm, the startup state ends, and a *transient state* begins. The transient state is usually characterized by the rapid influx of downloaders to the swarm, which leads to a system with proportionally many more leechers than seeders. Once this influx slows, the swarm will move towards a *steady state*, characterized by an unchanging number of seeders and leechers, so that the arrival rate of leechers must be the same as (or near to) the rate of change of leechers to seeders and the departure rate of seeds from the system. The amount of time spent in the startup state is determined solely by the upload rate of the initial seed and the size of the file, while the time spent in the transient state is determined by the popularity of the torrent.

A. The Rarest-First Policy

There are many policies at work in a BitTorrent client that govern how it downloads pieces. One of the most important is the *rarest-first* policy, which is responsible for choosing pieces to download with the goal of ensuring that pieces are uniformly distributed throughout the system. The client constantly updates a list of the pieces each of its connected peers has available. Using this information, the client can determine which piece (or set of pieces) it believes to be the rarest in the swarm. These rarest pieces will be selected first to download from the connected peers. Due to the limitations of the local knowledge each peer has, the pieces chosen to download may not be the rarest in the entire swarm.

B. Piece Population in BitTorrent

The piece population is the number of copies of each piece in the BitTorrent swarm. Each piece can have a number of copies varying from the number of seeds in the system to the total number of peers in the system (seeders and leechers). Since all seeds always have all pieces, we will restrict our discussion to the population of pieces among the downloaders in the system. Therefore, each piece in this limited view of the swarm can have a number of copies varying from 0 to the number of downloaders in the system.

The population is expected to form some distribution around a mean value. This mean depends solely on the average completion of the download, which is itself determined by the arrival rate of downloaders and the transition rate of leechers into seeders. During the startup phase, this mean will be low as not all pieces are available yet in the swarm. In the transient state, the mean will be less than half the number of downloaders, as new downloaders are arriving faster than pieces can be copied in the system. In the steady state the mean should be close to half the downloaders, as the average completion will be 50%, as new downloaders arrive to the system at the same rate as downloaders become seeders. Only at the end of the torrent's life will the mean be larger than half the number of downloaders in the system.

Though the policy for choosing pieces to download will not affect the mean of the population distribution, it will have a large effect on the width of the distribution about the mean. Ideally, this distribution width would be very small, signifying that copies are equally distributed to all pieces in the system. It is not hard to imagine far from optimal scenarios where this distribution could range from 0 all the way to the number of downloaders, especially considering the limited knowledge peers have of the system when it is very large. The most problems will be expected in large swarms as some peers will have chosen pieces to download that are not rare in the system, but are only rare for the local view the peer has of the system, creating a tail in the distribution towards a higher number of downloaders.

The distribution width will also vary at different stages of the system, through no fault of the policy itself. For example, early on in the startup phase it is very difficult to keep a small distribution about the mean, when most of the pieces have not yet been copied in the system. Ideally, once this phase of the swarm is complete, the distribution width should narrow very quickly as pieces are preferentially copied that were previously under-represented.

III. Experimental Setup for Measurements

We gathered all experimental data using a modified BitTornado program [8], which is a typical and widely-used BitTorrent client. We modified both the default behavior (through options), and the inner workings of the program. These changes allowed us to log the pieces downloaded by the client, as well as collect data of all the piece information communicated to the client by other connected clients.

We have conducted a series of measurement experiments from December 2006 to May 2007 in both the global Internet and the PlanetLab research network testbed [9].

1) Real Internet Swarms: For these swarms, we used multiple administrated clients to record snapshots of the piece population in a real BitTorrent swarm. This consisted of monitoring the number of peers contacted by our clients to determine when most of the swarm had been contacted, at which point the experiment was terminated. These experiments usually take less than one hour to contact over 90% of the peers in a swarm, which is enough to get a clear view of the global piece population.

We also studied a swarm's changing piece population over time. This is useful to study how the piece population varies

TABLE I
THE TORRENTS USED IN THE EXPERIMENTS

Torrent Name	Pieces	Size (MB)	Leechers	Clients[a]
KNOPPIX[b]	4125	4325	169	10
FreeBSD[b]	5699	1494	34	10
mandriva[b]	2803	735	89	9
openSUSE[b]	14805	3881	398	9
feisty[c]	1387	727	65-120	20
openSUSE-2[c]	14977	3926	100-150	18
PlanetLab[c]	1497	784	0-340	340

[a] Number of administered clients used to connect to the swarm. All peers are administrated clients in PlanetLab experiment
[b] Snapshots of population taken
[c] Evolution of population monitored

with time, especially when the torrent swarm is in an initial transient state. The method is the same as the snapshot, except that the experiment is run for a much longer time.

We chose 6 torrents from well-known Linux distributions that use BitTorrent to distribute their files, which are shown in Table I. They are all freely available online, and also have tracker information available. Their file sizes are quite representative of the file sharing being done by BitTorrent.

2) Simulated PlanetLab Swarm: In the PlanetLab swarm, all the clients are controlled and use the modified BitTornado program, which enables us to have an even closer look at the piece distribution and evolution. We created a sample torrent file that resembles real ones, shown in Table I, and ran a tracker for the purpose of the experiments. The downloading and uploading bandwidths of the clients are set to 75 and 25 KB/s respectively, and we restricted each client's maximum connections to 40 to further enhance the locality effects of the piece population. All other parameters were left at the default values for the client.

IV. MEASUREMENT RESULTS AND ANALYSIS

We now present and analyze our measurement results for piece populations. In order to make comparisons between different swarms, some normalization of the data is needed. For all data, we normalize the number of copies (x-axis) by the total number of downloaders, so that it varies from 0 to 1. If the populations are all from the same swarm, the population size data (y-axis) will be normalized by the number of pieces, so it will also vary from 0 to 1. However, this normalization does not make sense when comparing different swarms' populations, as it leads to a much smaller population when the number of pieces is larger. Therefore, to facilitate the comparison of multiple swarms, they will be normalized so that the area under their population graph is 1.

A. Snapshots of Piece Population

Figure 1 shows the snapshots of the piece populations for various real torrent swarms. All four appear to be normally distributed with mean values slightly less than half the downloaders, indicating that they are in the transient state. The least normally distributed populations are Knoppix and openSUSE, which corresponds to their being the largest swarms. This

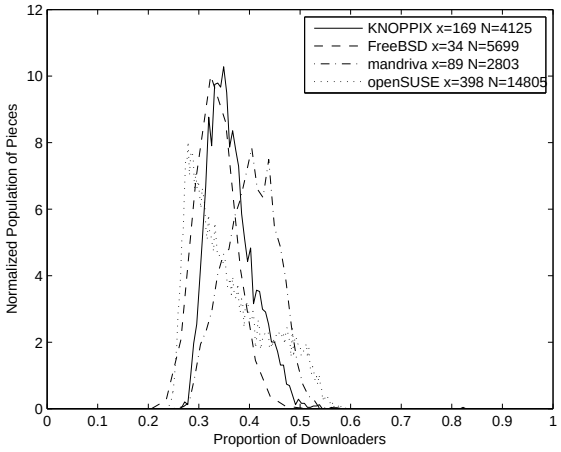

Fig. 1. The piece populations of different torrents (x = number of downloaders, N = number of pieces).

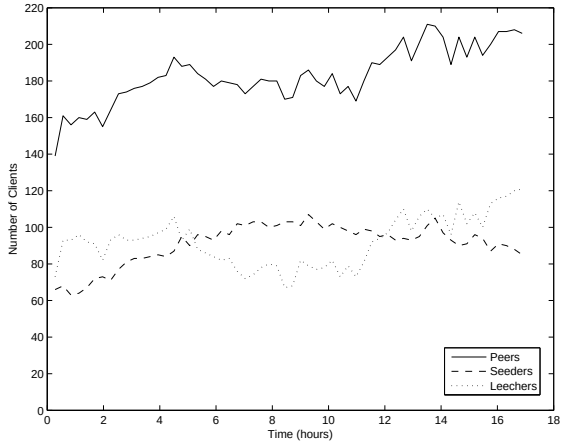

Fig. 2. The number of peers in the feisty swarm.

larger swarm size results in peers having a limited local view of which pieces are rarest, which leads to a distortion of the normal curve towards some pieces having extra copies (the tail evident in figure 1). The other two populations are small enough that a peer's local view is nearly complete, resulting in a near perfect normal distribution.

B. Evolution of Piece Population

To further understand the dynamics of piece population in the different states of the swarm, we have also monitored swarms throughout their lifetime. It is worth noting that such experiments can be difficult to conduct for real Internet swarms because, in general, we do not know the exact start time of a swarm unless it is launched by ourselves. Swarms launched by ourselves, however, are not necessarily representative and the measurement results can be biased. Through constant online tracking, we did find several swarms that we began monitoring very early, and we now show two of them.

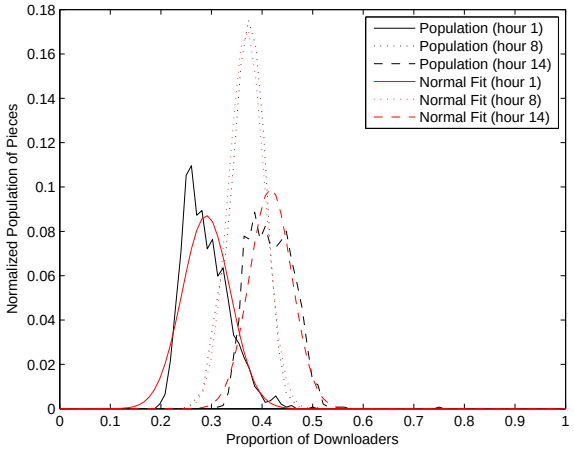

Fig. 3. Selected piece populations from the feisty swarm.

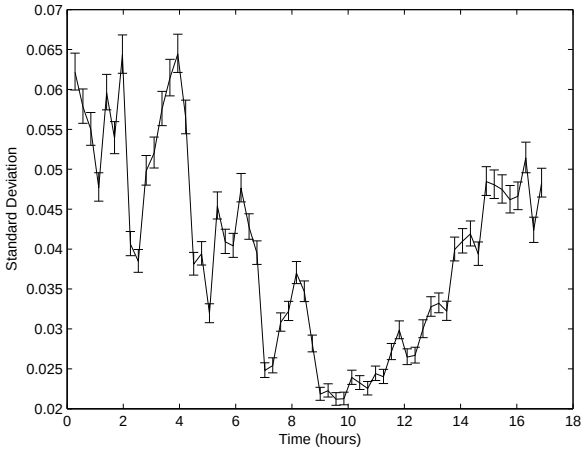

Fig. 5. The standard deviation of the piece population in the feisty swarm (error bars are 95% confidence intervals).

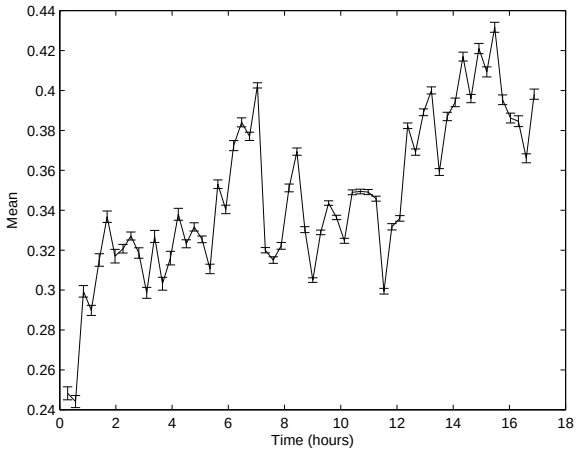

Fig. 4. The mean of the piece population in the Feisty swarm (error bars are 95% confidence intervals).

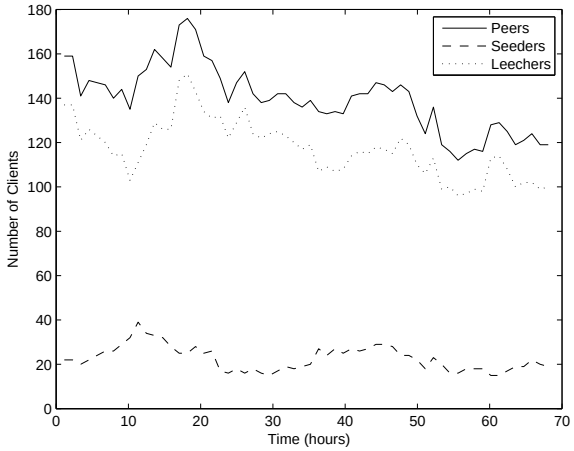

Fig. 6. The number of peers in the openSUSE-2 swarm.

1) The feisty Swarm: Figure 2 shows the evolution of the Feisty swarm over a period of 17 hours. The monitoring began soon after the torrent was launched, and though the number of leechers has already peaked, a peak is clear in the number of seeders near the middle of the experiment. This leads to the conclusion that this swarm was in a transition from the transient state to the steady state as the experiment progressed. However, there was a large influx of peers near the 12 hour mark, probably due to a news posting.

Figure 3 shows three representative plots of the piece population in the Feisty swarm, as well as fitted normal distributions. The piece population is seen to be progressing towards a more normal distribution, and towards a mean closer to 0.5. This progression of the mean can be more clearly seen in Figure 4, though somewhat noisy. Figure 5 shows the progression of the standard deviation towards a more narrow distribution, although the width does increase near the end of the experiment after the large influx of peers occurs.

2) The openSUSE-2 Swarm: Figure 6 shows the evolution of the openSUSE-2 swarm over a period of 70 hours while it was being monitored. The monitoring began very soon after the torrent was launched, as seen by the large number of leechers and few number of seeders. Due to the size of the torrent, not much change is seen in the swarm during the experiment, though the number of leechers decreases without an increase in the number of seeders, indicating that seeders are leaving the system at the same rate as peers are becoming seeders while few new peers are joining the system. This system is considered to still be in the transient state, and is in an earlier state than the Fiesty swarm was.

Figure 7 shows three representative plots of the piece population in the openSUSE-2 swarm, as well as fitted normal distributions. The piece population is very clearly seen to be progressing towards a more normally distributed shape, as the population shown for hour 2 has a very large tail. The mean is omitted due to its similarity to figure 4, as it is also clearly

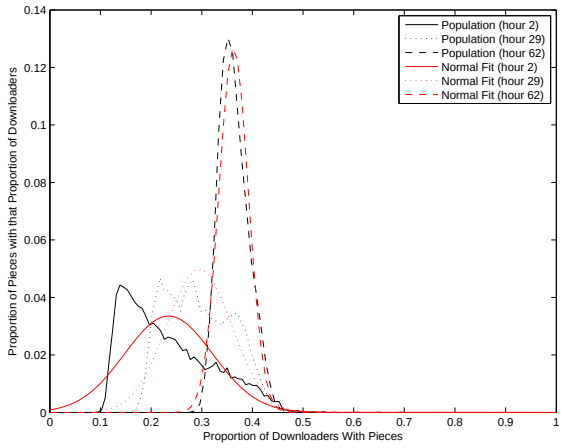

Fig. 7. Selected piece populations from the openSUSE-2 swarm.

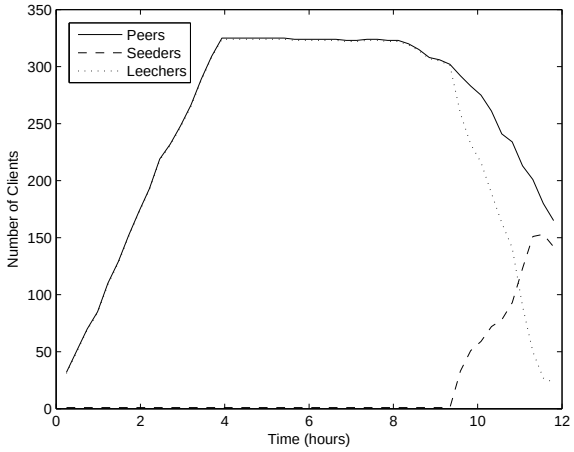

Fig. 9. The number of peers in the PlanetLab swarm.

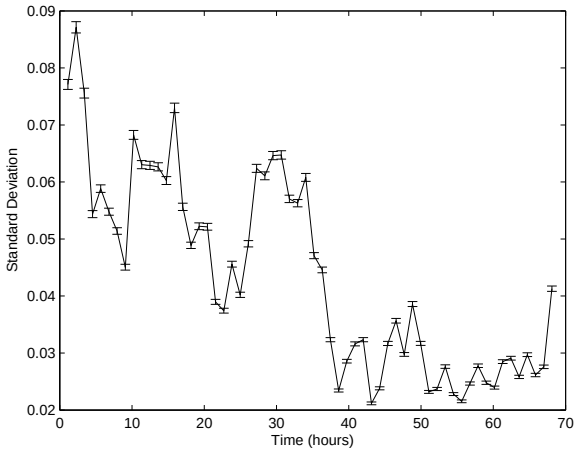

Fig. 8. The standard deviation of the piece population in the openSUSE-2 swarm (error bars are 95% confidence intervals).

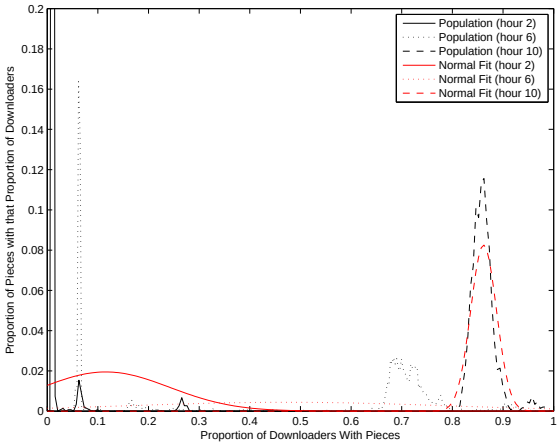

Fig. 10. Selected piece populations from the PlanetLab swarm.

seen to be progressing towards higher values. Figure 8 also shows the clear progression of the standard deviation towards a more narrow distribution of pieces.

C. Results of the PlanetLab Swarm

The controlled PlanetLab environment enables us to closely investigate the piece population of a swarm in any period throughout its lifetime. Figure 9 shows the evolution of the PlanetLab swarm over the period of 12 hours that we simulated it. This system is in the startup state (as seen by the single seed that is available) through most of the experiment, transitioning to the transient state after approximately 9 hours. The system then moves quickly to an end state not seen in the other real swarms, as no new clients join the system but many are completing their download, and many are leaving the system.

Figure 10 shows three representative plots of the piece population in the PlanetLab swarm, as well as fitted normal distributions. The normal distribution does not match the first two plots at all, as most pieces suffer from a low replication rate, while peaks higher in the population show some pieces have a much higher replication rate. This is due to the limited upload bandwidth of the original seed, and the time it takes for a single copy of the file to be present in the network (approximately 9 hours). However, the third population at 10 hours shows that the swarm takes very little time to become very narrowly distributed after the first full copy of all pieces are present in the network.

The mean in this experiment (also not shown) progresses linearly towards a higher value, finishing very close to 1. Figure 11 shows the rapid progression of the standard deviation towards a very narrow distribution of pieces around the mean value once the first copy of the file is present in the network (at 9 hours). The wide distribution of the piece population in the first 6 hours is expected, as the piece population goes from an initial state of very narrow (all pieces have no copies), to a split population, and finally to a normal distribution of narrow size soon after all pieces enter the system.

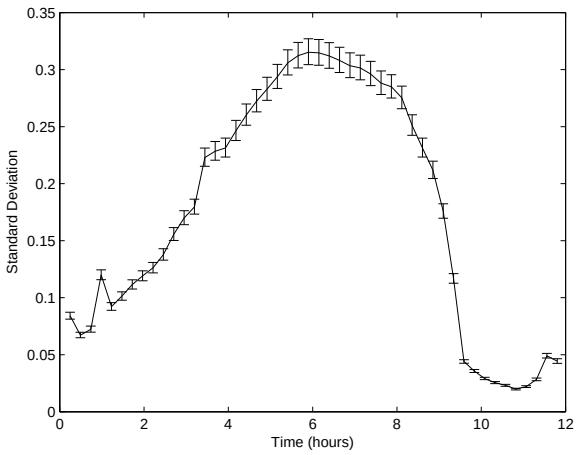

Fig. 11. The standard deviation of the piece population in the PlanetLab swarm (error bars are 95% confidence intervals).

D. Summary and Further Discussion

As we have seen, the snapshots of piece populations roughly follow normal distributions with means near half the number of downloaders. Their widths were small, and the tails expected for the larger swarms were small as well, indicating the effectiveness of the rarest-first policy at replicating pieces uniformly using only the local information available to each peer. Note that this is the same conclusion reached by Legout et al. [7], but in this case we did not rely on the somewhat questionable definitions of *entropy* and *peer availability* to reach that conclusion, as we have the global piece data needed.

The population evolution over time showed similar shapes. In addition, as the swarm progressed out of the transient state and into the steady state, we have seen a clear progression in the shape of the piece population. This progression leads to a more ideal shape, as the mean increased and the standard deviation decreased.

In the PlanetLab measurement the population of the early stages in a swarm's lifetime was quite poor at matching the ideal narrow distribution. This is attributable to the poor availability of some pieces in the system, which have not yet been uploaded by the initial seed. This changed very quickly after a single copy of the file was available, indicating that the policy works well once this startup phase is complete. Once the last piece is available in the system, the swarm quickly replicates the last pieces uploaded so that they can "catch up" to the replication that earlier pieces have already had.

In summary, the rarest-first policy employed by BitTorrent is fairly successful, both in moving quickly to a narrow distribution, and in maintaining and even improving that distribution as the swarm progresses. There are concerns though, as both the early startup phase and some of the larger swarms show increased width due to the limited piece availability and the limited knowledge of the system a peer has. The first is not easy to improve, as any policy will be hampered by the limited selection of pieces available in the system. However, the tails seen on some of the larger swarms' population distributions could be improved upon by increasing the amount of knowledge each peer has of the system. For example, when communicating the pieces each peer has to its neighbors, extra information could be included such as the rarest pieces seen by the peer or the number of copies each piece has in it's limited local view. This new information, combined with the knowledge a client already has of its neighboring pieces, is believed to be enough to reduce the size of the tail on the population distribution by an order of magnitude at very little cost in extra communication.

V. Conclusions

In this paper, we have presented measurements on the piece populations in BitTorrent swarms, and investigated the effectiveness of the rarest-first policy for piece replication from a piece distribution and evolutionary perspective. We have shown that the policy is quite effective once all pieces become available in the system, and throughout the lifetime of the swarm. However, some deviations from the ideal were apparent soon after creation of the swarm, and in some of the larger swarms studied.

We are currently conducting measurements of ultra-large swarms to determine the size that population tails can grow to. We are also working on PlanetLab experiments consisting of a large number of peers joining after the startup phase, and progressing through a transient to a steady state.

With these results, we plan to modify the rarest-first policy to determine if it is effective in reducing the size of the tail. Furthermore, we are building piece-level models describing the piece population of a torrent swarm, which will better explain the pros and cons of the rarest-first policy, as well as facilitate our modifications. Our preliminary modeling results can be found in our technical report [10].

References

[1] B. Cohen. (2003, May) Incentives build robustness in BitTorrent. [Online]. Available: http://bitconjurer.org/BitTorrent/bittorrentecon.pdf
[2] (2004) CacheLogic. [Online]. Available: http://www.cachelogic.com
[3] D. Qiu and R. Srikant, "Modeling and performance analysis of BitTorrent-like peer-to-peer networks," in *Proc. SIGCOMM '04*, Portland, Oregon, USA, Aug. 30–Sep. 3, 2004.
[4] M. Izal, G. Urvoy-Keller, E. Biersack, P. Felber, A. A. Hamra, and L. Garces-Erice, "Dissecting BitTorrent: Five months in a torrent's lifetime," in *Passive and Active Measurements*, Antibes Juan-les-Pins, France, Apr. 2004.
[5] J. A. Pouwelse, P. Garbacki, D. H. J. Epema, and H. J. Sips, "The BitTorrent P2P file-sharing system: Measurements and analysis," in *Proc. 4th International Workshop on Peer-to-Peer Systems*, Ithaca, NY, USA, Feb. 2005.
[6] G. de Veciana and X. Yang, "Fairness, incentives and performance in peer-to-peer networks," in *Proc. Forty-first Annual Allerton Conference on Communication, Control and Computing*, Monticello, Illinois, USA, Oct. 2003.
[7] A. Legout, G. Urvoy-Keller, and P. Michiardi, "Rarest first and choke algorithms are enough," in *Proc. IMC'06*, Brazil, Oct. 2006.
[8] (2007) The BitTornado website. [Online]. Available: http://www.bittornado.com/
[9] (2007) The PlanetLab website. [Online]. Available: http://www.planet-lab.org/
[10] C. Dale and J. Liu, "Modeling piece population in BitTorrent," Simon Fraser University, Tech. Rep., 2007.